О.Г. Руденко, Є.В. Бодянський

Штучні нейронні мережі

О.Г. Руденко, Є.В. Бодянський

Штучні нейронні мережі

Рекомендовано Міністерством освіти і науки України
як навчальний посібник для студентів
вищих навчальних закладів

«Компанія СМІТ»
Харків
2006

УДК 519.71
ББК 004.338.8
Р 83

Рекомендовано Міністерством освіти і науки України як навчальний посібник для студентів вищих навчальних закладів, які навчаються за спеціальностями «Інтелектуальні системи прийняття рішень», «Комп'ютерні системи і мережі» (лист № 14/18.2-2756 від 06.12.2005 р.)

Видано за рахунок державних коштів. Продаж заборонено

Рецензенти:

В. Д. Дмитрієнко, доктор техн. наук, професор
(Національний технічний університет «ХПІ»);

О. Ю. Соколов, доктор техн. наук, професор
(Національний аерокосмічний університет ім. М. Є. Жуковського «ХАІ»);

Е. Г. Петров, доктор техн. наук, професор
(Харківський національний університет радіоелектроніки)

Руденко О. Г., Бодянський Є. В.

Р 83 Штучні нейронні мережі: Навчальний посібник. — Харків: ТОВ «Компанія СМІТ», 2006. — 404 с.

ISBN 966-8530-73-X

У навчальному посібнику викладено основні принципи побудови штучних нейронних мереж (ШНМ) як самостійного напрямку в теорії інтелектуальних систем, подано біологічний аналог штучного нейрона і процеси обробки інформації в біологічних системах. Наведено різноманітні моделі штучних нейронів і розглянуто властивості мереж, побудованих на їх основі, починаючи від персептрона і закінчуючи новітніми розробками в цій галузі. Значну увагу приділено методам навчання ШНМ, питанням раціонального вибору і спрощення їх архітектури. Окремий розділ посібника присвячений розгляду прикладного аспекту використання нейромережевих технологій (фінансове прогнозування, адаптивне управління складними об'єктами за умов невизначеності, обробка відео- та мовних сигналів, фільтрація і стиснення інформації тощо).

Для студентів, аспірантів і фахівців, які займаються створенням сучасних способів обробки інформації.

ISBN 966-8530-73-X

ПЕРЕДМОВА

Виникнення штучних нейронних мереж (ШНМ) пов'язане з розумінням того, що мозок живого організму працює інакше, ніж комп'ютер. Мозок людини — це дуже складна нелінійна паралельна інформаційно-керуюча система, здатна до мислення, нагромадження й відновлення інформації, вирішення проблем. Ця система складається з досить однотипних будівельних блоків — нервових клітин, або нейронів, що являють собою прості елементи обробки сигналів, які отримують і комбінують інформацію від інших нейронів через входи — дендрити.

З інженерної точки зору ШНМ — це паралельно розподілена система обробки інформації, утворена тісно зв'язаними простими обчислювальними вузлами (однотипними або різними), що має властивість накопичувати експериментальні знання, узагальнювати їх і робити доступними для користувача у формі, зручній для інтерпретації й прийняття рішень.

На сьогодні є всі підстави говорити про досягнення певних успіхів нейромережевих технологій у вирішенні складних завдань як суто наукових, так й у сфері техніки, бізнесу, фінансів, медичної діагностики й інших галузей, пов'язаних з інтелектуальною діяльністю.

В основу книги покладено матеріали з курсів, прочитаних авторами протягом декількох років у Харківському національному університеті радіоелектроніки.

Ця книга написана в першу чергу для студентів, що цікавляться ШНМ і мають намір використати їх у різних застовуваннях. Зазначимо, що нейромережі, однак, не є панацеєю, і перед тим як використовувати їх для вирішення практичних завдань, необхідно ще раз оцінити їхні переваги й недоліки. Теорія штучних нейронних мереж інтенсивно розвивається, й ми, природно, не можемо висвітлити всі її аспекти. У посібнику розглянуто основні архітектури мереж і парадигми навчання. Відсутність у ньому такого важливого класу ШНМ, як нейро-фаззі-мережі пояснюється тим, що для його опису необхідно ознайомитися з апаратом нечіткої логіки, що зазвичай не читається у вищих навчальних закладах. Поглибленню знань у галузі ШНМ має сприяти звертання до наявної у посібнику бібліографії.

Автори висловлюють подяку О. Бессонову за проведене моделювання ШНМ і кропітку роботу з набору й оформлення рукопису.

Вважаємо своїм обов'язком виразити вдячність Deutscher Akademischer Austauschdienst (DAAD) і Deutsche Forschungsgemeinschaft (DFG) за здійснену ними підтримку, що дала нам можливість відвідати ряд університетів ФРН, ознайомитися із сучасними досягненнями в галузі нейронних мереж й обмінятися думками із зарубіжними колегами.

Автори

ВСТУП

Уже ні в кого не викликає здивування проникнення комп'ютерів практично в усі сфери людської діяльності. Удосконалювання елементної бази, що визначає архітектуру комп'ютера, і розпаралелювання обчислень дозволяють швидко й ефективно вирішувати задачі все зростаючої складності. Вирішення багатьох проблем немислиме без застосування комп'ютерів. Однак, маючи величезну швидкодію, комп'ютер часто не в змозі впоратися з поставленим перед ним завданням так, як це робить людина. Прикладами подібних завдань є розпізнавання (наприклад, знайоме обличчя людина впізнає за 100–120 мс, а найсучасніший комп'ютер — за хвилини або години), розуміння мови й тексту, написаного від руки тощо. Таким чином, мережа нейронів, що створює мозок людини, будучи, як і комп'ютерна мережа, системою паралельної обробки інформації, у багатьох випадках є більш ефективною. Ідея переходу від обробки закладеним у комп'ютер алгоритмом деяких формалізованих знань до реалізації в ньому властивих людині прийомів обробки інформації (розумової діяльності) призвели до появи штучних нейронних мереж (ШНМ).

Характерною рисою біологічних систем є адаптація, завдяки якій такі системи в процесі навчання розвиваються й здобувають нові властивості. Як і біологічні нейронні мережі, ШНМ складаються із з'єднаних між собою елементів, штучних нейронів, функціональні можливості яких тією чи іншою мірою відповідають елементарним функціям біологічного нейрона. Як і біологічний прототип, ШНМ має такі властивості:

- адаптивне навчання: здатність поліпшувати свої характеристики, закладені у тому або іншому алгоритмі настроювання параметрів мережі, що відпрацьовує подані їй послідовності, що навчають, або використовує набутий досвід;
- самоорганізація: ШНМ здатні змінювати свою структуру (архітектуру) або форму подання інформації;

- узагальнення: після завершення процесу навчання мережа може бути нечутливою до незначних змін вхідних сигналів, що дозволяє застосовувати її при зашумлених або не повністю заданих даних;
- обчислення в реальному часі: нейромережеві обчислення можуть здійснюватися паралельно в часі, що істотно збільшує швидкодію ШНМ;
- стійкість до перебоїв: часткове руйнування мережі призводить до втрати якості, однак деякі її властивості зберігаються навіть у випадку руйнування більшої частини мережі.

Незважаючи на таку подібність, ШНМ ще дуже далекі від дублювання властивостей мозку людини. Однак дивна подібність функціонування деяких ШНМ з їхніми біологічними прототипами наводить на думку про можливості проникнення в людський інтелект уже в близькому майбутньому.

Новітня історія ШНМ бере свій початок з 1943 р. — з моменту виходу праці *У. Маккаллоха* та *У. Піттса* [1], у якій досліджено властивості найпростішої моделі нейрона із двома стійкими станами (модель Маккаллоха — Піттса), на вхід якої надходить прискорювальний і гальмуючий сигнали. Вони довели, що за допомогою такої моделі можуть бути реалізовані різні логічні функції, наприклад, І, АБО, НЕ та ін.

У 1949 р. *Д. Гебб* [2] не тільки встановив, що пам'ять у біологічних системах викликається процесами, що змінюють зв'язки між нейронами, але й запропонував правило зміни цього зв'язку — правило навчання нейрона, назване його ім'ям. І сьогодні правило Гебба в тій або іншій формі присутнє в багатьох алгоритмах навчання ШНМ.

Підбадьорюючими були й результати першого комп'ютерного моделювання ШНМ, проведеного в 1956 р. під керівництвом *Н. Рочестера* [3]. Цей рік вважається роком народження наукового напрямку, що має назву «штучний інтелект». В основу модельованої ШНМ покладено вдосконалену модель Гебба. Дана робота дала поштовх численним роботам з моделювання нейронних мереж.

Найбільш важливі результати для розвитку ШНМ були отримано в цей період у Массачусетському технологічному інституті групою вчених під керівництвом *Ф. Розенблатта* [4]. Ними не тільки досліджено різноманітні варіанти мереж, названі персептронами, та вивчено різні способи їх навчання, але й здійснено технічну реалізацію персептрона. Синтезований ними персептрон

використав граничну логіку, складався із трьох шарів і був здатен вирішувати прості завдання розпізнавання цифр. Більш докладно властивості персептрона ми розглянемо нижче. Зазначимо тільки, що робота *Ф. Розенблатта* [5], у якій наведено різні варіанти персептрона й доведено теорему збіжності для персептрона, викликала величезний інтерес до ШНМ.

Водночас під керівництвом Б. Уідроу розвивався напрямок, пов'язаний із практичним застосуванням ШНМ, побудованих на елементах ADALINE (ADAptive LInear NEuron), аналогічних персептрону. Мережа, що містить багато таких елементів, була названа MADALINE (Multiple ADALINE's). Для навчання такої мережі було використано алгоритм, що мінімізував квадратичну помилку навчання — алгоритм *Уідроу — Гоффа* [6, 7]. Слід зазначити, що згодом Б. Уідроу заснував фірму, що займається розробкою електронних компонентів для нейрокомп'ютерів.

Райдужним надіям щодо систем штучного інтелекту, компонентами яких були ШНМ, поклала кінець монографія *М. Мінські* й *С. Пейперта* [8], присвячена серйозному аналітичному дослідженню властивостей персептрона, що з'явилася у 1969 р. Автори показали, що можливості персептрона (мається на увазі одношаровий) обмежуються реалізацією тільки найпростіших логічних функцій, і то не всіх. Наприклад, з його допомогою неможливо організувати функцію «що виключає АБО» тощо. Хоча дослідники й розуміли, що можна будувати персептрони, що містять більше одного шару й мають, вочевидь, ширші можливості, зовсім незрозуміло було, як можна навчити приховані шари. Тому монографія М. Мінські й С. Пейперта викликала справжній шок серед учених і стала настільки сильним ударом по дослідженнях у цій галузі, що такі роботи практично припинилися.

Увага дослідників переключилася на інші види мереж. У 80-ті роки XX ст. досягнуто значних успіхів у такій галузі ШНМ, як асоціативна пам'ять. Піонерськими тут є роботи *Т. Когонена* [9–14] і *Дж. Андерсона* [15, 16]. Як модель Т. Когонен використав так званий одношаровий лінійний асоціатор, а навчання здійснював за правилом Гебба, при якому деякий пропонований мережі образ асоціювався з іншими. Дж. Андерсон, вирішуючи завдання побудови асоціативної пам'яті, що здійснює пошук інформації, яка зберігається, не за її адресою, а за змістом, також користувався правилом навчання Гебба для побудови матриць, що служать для подання збережених у пам'яті й виклику необхідних образів.

Важливою подією в теорії й практиці ШНМ стала поява алгоритму навчання багатошарових мереж, який отримав назву ***алгоритму зворотного поширення помилки***. Вперше запропонований у 1974 р. *П. Вербосом* у його дисертації [17] метод виявився непоміченим. Непоміченою виявилася й праця *Д. Паркера* [18], що вийшла в 1985 р. Лише після появи в 1986 р. праці *Д. Румельхарта, Г. Гінтона* й *Р. Уільямса* [19] цей алгоритм отримав широке розповсюдження. Алгоритм забезпечує мінімізацію вихідної помилки мережі шляхом настроювання вагових коефіцієнтів усіх шарів ШНМ, починаючи з останнього, переходячи до передостаннього й т. д., аж до вхідного шару. Таким чином, під час його роботи помилка немовби поширюється від виходу мережі на її вхід, чим і пояснюється назва алгоритму. Розробка цього алгоритму дозволила не тільки перебороти обмеженість одношарового персептрона й конструювати багатошарові мережі, що мають значно більші можливості, але й дала новий імпульс розвитку ШНМ. Поступово з'явився необхідний для конструювання потужних багатошарових мереж теоретичний фундамент. Оцінка М. Мінські виявилася надто песимістичною, і багато хто з поставлених у його книзі завдань вирішуються в цей час мережами за допомогою стандартних процедур.

Якщо перші структури ШНМ були простими й використовували лінійні статичні моделі нейронів, то з часом сконструйовано мережі, зокрема динамічні, що мають більш складні структури й здатні вирішувати значно складніші завдання. Особливий інтерес викликали праці *С. Гроссберґа* [20–22]. Так, займаючись дослідженням динамічних властивостей мереж, він разом з М. Коеном довів фундаментальну теорему про глобальну збіжність динамічних мереж — теорему Коена — Ґроссберґа. Дослідження біологічних об'єктів дозволили Ґроссберґу і його колегам установити дилему стабільності — пластичності (запам'ятовування нової інформації без руйнування тієї, що вже зберігається у пам'яті) і сконструювати мережу, що володіє такими властивостями. Ця мережа отримала назву ART (*Adaptive Resonance Theory*). Робота [22] поклала початок цілій низці публікацій С. Ґроссберґа і його колег, присвячених розробці й дослідженню різних видів ART: ART-1, ART-2, ART-3, ART MAP, Fuzzy-ART.

Величезну роль у розвитку ШНМ відіграли й праці фізика *Дж. Гопфілда* [23–25], у яких на основі функцій Ляпунова досліджувалася проблема стійкості симетричних рекурсивних мереж. У цих працях мережа вперше характеризувалася деякою енергетичною функцією, мінімуми якої відповідають збереженим образам. Сконструйована Дж. Гопфілдом мережа, що носить його ім'я,

здатна вирішувати дуже складні оптимізаційні завдання, наприклад відома задача про комівояжера, що й було продемонстровано в його спільній з *Д. Танком* роботі [25]. Слід зазначити, що підкреслена практична спрямованість робіт Дж. Гопфілда послужила основою того, що вже в 1987 р. під його керівництвом був створений ***нейрочип***.

Перша переконлива перевага використання ШНМ під час вирішення складних практичних завдань продемонстрована роботою *Т. Сейновського* й *Ч. Розенберга* [26] на прикладі автоматичного навчання англійській мові.

Сьогодні існує вже достатньо типів ШНМ, які дозволяють успішно вирішувати завдання розпізнавання образів і мови, класифікації, побудови математичних моделей і керування, оптимізації й прогнозування. З деякими з основних типів ШНМ ми й ознайомимося далі.

Галузі застосування ШНМ

ШНМ знаходять сьогодні широке застосування у будь-яких галузях, починаючи від завдань апроксимації функцій і закінчуючи створенням нейрокомп'ютерів. Важко описати всі можливі сфери застосування нейронних мереж, тому ми коротко зупинимося лише на деяких з них.

Апроксимація функцій

Установлення універсальних апроксимуючих властивостей ШНМ стало досить важливим етапом у становленні загальної теорії й стимулювало дослідження в даній галузі. Багатьма дослідниками було встановлено, що нейронна мережа з одним прихованим й одним вихідним шаром здатна апроксимувати з будь-якою наперед заданою точністю на компактній множині будь-яку неперервну функцію. Так, *Р. Гехт-Нільсен* [27, 28], спираючись на теорему Колмогорова — Арнольда [29, 30] про подання неперервних функцій декількох змінних у вигляді суперпозицій неперервних функцій одного змінного, довів можливість апроксимації функцій багатьох змінних досить загального виду за допомогою двошарової нейронної мережі із прямими повними зв'язками з фіксованою кількістю нейронів з заздалегідь відомими обмеженими функціями активації. *Дж. Цибенко* й *К. Хорник* [31–33], використовуючи теорему Хана-Банаха, довели, що кінцева лінійна комбінація фіксованих одновимірних функцій може однозначно апроксимувати будь-яку неперервну функцію n дійсних змінних на заданому

гіперкубі при досить м'яких припущеннях щодо функцій одного змінного.

Асоціативна пам'ять

Існують архітектури ШНМ, які запам'ятовують образи, що надходять на них, з чимось їх асоціюючи, а при пред'явленні деякого «асоціативного» образу витягують їх з пам'яті. Ця властивість дозволяє організовувати пошук інформації не за адресою, а за її змістом. Навіть якщо пропонована асоціація, що відновлюється, буде спотворена перешкодами, мережа може видати правильний результат. Залежно від того, чи збігається шуканий образ зі збереженим у пам'яті, чи ні, розрізнюють авто- і гетероасоціативну пам'ять.

Дослідженню асоціативної пам'яті присвячена монографія [10].

Стиснення даних

Деякі типи ШНМ мають властивості, що дозволяють використовувати ці мережі для стиснення даних, наприклад перед їхньою передачею, зменшуючи тим самим кількість переданих бітів інформації. Подібні завдання виникають і в кластерному аналізі, коли різні, схожі за певними ознаками образи об'єднуються в деякі групи або кластери, тобто здійснюється перехід від вихідного n-вимірного простору образів до m-вимірного простору кластерів, де $m < n$. Подальша робота в просторі меншої розмірності призводить до економії обчислювальних ресурсів і зменшення обсягу необхідної пам'яті.

Застосування ШНМ для стиснення даних у системах розпізнавання мови й зображень забезпечує стиснення даних у 100 і більше разів [34].

Розпізнавання та класифікація

Розпізнавання образів (зображень, зокрема текстів, друкованих і рукописних, звуку, мови тощо) є тією галуззю, де найбільш яскраво виявляються переваги ШНМ. Вирішення багатьох задач розпізнавання образів ускладнено внаслідок їхньої високої розмірності. Як ми вже зазначали раніше, використання ШНМ шляхом стиснення даних дозволяє знизити розмірність задачі, зберігаючи властивості подільності розподілів, що відповідають різним класам.

Багато важливих застосувань теорії розпізнавання образів відносяться до задач класифікації кривих і геометричних фігур. Такими є, наприклад, завдання діагностики, виявлення несправностей тощо. І в цьому випадку ШНМ дають ефективне вирішення

завдання незалежно від того, існує навчальна множина вже класифікованих об'єктів чи не існує. Однак наявність такої інформації прискорює процес пошуку вирішення [35, 36].

Оптимізаційні задачі

Більшість практично важливих задач можуть бути сформульовані як оптимізаційні, що доставляють екстремум деякому заздалегідь обраному критерію. Тут, однак, йтиметься тільки про одну таку задачу, задачу про комівояжера, що має величезне практичне значення й вирішення якої вимагає значних обчислювальних витрат. Ця задача полягає в тому, що комівояжер має відвідати задану кількість міст, вибравши для цього найкоротший маршрут, причому в кожному місті він повинен побувати не більше одного разу. Така задача виникає, наприклад, під час створення автоматичних технологічних ліній (свердлення отворів у друкованих платах, трасування друкованих плат та ін.), визначення оптимальних маршрутів перевезення тощо.

Доведено, що ця задача відноситься до класу «NP-повних» (недетерміністськи поліноміальних), кращим методом вирішення яких є повний перебір можливих варіантів. Оскільки у випадку n міст існує $n!$ варіантів обходу, очевидно, що із збільшенням кількості міст складність вирішення завдання різко зростає. Одне з можливих ефективних вирішень цієї задачі за допомогою ШНМ, що вимагає значно менших обчислювальних витрат і при цьому збігається з розв'язком, отриманим повним перебором, було запропоновано в роботі [25].

Керування складними процесами

Проблема синтезу ефективної системи керування є досить складною, оскільки реальні процеси характеризуються, як правило, нелінійними залежностями, високим рівнем шумів та їх корельованістю, які змінюються умовами функціонування, що обумовлюють зміну характеристик досліджуваних об'єктів тощо. Необхідність вирішення задач керування в реальному часі висуває певні вимоги як до самих алгоритмів керування, що входять до складу математичного забезпечення проектованої системи, так і до технічних засобів, що їх реалізують. У цих умовах найбільш ефективними виявляються методи й алгоритми, що базуються на теорії адаптації. Однак, як і при будь-якому підході, ці методи вимагають розробки математичних моделей досліджуваних об'єктів.

Слід зазначити, що отримані математичні моделі використовуються не тільки з метою безпосередньо керування, але й для

упередження поведінки об'єкта, що дозволяє підвищити ефективність керування шляхом завчасної корекції керованих параметрів. Якщо одержання математичної моделі ускладнено або вимагає істотних зусиль, доцільне застосування ШНМ. Використанню ШНМ у системах керування присвячені, наприклад, монографії [37–42].

Прогнозування

Будь-яке прогнозування (екстраполяція) спирається на формалізоване уявлення про існуючий зв'язок між причинами й наслідком. Багато процесів формуються під впливом великої кількості факторів, що діють у різних напрямках і нерідко невідомих. Статистичний аналіз цих процесів містить дослідження взаємозв'язків факторів як у статичному стані, так і в часі. Інформацією для вивчення взаємозв'язків служать часові ряди показників, що характеризують розвиток об'єктів. Найпоширенішим підходом до вирішення задачі прогнозування є екстраполяція чинних у цей час зв'язків і закономірностей на майбутнє. Побудовані відповідно до цього принципу моделі прогнозування відрізняються одна від іншої лише гіпотезами про конкретні види збережених зв'язків. Чим більш загальні припущення закладені у форму моделі й чим більший клас процесів можна описати з її допомогою, тим ширше її можливості під час дослідження окремої реалізації.

Таким чином, вибір і основа математичної моделі є центральним моментом прогнозування. На практиці ж нерідко виявляється, що внаслідок тих чи інших причин отримати математичну модель, яка адекватно відбиває властивості досліджуваного об'єкта, надзвичайно складно. І в цих випадках ефективним виявляється використання ШНМ.

У роботах [37, 43–46] наведено численні приклади успішного застосування ШНМ для вирішення завдань економічного, зокрема, фінансового прогнозування.

Нейрокомп'ютери

Нейрокомп'ютери є обчислювачами нового класу, що забезпечують більш швидке, більш дешеве та якісне вирішення конкретних завдань. Нейрокомп'ютер реалізує ідею створення аналого-цифрової ЕОМ, у якій аналогова частина виконує багатовимірні операції в граничному базисі, а цифрова реалізує алгоритми настроювання параметрів нейронних мереж [47].

Деякі порівняльні характеристики комп'ютера й ШНМ наведено в табл. В.1.

Таблиця В.1

	Комп'ютер	ШНМ
Підхід до вирішення задачі	Дедуктивний (використовує відомі правила перетворення вхідних даних у вихідні)	Індуктивний (задані вхідні й вихідні дані використовуються для конструювання правил)
Обчислення	Централізовані, синхронні, послідовні	Розподілені, асинхронні, паралельні
Пам'ять	Пакетована, зосереджена й адресована за розташуванням	Розподілена, адресована за змістом
Стійкість до перебоїв	Нестійкий (вихід з ладу одного елемента призводить до виходу з ладу всього пристрою)	Стійка за рахунок надлишковості й поділу функцій
Швидкодія	Висока (млн оп./с)	Низька (тис. оп./с)
Точність	Висока	Невисока
Архітектура	Фіксована	Що змінюється

Висока швидкодія нейрокомп'ютерів досягається за рахунок розпаралелювання обчислень. Кардинальним напрямком розвитку нейрокомп'ютерів як загального призначення, так і проблемно-орієнтованих є розробка нейрочипів. Досягнення в мікроелектроніці вселяють упевненість у тому, що незабаром будуть створені могутніші й водночас більш дешеві нейронні ЕОМ [47, 48].

1. БІОЛОГІЧНІ ОСНОВИ НЕЙРОННИХ МЕРЕЖ

Уперше думка про те, що живі організми складаються з клітин, була сформульована в 1838 р. *Маттіасом Шлейденом* із Берліна, а в 1839 р. він разом із *Т. Шванном* опублікував працю, у якій було сказано, що «існує універсальний принцип утворення організмів... [який] може бути позначений терміном клітинна теорія» (*перекл. ред.*) [49–50][1]. Упродовж багатьох років (більшу частину XIX століття) тривали суперечки про те, чи застосовна клітинна теорія до нервової системи. Незважаючи на здобуті на той час досягнення, одиночна нервова клітина не розглядалася як єдине ціле. Ключ до мікроскопічного дослідження нервової системи дав лікар *К. Ґольджі*, розробивши в 1875 р. метод вибіркового фарбування нервової тканини, при якому повністю зафарбовується лише невелика частина клітин (і донині ніхто не знає, як і чому спрацьовує метод Ґольджі, забарвлюючи повністю одну зі 100 клітин і зовсім не зачіпаючи всі інші). Скориставшись методом Ґольджі, іспанський гістолог *С. Рамон-і-Кахал* установив, що, по-перше, нервова система є сукупністю окремих, відокремлених клітин, які сполучаються одна з одною за допомогою синапсів, і, по-друге, неймовірно складні зв'язки між нервовими клітинами не випадкові, а високо структуровані й специфічні. Він вивів також основні принципи, згідно з якими нервові сигнали випливають як по дендритах, так і по аксонах клітин, і передача сигналів між клітинами здійснюється в місцях контактів аксонів з дендритами. Зазначимо, що електрична природа нервового імпульсу встановлена *Е. де Буа-Раймондом* і *Г. фон Гельмгольцем* ще в 1850 р.

У 1891 р. *В. Вальдеєр* вперше ввів термін «нейрон», що в перекладі з грецької означає «нерв», запропонувавши назвати так нервову клітину. З цього часу клітинна теорія, застосовувана до нервової системи, стає відомою як «нейронна доктрина».

[1] У даному розділі використані матеріали монографій [49–51].

У рамках цієї доктрини *З. Фройд* пояснював проходження електричного сигналу через клітину й, використовуючи моделі нейронів (рис. 1.1), намагався з'ясувати механізми пам'яті, мислення, задоволення й т. д. [52].

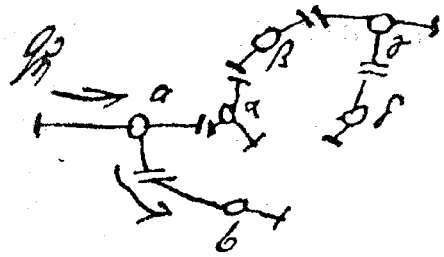


Рис. 1.1. Модель З. Фройда

1.1. Мозок людини

Характерною рисою мозку людини є, по-перше, різноманітність високоспеціалізованих дій, яким він здатний навчатися й, по-друге, розподіл функцій між півкулями. Передбачається, що перша особливість обумовлена структурою кори головного мозку, а розумова діяльність людини пов'язана з певними нервовими мережами, що є спеціалізованими системами в мозку, структури яких поки ще не встановлено. Факт того, що мозок людини не повністю симетричний у своїх функціях, вірогідно був установлений давно.

Дослідження мозку дозволило зробити два важливих висновки:

- різні ділянки мозку виконують різну роботу;
- мозок переробляє інформацію способами, зовсім відмінними від тих, які можна було б уявити (наприклад, упізнання букв і цифр, які, здавалося б, мають відбуватися в тому самому місці, очевидно, здійснюються в різних місцях. Справедливо й протилежне: деякі процеси, що уявляються роздільними, порушуються при ушкодженні однієї й тієї ж ділянки).

На карті кори мозку людини (рис. 1.2) показано ділянки, функціональна спеціалізація яких установлена. Більша частина кори відведена під порівняно елементарні функції: керування рухом

і первинний аналіз подразників. Ці ділянки, що містять моторну й соматосенсорну зони, а також первинні зорові, слухові й нюхові ділянки, є у всіх видів, що мають добре розвинену кору, і залучаються в роботу при багатьох родах діяльності.

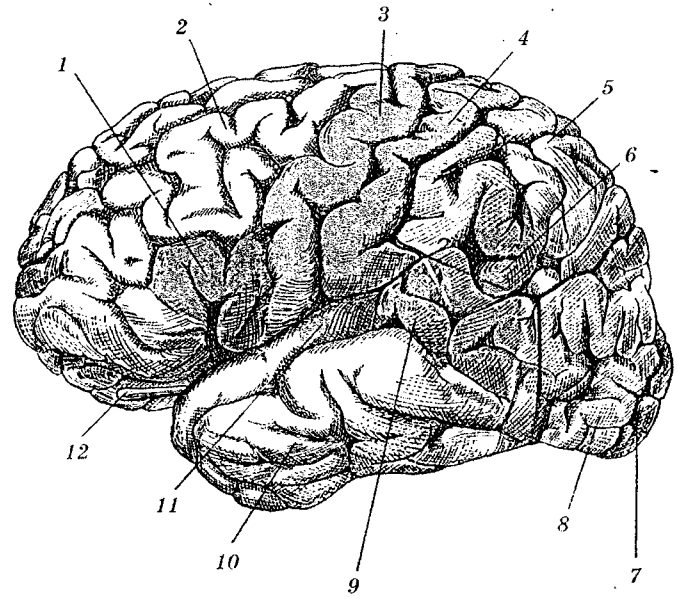


Рис. 1.2. Карта кори головного мозку людини:
1 — зона Брока; *2* — лобова частка; *3* — моторна кора; *4* — соматосенсорна кора; *5* — тім'яна частка; *6* — кутова звивина; *7* — первинна зорова зона; *8* — потилична частка; *9* — зона Верніке; *10* — скронева частка; *11* — первинна слухова зона; *12* — нюхова цибулина

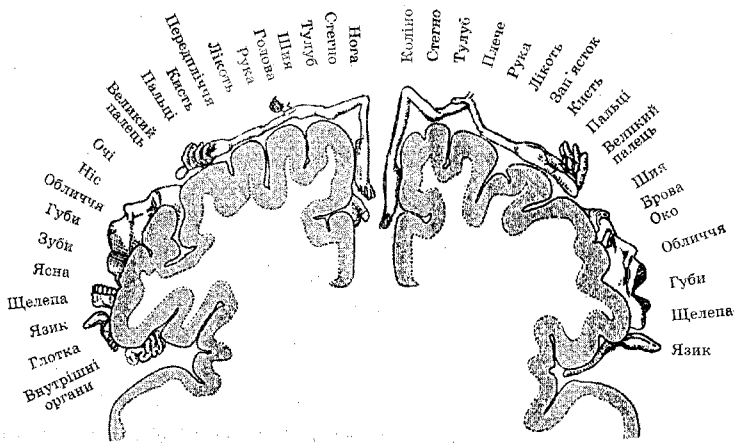
Деякі інші ділянки, зображені на рис. 1.2 темними кольорами, більш вузько спеціалізовані. Зони Брока й Верніке беруть участь у формуванні й сприйнятті мови. Передбачається, що співвіднесення зорового й слухового подання інформації здійснює кутова звивина. Такі функціональні спеціалізації виявлено тільки на лівій половині мозку; відповідні ділянки правої півкулі не мають аналогічного зв'язку з лінгвістичними здібностями. Права півкуля, що тут не показана, визначає свої власні специфічні здібності, зокрема дотичні деяких аспектів сприйняття музики й складних зорових образів. Однак анатомічні зони, що асоціюються із цими

здібностями, визначені не так добре, як мовні зони. Навіть у лівій півкулі співвідношення функцій з ділянками кори лише приблизне; деякі зони можуть мати інші функції, крім зазначених, а у виконанні окремих функцій може брати участь кілька зон.

Останнім часом ці функціональні асиметрії були співвіднесені з анатомічними і був покладений початок дослідженню їхнього значення в інших біологічних видів, крім людини.

У людини, як і в інших ссавців, великі ділянки кори відведені під відносно елементарні сенсорні й моторні функції. Дуга, що йде, грубо кажучи, від вуха до вуха через покрівлю мозку, відповідає первинній моторній корі, що здійснює довільне керування м'язами. Паралельно цій дузі й прямо за нею розташована соматосенсорна зона, що отримує сигнали від шкіри, кісток, суглобів і м'язів. Майже кожна ділянка тіла представлена відповідною ділянкою як у первинній моторній, так й у соматосенсорной корі. У задній частині мозку й, зокрема, на внутрішній поверхні потиличних часток, розташовується первинна зорова кора. Первинні слухові зони знаходяться у скроневих частках, нюх зосереджений у певній ділянці на нижній частині лобових часток.

Спеціалізація соматосенсорної і моторної зон кори мозку виражається в тому, що кожну ділянку цих зон можна з'єднати з певною частиною тіла. Інакше кажучи, більша частина тіла мо-



Соматосенсорна кора Моторна кора

Рис. 1.3. Спеціалізація зон кори мозку

же бути представлена на корі у вигляді схеми; результатом цього будуть два перекручених гомункулуси (рис. 1.3). Перекручування пов'язані з тим, що відведена під дану частину тіла площа кори пропорційна не величині цієї частини, а необхідній точності керування. У людини дуже великі площі моторної й соматосенсорної зон, що відповідають обличчю й рукам. Тут показано тільки половини кожної з таких коркових зон: ліва соматосенсорна зона (яка отримує сигнали переважно від правої половини тіла) і права моторна зона (яка здійснює керування рухами лівої половини тіла).

Структурними елементами мозку є нейрони: гліальні клітини, що скріплюють нейрони й, імовірно, допомагають живити їх і вилучати непотрібні продукти обміну речовин; шванківські клітини, що утворюють захисну (мієлінову) оболонку аксонів; кровоносні судини та клітини, що їх складають; череп, що вміщує інші структури й забезпечує їхній захист.

1.2. Нейрон

Нейрони, або нервові клітини, є основними «будівельними блоками» мозку.

Вважають, що мозок людини складається з 10^{11} нейронів: це приблизно стільки ж, скільки зірок у нашій Галактиці.

Нервові клітини аж ніяк не однакові, вони поділяються на безліч різних типів. Хоча є й проміжні форми, у цілому цей розподіл на типи є досить чітким. Ніхто не знає, скільки типів нейронів існує в головному мозку, — їх, безсумнівно, більше сотні, а можливо, більше тисячі. Не існує двох зовсім однакових нейронів. Дві клітини того самого класу приблизно так само подібні між собою, як два дуби або два клени, а розходження двох класів можна порівняти з відмінністю кленів від дубів або навіть від кульбаб.

Незважаючи на це, їхні форми звичайно вміщуються в невелику кількість широких категорій, і більшості нейронів властиві певні структурні особливості, що дозволяють виділити: клітинне тіло, дендрити й аксон (рис. 1.4). Тіло містить ядро й біохімічний апарат синтезу ферментів та інших молекул, необхідних для життєдіяльності клітини. Зазвичай тіло нейрона має приблизно сферичну або пірамідальну форму. Від тіла клітини відходить кілька дендритів й один аксон. Тіло клітини та дендрити вкриті синапсами — дрібними коподібними структурами, через які нейрони одержують сигнали від інших нейронів. Мітохондрії живлять клітину енергією.

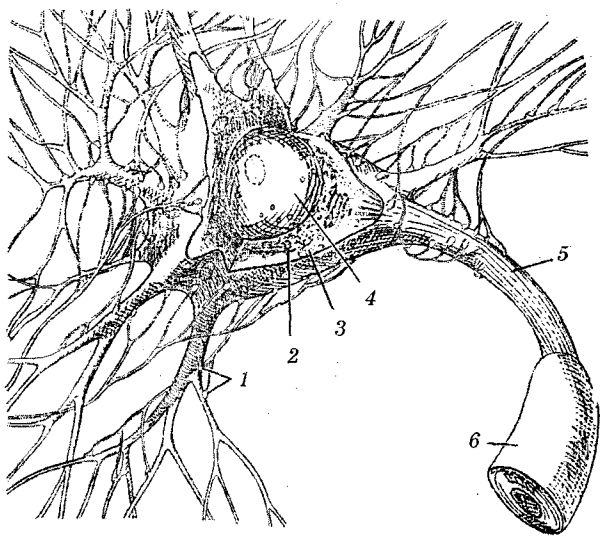


Рис. 1.4. Нейрон:
1 — синапси; *2* — шорсткуватий ендоплазматичний ретикулум; *3* — мітохондрія; *4* — ядро; *5* — аксон; *6* — мієлінова оболонка

Дендрити є тонкими трубчастими виростами, які багаторазово діляться й утворюють гіллясте дерево навколо тіла клітини. Вони створюють ту основну фізичну поверхню, на яку надходять сигнали, що йдуть до даного нейрона. Аксон тягнеться далеко від тіла клітини й служить тією лінією зв'язку, по якій сигнали, генеровані в тілі даної клітини, можуть передаватися на більші відстані в інші ділянки мозку й решти нервової системи. Аксон відрізняється від дендритів як за будовою, так і за властивостями своєї зовнішньої мембрани. Більшість аксонів довше й тонше дендритів і має відмінний від них характер галуження: якщо відростки дендритів в основному групуються навколо клітинного тіла, то відростки аксонів розташовуються на кінці волокна, у тому місці, де аксон взаємодіє з іншими нейронами.

Функціонування мозку пов'язане з рухом потоків інформації по складних ланцюгах, що складаються із нейронних мереж. Інформація передається від однієї клітини до іншої в спеціалізованих місцях контакту — синапсах. Типовий нейрон може мати від 1000 до 10 000 синапсів й отримувати інформацію від тисячі інших нейронів. Хоча у своїй більшості синапси утворюються між аксонами однієї клітини й дендритами іншої, існують інші типи

синаптичних контактів: між аксоном й аксоном, між дендритом і дендритом та між аксоном і тілом клітини.

В області синапса аксон звичайно розширюється, утворюючи на кінці пресинаптичну бляшку, що є частиною контакту, яка передає інформацію (рис. 1.4). Кінцева бляшка містить дрібні сферичні утворення, що називаються синаптичними пухирцями, кожний з яких містить кілька тисяч молекул хімічного медіатора. Після прибуття в пресинаптичне закінчення нервового імпульсу деякі з пухирців викидають свій уміст у вузьку щілину, що відокремлює бляшку від мембрани дендрита іншої клітини, призначеного для прийому таких хімічних сигналів. Таким чином, інформація передається від одного нейрона іншому за допомогою деякого посередника, або медіатора. Імпульсація нейрона відбиває активацію нейронами сотень синапсів. Деякі синапси є збуджувальними, тобто вони сприяють генерації імпульсів, тоді як інші — гальмові — здатні анулювати дію сигналів, які за їх відсутності могли б викликати розряд нейрона.

Для збуджувальних синапсів характерні сферичні пухирці й суцільне стовщення постсинаптичної мембрани, а для гальмових — сплощені пухирці й несуцільне стовщення мембрани. Синапси можна також класифікувати за їхнім розташуванням на поверхні сприймаючого нейрона — на тілі клітини, на стовбурі або «шипику» дендрита, або на аксоні (рис. 1.5).

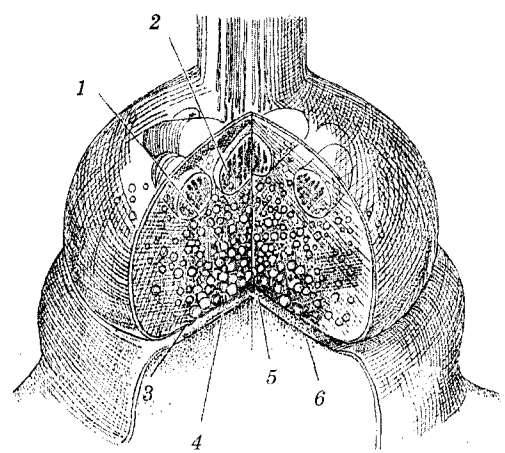


Рис. 1.5. Синапс:
1 — пухирці; *2* — мітохондрія; *3* — пресинаптична мембрана; *4* — постсинаптична мембрана; *5* — синаптична щілина; *6* — іонні канали

1.3. Передавання інформації

Завдання нервової клітини полягає в тому, щоб приймати інформацію від клітин, які її передають, підсумовувати, або інтегрувати, цю інформацію й доставляти інтегровану інформацію іншим клітинам. Інформація звичайно передається у формі короткочасних процесів, що називаються *нервовими імпульсами*. У будь-якій клітині кожен імпульс має бути таким самим, як і будь-який інший, тобто імпульс — це стереотипний процес. У будь-який момент частота імпульсів, що посилається нейроном, визначається сигналами, тільки що отриманими ним від передавальних клітин, і передає інформацію клітинам, стосовно яких цей нейрон є передавальним. Частота імпульсів варіює від одного в кожні кілька секунд або ще нижче до максимуму близько тисячі за секунду.

Сигнали можуть бути двоякими — електричними й хімічними. Сигнал, що генерується нейроном і проводиться по його аксону, є електричним імпульсом, але від клітини до клітини він передається молекулами передавачів, медіаторів.

Нейрони здатні виконувати свою функцію тільки завдяки тому, що їхня зовнішня мембрана має особливі властивості. Мембрана аксона по всій його довжині спеціалізована для проведення електричного імпульсу. Мембрана аксонних закінчень здатна виділяти медіатор, а мембрана дендритів реагує на медіатор. Крім того, мембрана забезпечує впізнавання інших клітин у процесі ембріонального розвитку так, що кожна клітина відшукує призначене їй місце в мережі, що складається з 10^{11} клітин.

Що відбувається, коли інформація передається від однієї клітини до іншої через синапс? У першій — пресинаптичній клітині біля основи аксона виникає електричний сигнал, або імпульс. Імпульс переміщується по аксону до його закінчень. З кожного закінчення в результаті цього імпульсу у вузький (0,02 мкм) заповнений рідиною проміжок, що відокремлює одну клітину від іншої, — *синаптичну щілину* — вивільняється хімічна речовина, що дифундує до другої — *постсинаптичної* — клітини. Вона впливає на мембрану цієї другої клітини таким чином, що ймовірність виникнення в ній імпульсів або зменшується, або зростає. Після цього короткого опису повернемося назад і розглянемо весь процес докладно.

Нервова клітина обмивається сольовим розчином і містить його всередині. До солей відноситься не тільки хлористий натрій, але також хлористий калій, хлористий кальцій і ряд інших, менш

звичайних солей. Оскільки більшість молекул солі дисоційовано, рідини як усередині, так і зовні клітини містять іони хлору, калію, натрію й кальцію (Cl^-, K^+, Na^+ й Ca^{2+}).

У стані спокою електричні потенціали всередині й зовні клітин розрізняються приблизно на одну десяту частку вольта, причому плюс знаходиться зовні (рис. 1.6). Точне значення ближче до величини 0,07 вольта, або 70 мілівольт. Передані нервами сигнали є швидкими змінами потенціалу, що переміщуються по волокну від тіла клітини до закінчень аксона.

Мембрана нервової клітини, що вкриває весь нейрон, — структура надзвичайно складна. Вона не суцільна, а містить мільйони «пор», через які речовини можуть переходити з одного боку на інший. Деякі з них — це дійсно пори різного розміру; вони є білками у формі трубок, що наскрізь пронизують жирову речовину мембрани. В інших випадках це не просто пори, а мініатюрні білкові механізми, що називаються насосами; вони здатні уловлювати іони одного типу й викидати їх із клітини, одночасно захоплюючи інші іони всередину із зовнішнього простору. Таке перекачування вимагає витрати енергії, яку клітина в остаточному підсумку отримує у процесі окислювання глюкози. Існують також пори, що називаються каналами, — це «клапани», які можуть відкриватися й

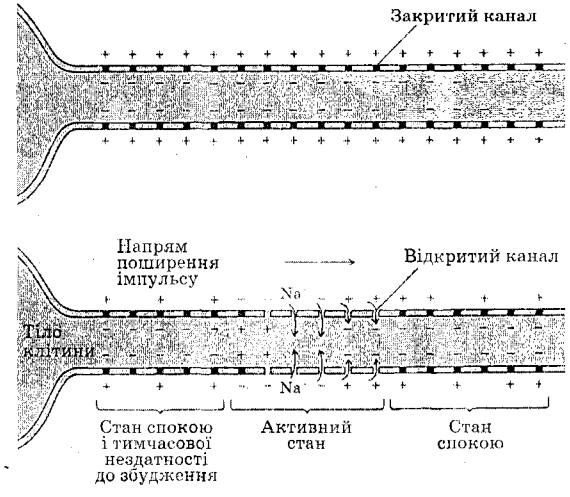


Рис. 1.6. Процес поширення імпульсу

закриватися. Який вплив призводить до їхнього відкриття або закриття, залежить від типу пор. На деякі з них впливає мембранний потенціал, інші відкриваються або закриваються за наявності певних речовин у внутрішній і зовнішній рідині.

Різниця потенціалів на мембрані в будь-який момент визначається концентрацією іонів усередині й зовні, а також тим, відкриті чи закриті різні пори.

Відкриття калієвих пор призводить до виникнення на мембрані різниці потенціалів з позитивним полюсом зовні. Якщо відкрити натрієві пори — зовні виникає негативний заряд.

Коли нерв перебуває в спокої, більшість калієвих каналів відкрито, а більшість натрієвих закрито; тому зовні буде позитивний заряд. Під час імпульсу на короткому відрізку нервового волокна раптово відкривається велика кількість натрієвих каналів, що призводить до короткочасної переваги потоку іонів натрію, і ця ділянка швидко стає електронегативною зовні стосовно внутрішнього простору. Потім натрієві пори знову закриваються, у той час як калієві залишаються відкритими, причому навіть у більшій кількості, ніж у стані спокою. Обидва процеси — закриття натрієвих пор і додаткове відкриття калієвих пор — призводять до швидкого відновлення потенціалу спокою з позитивним полюсом зовні. Вся послідовність подій займає приблизно тисячну частку секунди.

Зменшення потенціалу на мембрані з подальшою зміною його знака (реверсією) не відбувається відразу по всій довжині волокна, оскільки перенос заряду вимагає часу. Активна ділянка виникає в одному місці й переміщується по волокну зі швидкістю від 0,1 до приблизно 10 метрів за секунду. У будь-який момент часу існує одна активна ділянка з реверсованим потенціалом, і ця область реверсії пересувається, віддаляючись від тіла нейрона; попереду неї знаходиться ділянка із ще не відкритими каналами, а позаду — ділянка, де канали знову закрилися й тимчасово нездатні до повторного відкриття. Процес поширення імпульсу представлений на рис. 1.6.

На цьому рисунку вгорі — ділянка аксона в стані спокою. Натрієвий насос перекачав назовні зайві іони натрію, а всередину — відсутні іони калію. Натрієві канали переважно закриті. Оскільки відкрито багато калієвих каналів, клітину покинула достатня кількість калію, щоб мембранний потенціал досяг рівноважного в таких умовах рівня — близько 70 мілівольтів із плюсом зовні. Внизу — зліва направо переміщається нервовий імпульс. На крайньому правому кінці аксон перебуває ще в стані спокою. У середній ділянці відбуваються процеси, пов'язані з імпульсом: натрієві канали відкриті, іони натрію переходять усередину (хоча й не в такій кількості, щоб їхня концентрація після одного імпульсу помітно змінилася); мембранний потенціал 40 мілівольтів із плюсом усередині. На крайньому лівому кінці мембрана повертається у вихідний стан, оскільки відкрилися (а потім закрилися) додаткові калієві канали, а натрієві канали автоматично закрилися. Оскільки натрієві канали не здатні відразу ж повторно відкритися, другий імпульс не може виникнути раніше, ніж через приблизно мілісекунду. Це дозволяє зрозуміти, чому імпульс не може повернути назад до тіла клітини.

Центральна нервова система, що знаходиться між вхідними й вихідними нейронами, є тим апаратом, що здійснює сприйняття, реагування, запам'ятовування й т. д. Чимало відділів центральної нервової системи організовані у вигляді послідовних шарів-рівнів (рис. 1.7). Клітина одного рівня отримує численні збуджувальні й гальмові входи від попереднього рівня й посилає вихідні сигнали багатьом клітинам наступного рівня. Основну масу вхідної інформації нервова система отримує від рецепторів: очей, вух, шкіри й т. д., які перетворюють такі зовнішні впливи, як світло, тепло або звук в електричні нервові сигнали. Виходом можуть бути скорочення м'язів або залозистих клітин.

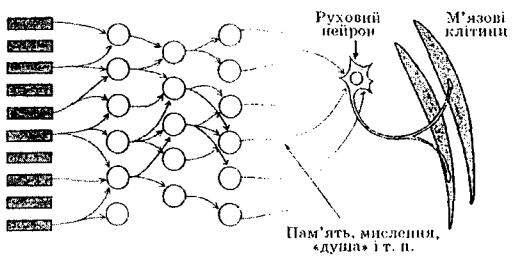


Рис. 1.7. Багаторівнева організація центральної нервової системи

Таким чином, з погляду кібернетики як власне центральна нервова система, так й її відділи являють собою здебільшого «чорний ящик», дослідження якого є надзвичайно складним. Навіть результати найбільш вивченої частини мозку, зорової системи («сірого ящика»), що дозволили простежити шлях інформації від клітин сітківки до шостого-сьомого етапу й установити роль кори півкуль великого мозку, пов'язаної із зором, не дають відповіді на

запитання, як відбувається сприйняття або впізнання предметів. Однак, незважаючи на це, багато виявлених властивостей центральної нервової системи можуть бути промодельовані й реалізовані за допомогою технічних засобів.

Контрольні запитання

1. Які основні висновки зроблено в результаті дослідження ШНМ?
2. Що являє собою карта кори головного мозку?
3. З чого складається нейрон?
4. Яким чином відбувається передача інформації?
5. Що являє собою центральна нервова система з точки зору кібернетики?

2. ОСНОВНІ ПОНЯТТЯ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

2.1. Структура штучного нейрона

Штучні нейрони, що також називаються *нейронними клітинами*, *вузлами*, *модулями*, моделюють структуру й функції біологічних нейронів. Архітектура й особливості штучних нейронних мереж, утворених нейронами, залежать від конкретних завдань, які мають бути вирішені з їхньою допомогою [34, 53–60].

Структуру штучного нейрона зображено на рис. 2.1.

Вхідними сигналами штучного нейрона $x_i (i = \overline{1,N})$ є вихідні сигнали інших нейронів, кожний з яких узятий зі своєю вагою w_i $(i = \overline{1,N})$, аналогічною синаптичній силі.

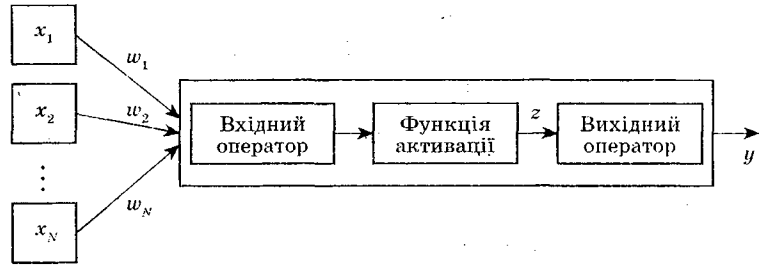


Рис. 2.1. Структура штучного нейрона

Вхідний оператор $f_{вх}$ перетворює зважені входи й подає їх на *оператор активації* f_a. Вихідний сигнал нейрона y являє собою перетворений *вихідним оператором* $f_{вих}$ вихідний сигнал оператора активації. Таким чином, нелінійний оператор перетворення вектора вхідних сигналів x у вихідний сигнал y може бути записаний у такий спосіб:

$$y = f_{вих}(f_a(f_{вх}(\boldsymbol{x},\boldsymbol{w}))). \qquad (2.1)$$

Як ми вже зазначали, вихідний сигнал даного нейрона є вхідним для наступного.

Вхідний оператор

Вхідний оператор (вхідна функція) нейрона задає вигляд використовуваного в нейроні перетворення зважених входів. Відмінність гальмуючих входів від збуджувальних відбивається у знаках відповідних ваг. Звичайно використовуються такі вхідні функції:

— сума зважених входів

$$f(\boldsymbol{x},\boldsymbol{w}) = \sum_{i=1} w_i x_i; \tag{2.2}$$

— максимальне значення зважених входів

$$f(\boldsymbol{x},\boldsymbol{w}) = \max_i (w_i x_i); \tag{2.3}$$

— добуток зважених входів

$$f(\boldsymbol{x},\boldsymbol{w}) = \prod_{i=1}^{N} w_i x_i; \tag{2.4}$$

— мінімальне значення зважених входів

$$f(\boldsymbol{x},\boldsymbol{w}) = \min_i (w_i x_i). \tag{2.5}$$

Функція активації

Функція активації (*activation function*) $f_a(\cdot)$ описує правило переходу нейрона, що перебуває в момент часу k у стані $z(k)$, у новий стан $z(k+1)$ при надходженні вхідних сигналів $\boldsymbol{x}$

$$z(k+1) = f_a(z(k), f_{вх}(\boldsymbol{x}, \boldsymbol{w})). \tag{2.6}$$

Надалі позначатимемо функцію активації без індексу «а».

Найбільш простими активаційними функціями є

— *лінійна*

$$f(z) = Kz, \quad K = \text{const}; \tag{2.7}$$

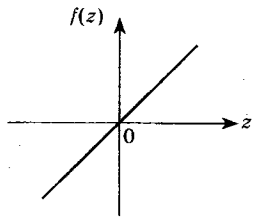


Рис. 2.2. Лінійна функція

— *лінійна біполярна з насиченням*

$$f(z) = \begin{cases} 1 & \text{при} \quad z > \alpha_2; \\ Kz & \text{при} \quad -\alpha_1 \le z \le \alpha_2; \\ -1 & \text{при} \quad z < \alpha_1; \end{cases} \tag{2.8}$$

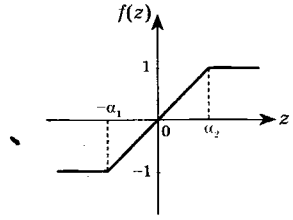


Рис. 2.3. Лінійна біполярна функція з насиченням

— *лінійна уніполярна з насиченням*

$$f(z) = \begin{cases} 1 & \text{при} \quad z \ge \dfrac{1}{2\alpha}; \\ \alpha z + 0{,}5 & \text{при} \quad |z| < \dfrac{1}{2\alpha}; \\ 0 & \text{при} \quad z \le -\dfrac{1}{2\alpha}. \end{cases} \tag{2.9}$$

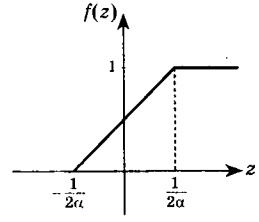


Рис. 2.4. Лінійна уніполярна функція з насиченням

Незважаючи на те, що лінійні функції є найбільш простими, їхнє застосування обмежене, в основному, найпростішими ШНМ, які не мають у своєму складі прихованих шарів, у яких, крім того, існує лінійна залежність між вхідними й вихідними змінними. Такі мережі мають обмежені можливості. Для багатошарової ж

лінійної мережі справедливо наступне. Оскільки після вхідного оператора на оператор активації надходить сукупність зважених вхідних сигналів, записана, наприклад, у матричному вигляді (більш докладно *див.* нижче) $W_1\boldsymbol{x}$, використання лінійної активаційної функції призводить до того, що на виході другого шару з'явиться сигнал $W_2(W_1\boldsymbol{x}) = (W_2W_1)\boldsymbol{x}$. Це означає, що двошарова лінійна мережа еквівалентна одношаровій з ваговою матрицею, що дорівнює добутку вагових матриць першого й другого шарів. Звідси випливає, що будь-яка багатошарова лінійна мережа може бути замінена еквівалентною одношаровою. Хоча, як показано в [34], використання лінійних активаційних функцій не є зайвим у багатошарових ШНМ, для розширення ж можливостей мережі застосовують нелінійні функції активації.

У роботі *У. Маккаллоха* і *У. Піттса* [1] як активаційна використовувалася функція Хевісайда — *уніполярна гранична функція* вигляду

$$f(z)=\begin{cases}1 & \text{при } z\ge\alpha;\\ 0 & \text{при } z<\alpha.\end{cases} \tag{2.10}$$

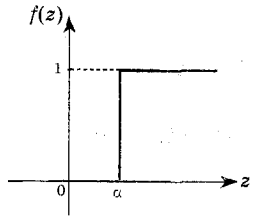


Рис. 2.5. Уніполярна порогова функція

Різновидом даної функції є *біполярна порогова функція*

$$f(z)=\begin{cases}1 & \text{при } z\ge\alpha;\\ -1 & \text{при } z<\alpha.\end{cases} \tag{2.11}$$

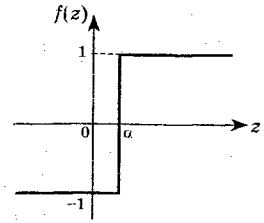


Рис. 2.6. Біполярна порогова функція

Ці функції активації застосовувалися в основному в класичних ШНМ. При побудові нових структур ШНМ найчастіше доводиться працювати як із самою активаційною функцією, так і з її першою похідною. У цих випадках необхідним є використання як активаційної монотонної диференційованої й обмеженої функції. Особливо важливу роль відіграють такі функції під час моделювання нелінійних залежностей між вхідними й вихідними змінними. Це так звані *логістичні*, або *сигмоїдальні* (*S-подібні*), функції.

Функція $f(\cdot)$ називається *сигмоїдальною*, якщо вона є монотонно зростаючою, диференційованою і задовольняє умові

$$\lim_{\lambda\to-\infty} f(\lambda)=k_1,\quad \lim_{\lambda\to\infty} f(\lambda)=k_2,\quad k_1<k_2. \tag{2.12}$$

До таких функцій належать:

— *логістична* (*уніполярна*)

$$f_{\log}(z)=\frac{1}{1+e^{-\alpha z}}; \tag{2.13}$$

$$\frac{d}{dz}f_{\log}(z)=\alpha f_{\log}(z)\bigl(1-f_{\log}(z)\bigr);$$

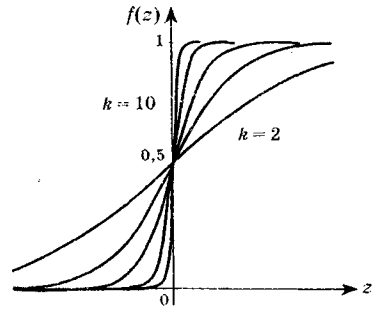


Рис. 2.7. Логістична функція

— *гіперболічного тангенса* (*біполярна*)

$$f_{\text{th}}(z)=\tanh(\alpha z)=\frac{e^{\alpha z}-e^{-\alpha z}}{e^{\alpha z}+e^{-\alpha z}}; \tag{2.14}$$

$$\frac{d}{dz}f_{\text{th}}(z)=1-\tanh^2(\alpha z).$$

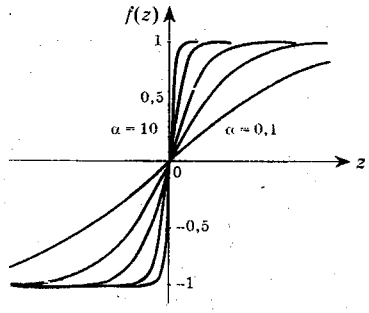


Рис. 2.8. Функція гіперболічного тангенса

Функції (2.13) і (2.14) можуть бути виражені одна через одну. Наприклад,

$$f_{\text{th}}(z)=\tanh(\alpha z)=\frac{e^{\alpha z}(1-e^{-2\alpha z})}{e^{\alpha z}(1+e^{-2\alpha z})}=\frac{2-(1+e^{-2\alpha z})}{1+e^{-2\alpha z}}=2f_{\log}(z)-1.$$

Аналогічно можна показати, що $f_{\log}(z)=\frac{1}{2}(\tanh(\frac{z}{2})-1)$.

Слід також зазначити, що перевага функції $f_{\text{th}}(z)$ перед $f_{\log}(z)$ полягає в її симетричності відносно початку координат (у деяких випадках це істотно полегшує обчислення).

— *синусоїдальна з насиченням (біполярна)*

$$f_{\sin}(z)=\begin{cases}1 & \text{при } z\ge\alpha;\\ \sin z & \text{при } |z|<\alpha;\\ -1 & \text{при } z\le-\alpha,\end{cases} \qquad (2.15)$$

$$\frac{d}{dz}f_{\sin}(z)=\sqrt{1-f_{\sin}^2(z)}\,;$$

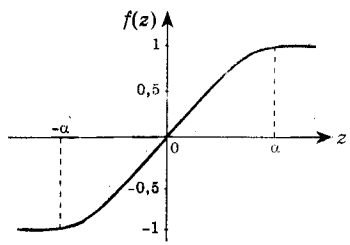


Рис. 2.9. Функція синусоїдальна з насиченням

— *косинусоїдальна з насиченням (уніполярна)*

$$f_{\cos}(z)=\begin{cases}1 & \text{при } z\ge\frac{\pi}{2};\\ \frac{1}{2}(1+\cos(z-\frac{\pi}{2})) & \text{при } |z|<\frac{\pi}{2};\\ 0 & \text{при } z\le-\frac{\pi}{2}.\end{cases} \qquad (2.16)$$

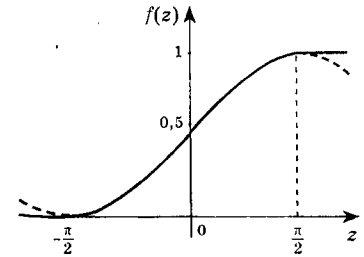


Рис. 2.10. Функція косинусоїдальна з насиченням

— *модульована сигмоїда*

$$f(x,y)=\frac{1}{1+e^{-x-y-\theta}}-\frac{1}{1+e^{x-y-\theta}}. \qquad (2.17)$$

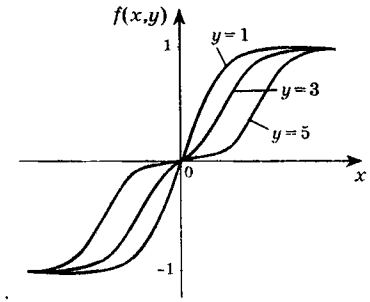


Рис. 2.11. Функція «модульована сигмоїда»

Сигмоїдальну функцію за аналогією з електронними системами можна вважати нелінійною підсилювальною характеристикою штучного нейрона. Центральна область такої функції, що має великий коефіцієнт підсилення, вирішує проблему обробки слабких сигналів, а області зі спадним посиленням на позитивному й негативному кінцях служать для обробки значних збуджень. Таким

чином, нейрон функціонує з більшим підсиленням у широкому діапазоні рівнів вхідного сигналу.

На завершення зазначимо, що вибір конкретного виду активаційної функції специфічний для кожного виду ШНМ і залежить від розв'язуваного завдання.

Вихідний оператор

Вихідний оператор служить для представлення стану нейрона в бажаній області значень. Зазвичай у більшості робіт цей оператор не виділяють, а під вихідним сигналом нейрона розуміють сигнал після оператора активації. Однак під час аналізу й синтезу ШНМ, що містять різні активаційні функції, які мають різні області значень й області визначення, виникає необхідність використання такого оператора $f_{вих}$.

2.2. Моделі штучних нейронів

Моделі штучних нейронів залежать від конкретних застосувань. Тому синтез моделі в кожному окремому випадку є нетривіальним завданням.

2.2.1. Формальна модель нейрона Маккаллоха — Піттса

Формальний штучний нейрон (його називають також *нейроном Маккаллоха — Піттса*) може бути поданий як багатовхідний нелінійний перетворювач із ваговими коефіцієнтами w_{ji}, які також називаються синаптичними вагами або підсилювачами (рис. 2.12). *Клітина тіла* (*сома*) описується нелінійною обмежувальною або пороговою функцією $f(u_j)$. Найпростіша модель штучного нейрона підсумує N ваг входів і здійснює нелінійне перетворення (*див.* рис. 2.1)

$$y_j = f(\sum_{i=1}^{N} w_{ji}x_i + \theta_j), \qquad (2.18)$$

де y_j — вихідний сигнал j-го нейрона; f — обмежувальна або порогова функція (активаційна); N — кількість входів; w_{ji} — синаптичні ваги; x_i — вхідні сигнали ($i = \overline{1,N}$); θ_j, ($\theta_j \in R$) — пороговий сигнал, що також називається *зсувом*.

Позначаючи $\theta_j = w_{j0}x_0$ (зазвичай $x_0 = 1$), вираз (2.18) можна переписати у вигляді

$$y_j = f(\sum_{i=0}^{N} w_{ji}x_i) = f(\boldsymbol{w}_j^T \boldsymbol{x}), \qquad (2.19)$$

де $\boldsymbol{x} = (1, x_1, ..., x_N)^T$; $\boldsymbol{w}_j = (w_{j0}, w_{j1}, ..., x_{jN})^T$ — відповідні вектори входів і ваг розмірності $(N + 1) \times 1$.

Як випливає з (2.19), будь-який формальний нейрон характеризується своєю активаційною функцією й *порогом* θ_j. Перша модель нейрона, запропонована У. Маккаллохом і У. Піттсом, використовувала тільки бінарні (жорстко обмежені) функції (2.10). У цій моделі сума всіх зважених входів порівнювалася із пороговим значенням θ_j, і якщо вона перевищувала це порогове значення, вихід нейрона встановлювався у «верхнє значення» або логічну 1, в іншому випадку — в «нижнє» або логічний 0.

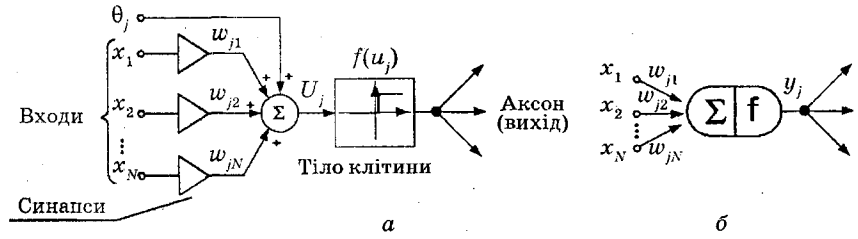


Рис. 2.12. Модель штучного нейрона Маккаллоха — Піттса та її позначення

У сучасних застосуваннях порогова функція заміняється більш загальною нелінійною функцією й, отже, вихід нейрона y_i може приймати як дискретне значення на множині (наприклад, {–1, 1}), так і неперервне (наприклад, між –1 й 1 або в загальному випадку між y_{min} й y_{max}). Рівень активації або стан нейрона залежить від значення його вихідного сигналу y_i (наприклад, y_i = 1, якщо нейрон активний, і y_i = 0 — нейрон перебуває в незбудженому стані) і звичайно визначається монотонно зростаючою сигмоїдальною функцією.

Модель сигмоїдальної функції може бути побудована на основі традиційних електронних компонент. В електричному ланцюзі напруга імітує *тіло клітини* (*сому*), провідності замінюють вхідну структуру (*дендрити*) і вихідну (*аксон*) та різні резистори *моделюють синаптичні ваги* (*синапси*). Вихідна напруга y_i імітує вихідну реакцію біологічного нейрона. Сигмоїдальна активаційна функція

представлена вольт-амперною характеристикою підсилювача з насиченням. Сигнали x_i надходять у формі вхідної напруги у провідники-дендрити у співвідношенні сум вихідних напруг і відповідних електропровідностей (рис. 2.13).

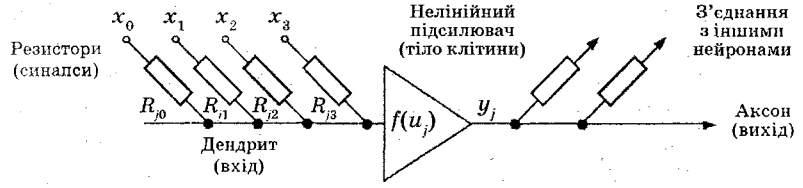


Рис. 2.13. Електронна аналогова модель нейронної клітини

Струм у вхідних ланцюгах схеми (дендритах) прямо пропорційний вхідним напругам x_i і провідностям w_{ji}. Напругу на виході може бути отримано відповідно до закону Кірхгофа

$$y_j = f(u_j) = f(\sum_{i=0}^{N} G_{ji} x_i / \sum_{i=0}^{N} G_{ji}), \qquad (2.20)$$

де y_i — вихідна напруга j-го нейрона; $f(\cdot)$ — сигмоїдальна функція підсилювача; u_j — вхідна напруга j-го підсилювача; $G_{ji} = R_{ji}^{-1}$ — провідність резисторів на вході схем; $x_j (j = \overline{0,N})$ — вхідні напруги.

Вираз (2.20) можна переписати в компактній формі у вигляді (2.19) з $w_{ji} = G_{ji} / \sum_{i=0}^{n} G_{ji}$.

2.2.2. Модель нейрона Адаліни

Однією з найважливіших властивостей біологічних і штучних нейронів є їхня здатність до навчання за допомогою зміни синаптичних ваг, здійснювана за тим чи іншим алгоритмом. Навчання штучних нейронних мереж полягає у виборі їхньої правильної реакції на пропоновані навчальні сигнали настроювання синаптичних ваг.

Практично одночасно з моделлю Маккаллоха — Піттса з'явилася модель штучного нейрона, що навчається, у вигляді *адаптивного лінійного елемента*, розробленого *Б. Уідроу* й названого *Адаліною* (***AD**Aptive **LI**near **NE**uron*) [6,7]. Адаліна складається з двох частин: лінійного підсилювача, що настроюється, і дворівневого

квантувача (з жорстко обмеженою або релейною активаційною функцією). Докладно Адаліна буде розглянута нижче.

2.2.3. Модель нейрона Фукушіми

У загальному випадку, як й у випадку моделі Маккаллоха — Піттса, синаптичні ваги нейрона можуть бути позитивними, рівними нулю або негативними залежно від того, як впливають сигнали, що надходять, на його реакцію. Сигнали називаються *збуджувальними*, якщо ваги позитивні ($w_{ji} > 0$), і *гальмуючими* — якщо негативні ($w_{ji} < 0$). Однак у моделі штучного нейрона, запропонованого *К. Фукушімою*, всі синаптичні ваги й всі вхідні й вихідні сигнали є ненегативними, тобто вони можуть бути рівними нулю або приймати будь-яке позитивне значення [61–64]. У цій моделі входи й відповідні синаптичні ваги поділяються на дві групи: збуджувальні й гальмуючі (рис. 2.14).

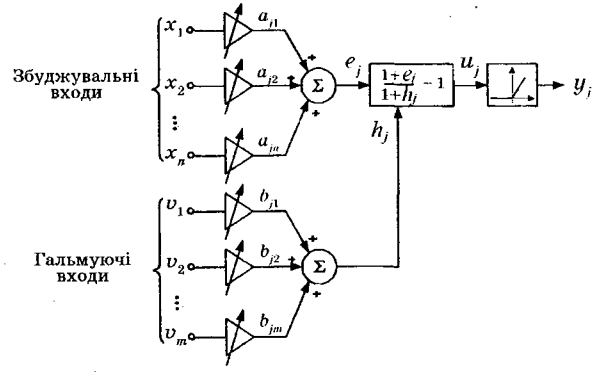


Рис. 2.14. Спрощена модель нейрона Фукушіми

Збуджувальний ефект e_j визначається зваженою сумою всіх збуджувальних входів. Вихід нейрона може бути описаний виразом

$$y_j = f\left[\frac{1+\sum_{i=1}^{n} a_{ji} x_i}{1+\sum_{i=1}^{m} b_{ji} v_i} - 1\right], \qquad (2.21)$$

де $f(u_j) = \begin{cases} u_j, & \text{якщо } u_j \geq 0; \\ 0 & \text{в іншому випадку.} \end{cases}$

У даному виразі a_{ji} відповідає збуджувальним синаптичним вагам, реалізованим у вигляді негативних електропровідностей, і b_{ji} — гальмуючим, також реалізованим у вигляді негативних електропровідностей. Синаптичні ваги звичайно є змінними, і вони змінюються у процесі навчання нейронної мережі.

2.2.4. Модель штучного нейрона Гопфілда

Усі описані вище структури нейронів належать до статичних структур і не здатні моделювати динамічний процес. Альтернативою їм може виступати модель нейрона *Дж. Гопфілда*, що є сьогодні найбільш популярною динамічною моделлю біологічного нейрона [23]. На рис. 2.15 зображено електричну й функціональну структури нейрона. Електрична схема складається з конденсатора C_j, резистора R_{ji} і нелінійного підсилювача із сигмоїдальною функцією. Передбачається, що підсилювач має два асиметричних виходи, при цьому всі резистори, що моделюють синаптичні ваги, мають позитивні значення. Це означає, що позитивна синаптична вага реалізується з'єднанням резистора R_{ji} з $(+v_i)$ і негативною вагою — з'єднанням R_{ji} із сигналом $(-v_i)$. Струм I_j представляє зсув (незалежний зовнішній вхідний сигнал).

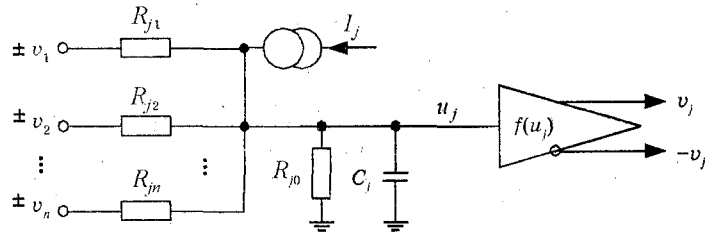


Рис. 2.15. Модель Гопфілда динамічної нейронної клітини

Застосовуючи закон Кірхгофа, отримуємо таке диференціальне рівняння, що описує нейрон:

$$C_j \frac{du_j}{dt} = -\frac{u_j}{R_j} + \sum_{i=1}^{N} \frac{v_i}{R_{ji}} + I_j, \qquad (2.22)$$

де $\frac{1}{R_j} = \frac{1}{R_{j0}} + \sum_{n=1}^{n} \frac{1}{R_{ji}} = G_{j0} + \sum_{i=1}^{N} G_{ji}$. Тут G_{ji} визначає електропровідності $(i = \overline{1,N})$; $v_j = f(u_j)$, $(j = \overline{1,N})$.

Вираз (2.22) може бути переписаний у більш загальній формі

$$T_j \frac{du_j}{dt} = -a_j u_j + \left(\sum_{i=0}^{N} w_{ji} x_i + \theta_j \right), \qquad (2.23)$$

$$y_j = f(u_j), \qquad (2.24)$$

де $T_j = r_j C_j$ — стала інтегрування; u_j — внутрішній сигнал, що називається внутрішнім потенціалом, або потенціалом дії; $a_j = r_j / R_j$ — коефіцієнт загасання; $w_{ji} = \pm r_j / R_{ji}$ — синаптичні ваги (з позитивним знаком, якщо R_{ji} з'єднаний з $(+x)$, з негативним знаком, якщо R_{ji} з'єднаний з $(-x)$); $x_i = v_i$ $(i = \overline{1,N})$ — вхідні сигнали (напруги, потенціали); $\theta_j = r_j I_j$ — сигнал зсуву. Тут r_j — масштабуючий опір.

Схему нейрона, що відповідає виразам (2.22), (2.23), зображено на рис. 2.16. Схема містить суматор налаштовуваних синаптичних ваг w_{ji}, інтегратор і нелінійний елемент із сигмоїдальною активаційною функцією.

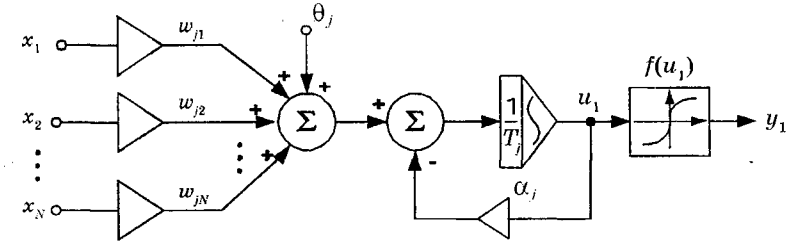


Рис. 2.16. Структура нейрона Гопфілда

Динаміка нейрона в цілому визначається ємністю C_j й опором R_{j0}. Синаптичні ваги визначаються вхідними електропровідностями $G_{ji} = 1/R_{ji}$, з'єднаними з одним з виходів $(+v$ або $-v)$ j-го підсилювача. Видно, що позитивні синаптичні ваги $(w_{ji} > 0)$ вимагають, щоб резистор R_{ji} був з'єднаний з позитивним, а негативні $(w_{ji} < 0)$ — з негативним входом j-го підсилювача.

Дискретну модель нейрона Гопфілда зображено на рис. 2.17.

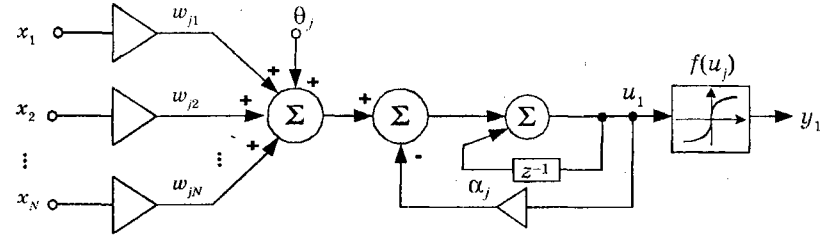


Рис. 2.17. Дискретна модель нейрона Гопфілда

2.2.5. Модель Ґроссберґа

Штучний нейрон, розроблений *С. Ґроссберґом* [20], є узагальненням моделі Гопфілда й може бути описаний таким диференціальним рівнянням:

$$T_j \frac{du_j}{dt} = -\alpha_j u_j + (\gamma_j - \beta_j u_j)\left(\sum_{i=1}^{n} w_{ji} f_i(u_i, \theta_j)\right), \qquad (2.25)$$

де u_i — внутрішня активність j-го нейрона; α, β, γ — константи, що визначають динаміку нейрона.

Настроювання параметрів нейрона здійснюється за допомогою алгоритму навчання вигляду

$$\frac{dw_{ji}}{dt} = \left[-\theta_{ji} w_{ji} + d_{ji} f_i(u_i)\right] h_i(u_i). \qquad (2.26)$$

Спрощена функціональна схема моделі штучного нейрона Ґроссберґа, що відповідає виразу (2.25), показана на рис. 2.18. Нейрон може перебувати у двох станах: збудженому і загальмованому, які описуються рівняннями вигляду:

$$T_j \frac{du_j}{dt} = -\alpha_j u_j + (\gamma_{jE} - \beta_{jE} u_j)\left(\sum_{i=1}^{N_E} w_{jiE} f_{iE}(u-i) + \theta_{jE}\right) -$$

$$- (\gamma_{jI} + \beta_{jI} u_j)\left(\sum_{i=1}^{N_I} w_{jiI} f_{iI}(u-i) + \theta_{jI}\right). \qquad (2.27)$$

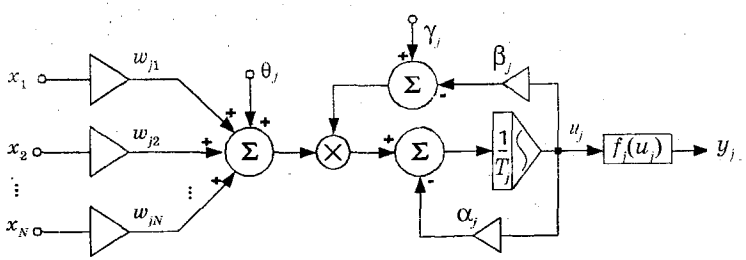


Рис. 2.18. Функціональна модель нейрона Ґроссберґа

На рис. 2.19 зображено дискретну модель нейрона Ґроссберґа.

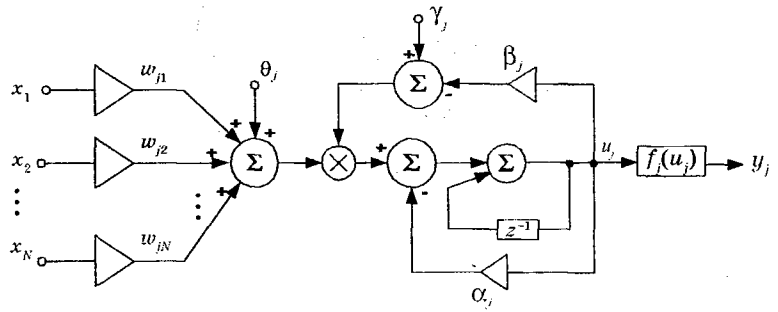


Рис. 2.19. Дискретна модель нейрона Ґроссберґа

Більшість відомих моделей штучних нейронів може бути розглянуто як окремі випадки описаних вище моделей.

2.2.6. Узагальнена модель нейрона

У деяких застосуваннях використовуються узагальнені моделі, які містять у собі більш складні комплексні математичні операції, ніж підсумовування. Найбільш загальне подання може бути записане у вигляді системи нелінійних диференціальних рівнянь

$$T_j \frac{du_j}{dt} = \varphi_j\left(J_j(u_j),\, w_{j1},\, w_{j2} f_2(u_2),\, \ldots,\, w_{jN} f_N(u_N)\right); \qquad (2.28)$$

$$T_{ji} \frac{dw_{ji}}{dt} = g_i(w_{ji},\, u_i,\, u_j), \qquad (2.29)$$

де $\varphi_j(\cdot)$ — нелінійне перетворення на вході нейрона; $J_j(\cdot)$ — функція, що описує внутрішню динаміку j-го нейрона; g_i — передавальна функція, що визначає динаміку зміни синаптичних ваг.

Усі наведені динамічні моделі штучних нейронів подано в неперервному часі. Динамічна поведінка таких нейронів звичайно описується системою диференціальних рівнянь, але під час моделювання роботи ШНМ використовується їхнє дискретне подання.

У загальному випадку дискретна математична модель нейрона має вигляд

$$x_j(k+1) = f\left(\sum_{i=1}^{N} w_{ji} x_i(k) + \theta_j\right). \qquad (2.30)$$

На рис. 2.20 зображено ще одну дискретно-часову модель штучного нейрона, описану виразами

$$u_j(k+1) = u_j(k) + \left(\sum_{i=1}^{N} w_{ji} x_i(k) + \theta_j \right),$$

$$x_j(k+1) = f(u_j(k+1)) \qquad j = \overline{1,N}, \tag{2.31}$$

де $k = 0, 1, 2, \ldots$ — індекс поточного дискретного часу.

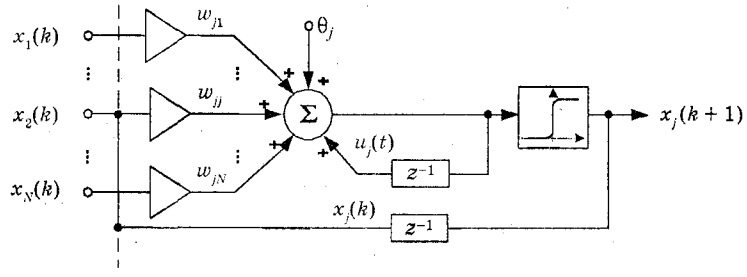


Рис. 2.20. Дискретно-часова модель нейрона

У цій моделі інтегратор замінюється суматором, охопленим елементом чистого запізнювання — дигратором. Дискретно-часова модель нейрона може бути отримана з аналогової моделі шляхом конвертування диференціальних рівнянь у відповідні різницеві рівняння.

Хоча можливості нейронів обмежені, мережі, побудовані з їхньою допомогою, мають практично необмежені можливості.

2.2.7. Σ-П-нейрон

Нейрон даного типу покликаний більш точно відбити властивості біологічного нейрона, а саме здатність моделювати певні синаптичні контакти. З цією метою в цьому нейроні, на відміну від лінійної моделі (2.18), використовується більш складне перетворення вхідних сигналів — їхній добуток або кореляція [37]

$$z(w,x) = \sum_{i=1}^{m} w_i \prod_{j=1}^{N_i} x_j. \tag{2.32}$$

Вихідний сигнал формується так само, як й у розглянутому в підрозділі 2.2.1 нейроні:

$$y = f(w,x) = f\left(\sum_{i=1}^{m} w_i \prod_{j=1}^{N_i} x_j \right), \tag{2.33}$$

де $f(\cdot)$ — функція активації (найчастіше гранична).

Σ-П-нейрон наведено на рис. 2.21.

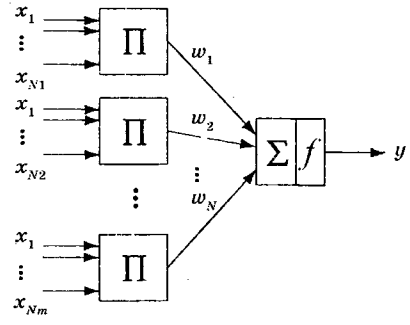


Рис. 2.21. Σ-П-нейрон

2.2.8. Стохастичний нейрон

Розглянуті вище моделі нейронів є детермінованими, оскільки при подачі на їхній вхід деяких сигналів вони видаватимуть однозначно обумовлений використовуваною активаційною функцією вихідний сигнал. У роботі [65] була розглянута модель стохастичного нейрона, у якій активаційна функція є випадковою й залежить від активності нейрона z. При цьому вихідний сигнал нейрона формується за таким правилом:

$$y = \begin{cases} +1 & \text{з імовірністю } P(z \mid y = 1); \\ -1 & \text{з імовірністю } P(z \mid y = -1), \end{cases} \tag{2.34}$$

де $P(z|y = 1) + P(z|y = -1) = 1$.

Модель такого нейрона зображено на рис. 2.22.

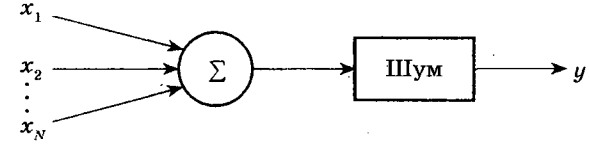


Рис. 2.22. Модель стохастичного нейрона

Так, якщо $P(z|y=1)=(1+\exp(-2\alpha z)^{-1})$, то $P(z|y=-1)=1-P(z|y=1)=$ $=-\exp(-2\alpha z)$, а математичне очікування вихідного сигналу $<y>$ визначається в такий спосіб:

$$<y> = 1P(z|y=1) + (-1)\,P(z|y=-1) = \tanh(\alpha z).$$

Зазначимо, що ця модель є основою *машини Больцмана* й *машини Коші*, а також лежить в основі навчання із підкріплюванням. Ці питання більш докладно будуть висвітлені нижче.

2.3. Топологія ШНМ

З'єднані між собою нейрони утворюють ШНМ. Таким чином, ШНМ — пара (M, V), де M — множина нейронів; V — множина зв'язків. Структура мережі задається у вигляді *графа*, у якому вершини є нейронами, а ребра являють собою зв'язки (з'єднання).

Кожен нейрон мережі має вхідні ланцюги, причому їхня кількість є довільною для кожного нейрона.

У загальному випадку ШНМ складається з декількох шарів, серед яких обов'язково є *вхідний*, що отримує зовнішні сигнали, *вихідний*, що відбиває реакцію нейронів на комбінації вхідних сигналів, і в багатошарових ШНМ — *приховані шари* (рис. 2.23). Така пошарова організація є аналогом шаруватих структур певних відділів мозку.

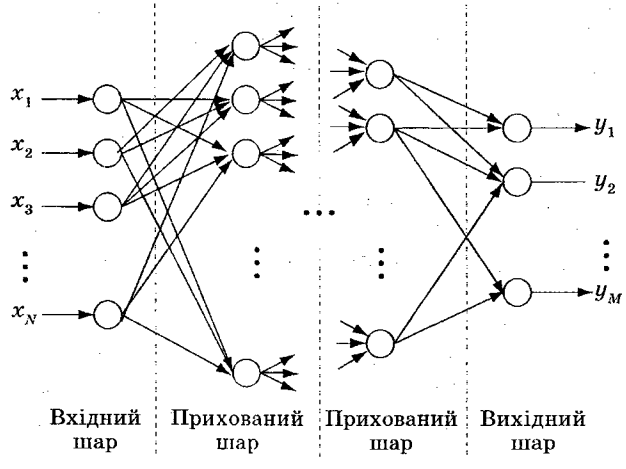


Рис. 2.23. Структура ШНМ

Зв'язки між нейронами задаються у вигляді векторів і матриць. Ваги зручно подавати елементами матриці $W = [w_{ij}]$ розмірності $N \times M$, де N — кількість входів; M — кількість нейронів. Елемент w_{ij} відбиває зв'язок між i-м й j-м нейронами. При цьому, якщо

$w_{ij} = 0$ — зв'язок між i-м й j-м нейронами відсутній;
$w_{ij} < 0$ — гальмуючий сигнал зв'язок;
$w_{ij} > 0$ — прискорювальний сигнал (збуджувальний) зв'язок.

Приклад 2.1. На рис. 2.24 зображено ШНМ і відповідну їй матрицю зв'язків.

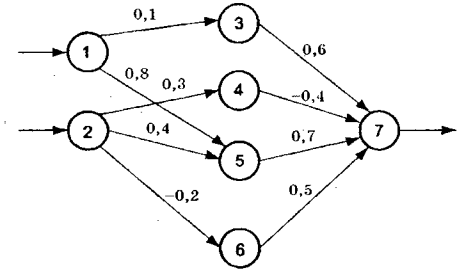


Рис. 2.24. Приклад представлення ШНМ

$$W = \begin{vmatrix} 0 & 0 & 0{,}1 & 0 & 0{,}8 & 0 & 0 \\ 0 & 0 & 0 & 0{,}3 & 0{,}4 & -0{,}2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0{,}6 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0{,}4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0{,}7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0{,}5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

Залежно від того, чи містять ШНМ зворотні зв'язки, чи ні, розрізнюють такі їхні топології:

- ШНМ без зворотних зв'язків (прямого поширення, *Feed forward*)
  — першого порядку;
  — другого порядку (з *«shortcut connections»*);
- ШНМ зі зворотними зв'язками (зворотного поширення, рекурентні, *Feedback*)

— з прямими зворотними зв'язками (*direct feedback*);
— з непрямими зворотними зв'язками (*indirect feedback*);
— з латеральними зв'язками (*lateral feedback*);
— повнозв'язні.

2.3.1. ШНМ прямого поширення

Дана топологія припускає наявність декількох шарів зі зв'язками між нейронами різних шарів. У мережах *першого порядку* існують тільки зв'язки між двома сусідніми шарами, тобто між i-м й $(i+1)$-м шарами. У цьому випадку говорять, що *зв'язки* ШНМ *пошарові*. Приклад такої мережі зображено на рис. 2.23. Якщо в мережі цього типу кожен нейрон шару i пов'язаний з кожним нейроном $(i+1)$-го шару, мережа називається *повнозв'язною прямого поширення*.

У мережах *другого порядку* поряд зі зв'язками між нейронами сусідніх i-го й $(i+1)$-го шарів присутні зв'язки між нейронами шарів i-го й $(i+l)$-го, де $l > 1$. Такий зв'язок називається «*shortcut*». Приклад такої ШНМ наведено на рис. 2.25.

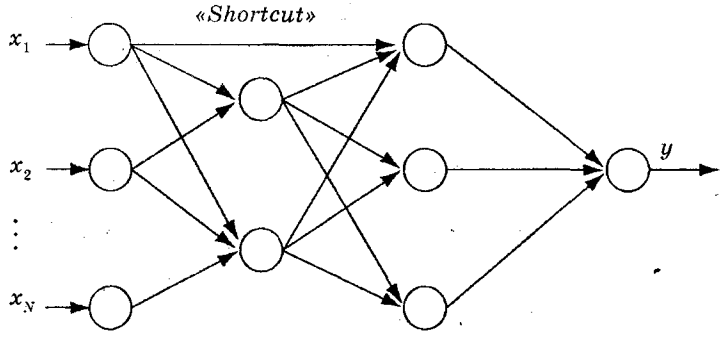


Рис. 2.25. ШНМ прямого поширення другого порядку

Зазначимо, що для мереж прямого поширення матриця зв'язків W є верхньою трикутною матрицею.

2.3.2. ШНМ зворотного поширення

Мережі цього типу припускають наявність зворотних зв'язків як між нейронами різних шарів, так і між нейронами одного шару. Використання мереж зі зворотними зв'язками необхідне у процесі

вивчення складних динамічних об'єктів, наприклад об'єктів, що змінюють свій стан при надходженні нових вхідних сигналів. Такі ШНМ можуть мати властивості, подібні до короткочасної людської пам'яті.

У ШНМ із *прямими зворотними зв'язками* (рис. 2.26) на вхід нейрона деякого i-го шару подається його вихідний сигнал, тобто даний нейрон підсилює або послаблює сигнал, перетворений його активаційною функцією, завдяки чому досягається його граничний активаційний стан.

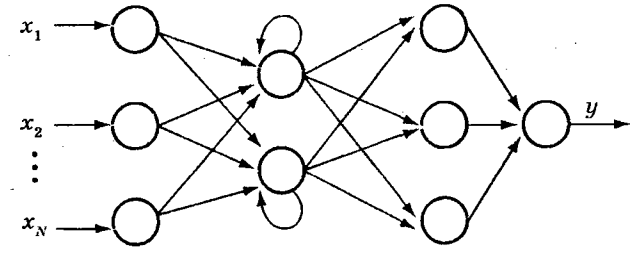


Рис. 2.26. ШНМ із прямими зворотними зв'язками

У ШНМ із *непрямими зворотними зв'язками* існують зв'язки нейрона i-го шару з нейронами $(i-k)$-го шару $k > 0$. При цьому одночасно можуть бути прямі зв'язки цього ж нейрона з нейроном $(i+l)$-го шару $(l > 0)$. Введення таких зворотних зв'язків необхідно, щоб виділити певну особливо важливу для даної ШНМ область вхідних сигналів (рис. 2.27).

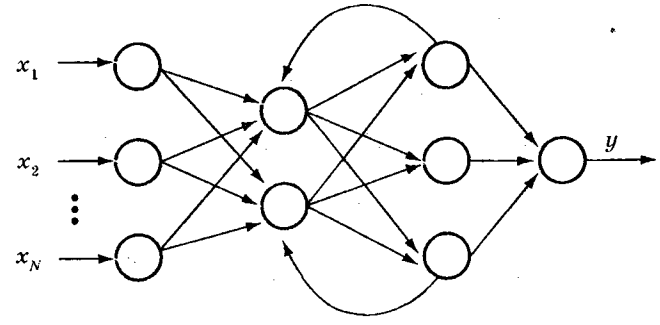


Рис. 2.27. ШНМ із непрямими зворотними зв'язками

ШНМ із *латеральними зв'язками* має зв'язки між нейронами одного шару (рис. 2.28). Такий тип зворотних зв'язків використовується у тому випадку, якщо тільки один нейрон з даної групи нейронів має бути активним. У цьому випадку на вхід кожного нейрона надходять *гальмуючий* (*ослабляючий, інгібіторний*) сигнал від інших нейронів і *звичайно збуджувальний* (*посилюючий, екссгібіторний*) сигнал власного зворотного зв'язку. Нейрон із найбільшою активністю (переможець) придушує інші нейрони. Тому цю топологію називають також топологією мережі *«переможець отримує все»* (*WTA — Net*).

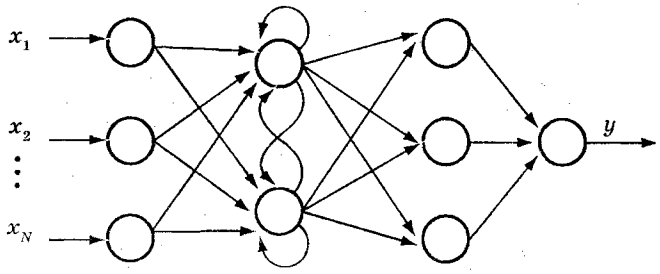


Рис. 2.28. ШНМ із латеральними зв'язками

2.3.3 Повнозв'язні ШНМ

Повнозв'язні ШНМ характеризуються наявністю зв'язків між усіма нейронами мережі (рис. 2.29). Цей вид топології відомий також як мережа Гопфілда.

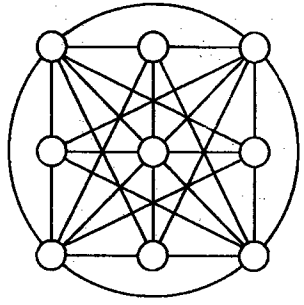


Рис. 2.29. Повнозв'язна ШНМ

Особливістю даної топології є те, що матриця зв'язків W має бути симетричною з нульовими діагональними елементами.

2.4. Навчання ШНМ

Характерною властивістю ШНМ є її здатність до навчання, що полягає у виробленні правильної реакції на подані їй різні вхідні сигнали. Існують такі можливості навчання ШНМ:

- зміна конфігурації мережі шляхом утворення нових або виключення деяких існуючих зв'язків між нейронами;
- зміна елементів матриці зв'язку (ваг);
- зміна характеристик нейронів (виду й параметрів активаційної функції й т. д.).

Найбільшого поширення сьогодні отримав підхід, при якому структура мережі задається апріорно, а мережа навчається шляхом настроювання матриці зв'язків (вагових коефіцієнтів) W. Від того, наскільки вдало побудована ця матриця, залежить ефективність даної мережі. У цьому випадку навчання полягає у зміні за певною процедурою елементів матриці W при послідовному поданні мережі деяких векторів, що навчають.

У зв'язку з цим штучний нейрон може бути представлений у такий спосіб (рис. 2.30).

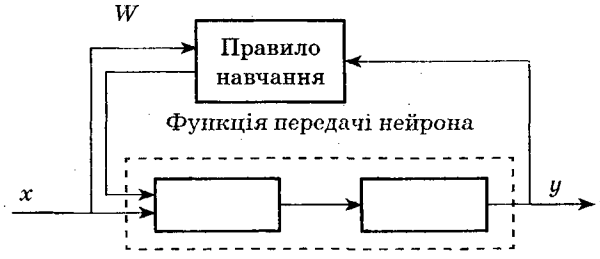


Рис. 2.30. Модель штучного нейрона

У процесі навчання ваги стають такими, що під час надходження вхідних сигналів мережа виробляє відповідні необхідні вихідні сигнали. Розрізнюють *навчання з учителем* і *без учителя*. Перший тип навчання припускає, що є «учитель», що задає пари, які навчають — для кожного вхідного вектора, що навчає, необхідний

вихід мережі. Для кожного вхідного вектора, що навчає, обчислюється вихід мережі, порівнюється з відповідно необхідним, визначається помилка виходу, на основі якої й коректуються ваги. Пари, що навчають, подаються мережі послідовно й ваги уточнюються доти, поки помилка за такими парами не досягне необхідного рівня.

Цей вид навчання неправдоподібний з біологічної точки зору. Дійсно, важко уявити зовнішнього «учителя» мозку, що порівнює реальні й необхідні реакції того, кого навчають, і коригує його поведінку (поведінку нейронів) за допомогою негативного зворотного зв'язку. Більш природним є навчання без учителя, коли мережі подаються тільки вектори вхідних сигналів, і мережа сама, використовуючи деякий алгоритм навчання, підстроювала б ваги так, щоб при поданні їй досить близьких вхідних векторів вихідні сигнали були б однаковими. У цьому випадку в процесі навчання виділяються статистичні властивості безлічі вхідних векторів, що навчають, і відбувається об'єднання близьких (подібних) векторів у класи. Подання мережі вектора з даного класу викликає її певну реакцію, яка до навчання є непередбаченою. Тому в процесі навчання виходи мережі мають трансформуватися в деяку зрозумілу форму. Це не є серйозним обмеженням, оскільки зазвичай нескладно ідентифікувати зв'язок між вхідними векторами й відповідною реакцією мережі.

Існує ще один вид навчання — *з підкріплюванням* (*reinforcement learning*), при якому також передбачається наявність учителя, що не підказує, однак, мережі правильної відповіді. Учитель тільки повідомляє, правильно чи неправильно відпрацювала мережа поданий образ. На основі цього мережа корегує свої параметри, збільшуючи значення ваг зв'язків, що правильно реагують на вхідний сигнал, і зменшуючи значення інших ваг.

Сьогодні існує велика кількість алгоритмів навчання. Деякі з них розглядатимуться пізніше, тут же коротко зупинимося на найбільш відомих.

2.4.1. Правило навчання Гебба (корелятивне, співвідносне навчання)

Більшість сучасних алгоритмів навчання виросло із *правила Гебба* [2]. Наприкінці 40-х років XX ст. років *Д. О. Гебб* теоретично встановив, що асоціативна пам'ять у біологічних системах викликається процесами, що змінюють зв'язки між нервовими клітинами. Відповідно до установленого їм правила, що називається «правилом Гебба», при одночасній активації (порушенні) двох нейронів синаптична сила (вага їхнього зв'язку) зростає. Таким чином, часто використовувані зв'язки в мережі підсилюються, що пояснює феномен звички й навчання повторенням.

У ШНМ зростання синаптичної сили еквівалентне збільшенню ваги зв'язку між нейронами i та j на величину

$$\Delta w_{ij} = \gamma x_i y_j, \tag{2.35}$$

де x_i — вихід i-го й вхід j-го нейронів; y_j — вихід j-го нейрона; γ — коефіцієнт, що впливає на швидкість навчання.

У векторному вигляді це правило може бути записане в такий спосіб:

для неперервного часу $\dfrac{\partial \boldsymbol{w}}{\partial t} = \gamma \boldsymbol{x} y$, (2.36)

для дискретного часу $\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \gamma \boldsymbol{x}(k) y(k)$. (2.37)

Правило Гебба використовується у зв'язках асоціативної пам'яті, а також у деяких інших, заснованих на навчанні без учителя (без підкріплювання). У мережах *асоціативної пам'яті* приймають $y = x$. У *гетероасоціативних мережах* $\boldsymbol{x}$ й $\boldsymbol{y}$ в загальному випадку різняться.

Існують різні варіанти реалізації правила Гебба, наприклад засновані на мінімізації енергетичної або ентропійної функції.

Так, якщо як енергетичну вибрати функцію вигляду

$$I(\boldsymbol{w}) = -\Phi(\boldsymbol{w}^T \boldsymbol{x}) + 0{,}5\alpha \|\boldsymbol{w}\|^2, \tag{2.38}$$

де $\|\boldsymbol{w}\|^2 = \sum_{i=1}^{N} w_i^2$ — евклідова норма; $\alpha \geq 0$ — деякий коефіцієнт; $\Phi(\cdot)$ — диференційована функція така, що

$$y = \frac{d\Phi(\boldsymbol{w}^T \boldsymbol{x})}{d(\boldsymbol{w}^T \boldsymbol{x})} = f(\boldsymbol{w}^T \boldsymbol{x}),$$

то правило Гебба у векторній формі записується в такий спосіб:

$$\frac{d\boldsymbol{w}}{dt} = \gamma(y\boldsymbol{x} - \alpha \boldsymbol{w}). \tag{2.39}$$

Тут $\gamma > 0$ — деякий параметр, що впливає на швидкість навчання.

Для випадку дискретного часу правило (2.39) приймає вигляд

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \gamma\left[y(k)\boldsymbol{x}(k) - \alpha \boldsymbol{w}(k)\right]. \tag{2.40}$$

Якщо замість фактичного значення реакції нейрона y в (2.39), (2.40) використовується необхідне значення y^*, то відповідні алгоритми

$$\frac{dw}{dt} = \gamma(y^* x - \alpha w) \tag{2.41}$$

і

$$w(k+1) = w(k) + \gamma\left[y^*(k)x(k) - \alpha w(k)\right] \tag{2.42}$$

називають *корелятивними правилами* навчання.

Використання як енергетичного функціоналу, що мінімізується, квадратичного вигляду

$$I(w) = 0{,}5\|e\|^2,$$

де $e = x - \hat{x}$ — вектор помилок; $\hat{x}$ — оцінка вхідного вектора, призводить до *правила навчання Ойя* [66–67]. При цьому передбачається, що, по-перше, вектор ваг нормалізований й, по-друге, нейрон має лінійну активаційну функцію, тобто $y = w^T x$. Оцінка вектора вхідного сигналу є зваженим вектором ваг

$$\hat{x} = wy. \tag{2.43}$$

Саме ж правило навчання Ойя записується у такий спосіб:
— для неперервного часу

$$\frac{dw}{dt} = \gamma(xy - wy^2), \tag{2.44}$$

— для дискретного часу

$$w(k+1) = w(k) + \gamma y(k)\left[x(k) - w(k)y(k)\right]. \tag{2.45}$$

Під час використання цього правила виникають деякі проблеми, про які йтиметься згодом. Слід лише зазначити, що це правило досить активно використовувалося в мережах, однак за останні роки виникло багато більш ефективних алгоритмів, що забезпечують значно вищу швидкість навчання.

2.4.2. Дельта-правило

Це важливе правило навчання було запропоновано *Б. Уідроу* й *М. Е. Гоффом* [6] і найбільше відповідає одношаровим ШНМ прямого поширення. Ідея його полягає в тому, що якщо під час навчання мережі можна встановити розбіжність між її бажаною й наявною реакціями, ця розбіжність може бути усунута або змен-

шена шляхом зміни певним чином вагових коефіцієнтів зв'язку. Для цього й використовується *дельта-правило*, відповідно до якого зміна ваги зв'язку між i-м й j-м нейронами визначається у такий спосіб:

$$\Delta w_{ij} = \gamma x_i (y_j^* - y_j), \tag{2.46}$$

де x_i — вихід попереднього i-го нейрона; y_j^*, y_j — бажана й реальна реакції j-го нейрона відповідно; γ — коефіцієнт, що впливає на швидкість навчання.

З (2.46) видно, що якщо різниця $(y_j^* - y_j)$ мала, тобто реакція j-го нейрона незначною мірою відрізняється від бажаної, зміна ваги зв'язку між цими нейронами також буде незначною.

2.4.3. Розширене дельта-правило

Розширене дельта-правило, на відміну від простого, знаходить застосування у багатошарових ШНМ прямого поширення другого порядку. Більш докладно це розглядатиметься нижче. Тут же зазначимо, що відповідно до цього правила зміна ваг здійснюється в такий спосіб:

$$\Delta w_{ij} = \gamma x_i \delta_j, \tag{2.47}$$

де $\delta_j = \begin{cases} f_j'(x,w)(y_j^* - y_j), & \text{якщо } j\text{-нейрон вихідного шару;} \\ f_j'(x,w)\sum_m \delta_m w_{im} & \text{в іншому випадку;} \end{cases}$

$f_j'(x,w)$ — похідна активаційної функції, використовувана в j-му нейроні; індекс j використовується для позначення всіх нейронів наступних шарів, пов'язаних з нейроном j.

2.4.4. Конкурентне навчання

У цьому виді навчання всі нейрони одного шару є конкуруючими, а перевага віддається тому нейрону, що найбільш сильно реагує на вхідний (дратівний) сигнал, тобто настроюються ваги тільки одного нейрона — *нейрона-переможця* (*winner-takes-all*).

Правило ж настроювання ваг звичайно є деякою модифікацією правила Гебба. Початковим значенням вагових коефіцієнтів привласнюються малі, відмінні від нуля й різні значення. При поданні образу, що навчає, реакція одного з нейронів буде найбільш сильною. Його вагові коефіцієнти підсилюються або змінюються таким чином, щоб найповніше відповідати поданому образу, ваги ж інших

(неактивних) нейронів або не змінюються, або зменшуються. Процес навчання завершується, коли вага активного нейрона дорівнюватиме загальній сумі ваг нейронів одного шару.

Реалізація цього виду навчання багатоваріантна.

2.4.5. Стохастичне навчання

Розглянуті вище методи є *детерміністськими*, у яких на кожному такті за певним алгоритмом відбувається корекція ваг, заснована на використанні значень вхідних і вихідних бажаних і фактичних сигналів.

Стохастичні методи навчання базуються на псевдовипадкових змінах ваг зі збереженням тих змін, які ведуть до поліпшень. Для корекції ваг використовується деяка імовірнісна функція. Більші первинні і випадкові корекції зі збереженням певних змін ваг поступово зменшують, досягаючи при цьому мети навчання. Це нагадує процес відпалювання металу, коли атоми розплавленого металу, які перебувають у хаотичному русі при його поступовому охолодженні, гублячи енергію, досягають нижчого з можливих енергетичних станів (глобального мінімуму). Тому для опису цього виду навчання часто використовують термін *«імітація відпалювання»* (*simulated annealing*).

Таке стохастичне навчання використовується в машинах Больцмана й Коші.

2.4.6. Градієнтні методи навчання

Багато методів навчання засновано на мінімізації деякої цільової (вартісної, енергетичної й т. д.) функції I, що являє собою звичайно деяку опуклу функцію. Якщо використовувані функції активації $f(\cdot)$ диференційовані, зручно застосовувати *градієнтні методи мінімізації*. У цьому випадку корекція ваг зв'язку між i-м й j-м нейронами відбувається за правилом

$$\Delta w_{ij} = -\gamma \nabla_w I(\boldsymbol{w}), \tag{2.48}$$

де γ — коефіцієнт, що впливає на швидкість навчання;

$$\nabla_w I(\boldsymbol{w}) = \frac{\partial I(\boldsymbol{w})}{\partial w_{ij}}.$$

Ці методи найчастіше використовуються при контрольованому навчанні, коли відома необхідна реакція нейронів y^*.

Більшість градієнтних методів засновано на мінімізації квадратичного функціонала

$$I(\boldsymbol{w}) = 0{,}5e^2(k) = 0{,}5(y^*(k) - \boldsymbol{w}^T(k)\boldsymbol{x}(k))^2. \tag{2.49}$$

У цьому випадку

$$\nabla_w I(\boldsymbol{w}) = -e(k)\boldsymbol{x}(k). \tag{2.50}$$

Підстановка (2.50) у (2.48) призводить до *алгоритму методу найменших квадратів* (*МНК*)

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \gamma e(k)\boldsymbol{x}(k). \tag{2.51}$$

Градієнтні методи застосовуються також і для навчання стохастичних нейронів, функціонування яких описується формулою (2.34). У цьому випадку замість критерію (2.49) мінімізується критерій

$$I(\boldsymbol{w}) = 0{,}5(y^*(k) - < y(k) >)^2,$$

де $< y(k) > = (+1)P(z(k) \mid y(k) = 1) + (-1)P(z(k) \mid y(k) = -1)$ — математичне сподівання виходу мережі.

Якщо $P(z(k) \mid y(k)) = (1 + \exp(-2\alpha z(k))^{-1}$, то, як було показано в підрозділі 2.2. 8, $< y(k) > = \tanh(\alpha z(k))$ і (*див.* (2.14))

$$\nabla_w I(\boldsymbol{w}) = \alpha(y^*(k) - < y(k) >)(1 - < y^2(k) >)\boldsymbol{x}(k),$$

що призводить до градієнтного алгоритму

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \gamma\alpha(y^*(k) - < y(k) >)(1 - < y^2(k) >)\boldsymbol{x}(k). \tag{2.52}$$

Як видно, цей алгоритм не відрізняється від градієнтного алгоритму навчання детермінованого нейрона, що має активаційну функцію $f(z) = \tanh(z)$.

Очевидно, що властивості (2.51) істотно залежать від вибору коефіцієнта γ. Існують різні рекомендації з вибору γ. Так, у теорії стохастичної апроксимації, що вивчає особливості роботи алгоритмів такого типу за наявності завад вимірів, цей коефіцієнт вибирається змінним і таким, що задовольняє умовам Дворецького [68–70]

$$\lim_{k\to\infty} \gamma(k) = 0, \quad \sum_{k=1}^{\infty} \gamma(k) = \infty, \quad \sum_{k=1}^{\infty} \gamma^2(k) < \infty, \tag{2.53}$$

зміст яких полягає в тому, що для забезпечення збіжності послідовності (2.51) у деяку точку $\boldsymbol{w}^*$ довжина кроку $\gamma(k)$ має з одного боку спадати досить повільно (для забезпечення власне збіжності),

а з іншого — досить швидко (з метою придушення завад). Таким умовам відповідає, наприклад, гармонійний ряд $\gamma(k) = \gamma_0 k^{-\alpha}$, де γ_0 — деяка константа, $0{,}5 < \alpha \le 1$. У теорії стохастичної апроксимації немає рекомендацій з вибору константи γ_0, крім її позитивності. Теорія оптимальної фільтрації, тісно пов'язана з теорією стохастичної апроксимації, дозволяє вибрати цю константу, що виявляється залежною від статистичних характеристик сигналів і завад. Зокрема, значення γ_0 може бути обране з умови

$$0 < \gamma < \frac{2}{\lambda_{\max}}$$

або з умови

$$0 < \gamma < \frac{2}{tr[R_x]},$$

де $R_x = \mathrm{M}\{\boldsymbol{x}\boldsymbol{x}^T\}$ — коваріаційна матриця; $\mathrm{M}\{\cdot\}$ — символ математичного сподівання; $tr[R_x]$ — слід матриці R_x

$$tr[R_x] = \sum_{i=1}^{N} \lambda_i = \sum_{i=1}^{N} r_{i,xx} \ge \lambda_{\max};$$

λ_i — власні числа матриці R_x; $r_{i,xx}$ — діагональні елементи коваріаційної матриці R_x; $\lambda_{\max}$ — найбільш власне число R_x.

Вибір коефіцієнта γ у вигляді

$$\gamma(k) = \|\boldsymbol{x}(k)\|^{-2}$$

призводить до *алгоритму Качмажа*, відомому в теорії ШНМ як *алгоритм Уідроу — Гоффа* [6]. У теорії ж оцінювання цей алгоритм називають *нормалізованим алгоритмом МНК*. Більш докладно цей алгоритм розглядатиметься нижче.

Найбільш відомим є *рекурентний алгоритм МНК (РМНК)*, що виходить із (2.51) при виборі замість скалярного коефіцієнта $\gamma(k)$ матричного змінного коефіцієнта $\Gamma(k)$

$$\Gamma(k) = P(k)\,[\lambda + \boldsymbol{x}^T(k)P(k)\boldsymbol{x}(k)]^{-1}, \qquad (2.54)$$

де $P^{-1}(k) = \sum_{i=1}^{k} \lambda^{k-i} \boldsymbol{x}(i)\boldsymbol{x}^T(i)$ — коваріаційна матриця; $0 < \lambda \le 1$ — параметр зважування інформації.

Матриця $P(k)$ допускає рекурентне обчислення. Тому в остаточному вигляді *РМНК із експонентним зважуванням інформації* має вигляд

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \alpha(k)P(k)\,\boldsymbol{x}(k)e(k); \qquad (2.55)$$

$$P(k+1) = \frac{1}{\lambda}\Big[I - \alpha(k)P(k)\boldsymbol{x}(k)\boldsymbol{x}^T(k)\Big]P(k), \qquad (2.56)$$

де $\alpha(k) = [\lambda + \boldsymbol{x}^T(k)P(k)\boldsymbol{x}(k)]^{-1}$; I — одинична матриця.

Використовуваний у (2.55), (2.56) параметр λ забезпечує зважування інформації, тобто надання більшого значення інформації, що знову надходить.

Даний механізм оцінювання важливості інформації особливо ефективний під час настроювання ваг, що змінюються в часі (дослідження нестаціонарних об'єктів). При $\lambda = 1$ алгоритм (2.55), (2.56) переходить у РМНК.

Різновидом РМНК є *багатокроковий (l-кроковий) проекційний алгоритм*, що використовує, на відміну від (2.55–2.56), фіксовану кількість інформації та має вигляд [71]:

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \gamma(k)\boldsymbol{r}_l(k); \qquad (2.57)$$

$$\begin{aligned} \boldsymbol{r}_l(k) &= R_{l-1}(k-1)\boldsymbol{x}(k)\Gamma^{-1}(k)e(k) + \\ &+\big(1-\gamma(k-1)\big)\big(I - R_l(k-1)\boldsymbol{x}(k)\boldsymbol{x}^T(k)\Gamma^{-1}(k)\big)\boldsymbol{r}_{l-1}(k-1); \end{aligned} \qquad (2.58)$$

$$\begin{aligned} R_i(k-l+i) &= \big(I - R_{i-1}(k-l+i-1)\boldsymbol{x}(k-l+i)\times \\ &\times \boldsymbol{x}^T(k-l+i)\Gamma^{-1}(k-l+i)\big)R_{i-1}(k-l+i-1), \end{aligned} \qquad (2.59)$$

де $r_0 = (0\ \ 0\ \ \ldots\ \ 0)^T$; $R_0 = I$; $i = \overline{1,l}$; l = const — пам'ять алгоритму (ця величина вибирається меншою, ніж розмірність завдання); $\gamma(i) \in (0, 2)$.

Використання в (2.57–2.59) фіксованої пам'яті робить цей алгоритм особливо привабливим у процесі навчання мереж, параметри яких змінюються в часі.

Якщо нейрони описуються нелінійними активаційними функціями $f(\boldsymbol{w},\boldsymbol{x}(k))$, то, скориставшись розкладанням вихідних сигналів нейронів у ряд Тейлора

$$y(k+1) = f(\boldsymbol{w}(k),\boldsymbol{x}(k+1)) + H^T(k+1)\big(\boldsymbol{w}^* - \boldsymbol{w}(k)\big) + \rho(k+1), \qquad (2.60)$$

де $H(k+1)$ — матриця розмірності $S \times M$ вигляду

$$H(k+1) = \left.\frac{\partial f(\boldsymbol{w},\boldsymbol{x}(k+1)}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}=\boldsymbol{w}(k)};$$

S — розмірність вектора ваг; M — розмірність вихідного сигналу; $\rho(k)$ — залишок розкладання, що враховує члени більш високих порядків, можна отримати такий алгоритм навчання:

$$w(k+1)=w(k)+K(k+1)\left(y^*(k+1)-f(w(k),x(k+1))\right); \quad (2.61)$$

$$K(k+1)=\frac{1}{\lambda}P(k)H(k+1)\left[I+\frac{1}{\lambda}H^T(k+1)P(k)H(k+1)\right]^{-1}; \quad (2.62)$$

$$P(k+1)=\frac{1}{\lambda}\left[I-K(k+1)H^T(k+1)\right]^{-1}, \quad (2.63)$$

відомий у теорії оцінювання як *фільтр Калмана*. Матриця $K(k+1)$, що обчислюється відповідно до (2.62), називається *матрицею посилення Калмана*.

Алгоритм навчання (2.55–2.56) випливає з (2.61–2.63) у процесі використання лінійної апроксимації активаційної функції нейронів.

Якщо вихідний сигнал i-го нейрона подати у вигляді

$$y_i(k+1)=f(w_i^T(k)x_i(k+1)), \quad (2.64)$$

то може бути отриманий такий алгоритм навчання його ваг, що заснований на *розширеному фільтрі Калмана* [72]:

$$w_i(k+1)=w_i(k)+K_i(k+1)e_i(k+1); \quad (2.65)$$

$$K_i(k+1)=\frac{1}{\lambda}P_i(k)Z_i(k+1)\left[I+\frac{1}{\lambda}Z_i^T(k+1)P_i(k)Z_i(k+1)\right]^{-1}; \quad (2.66)$$

$$P_i(k+1)=\frac{1}{\lambda}\left[I-K_i(k+1)Z_i^T(k+1)\right]P_i(k+1), \quad (2.67)$$

де $Z_i(k+1)=x_i(k+1)\nabla_w f\left(w_i^T(k)x_i(k+1)\right)$; $\nabla_w f(\cdot)=\dfrac{\partial f(\cdot)}{\partial w}$.

Наведені вище алгоритми навчання реалізують методи *оптимізації першого порядку*, у яких використовується обчислення градієнта функціонала. Серед цих методів найбільшу швидкість збіжності має *метод сполучених градієнтів* [73–74].

Ще більшу швидкість збіжності мають *методи другого порядку*, що вимагають обчислення других похідних мінімізованого

функціонала. Серед таких методів у першу чергу слід відзначити *метод Ньютона*:

$$w(k+1)=w(k)-H_k^{-1}g_k, \quad (2.68)$$

де H_k^{-1} — матриця, зворотна матриці Гессе

$$H_k=\nabla_w^2 I(w)=\begin{bmatrix}\frac{\partial^2 I(w)}{\partial w_1^2} & \frac{\partial^2 I(w)}{\partial w_1 w_2} & \cdots & \frac{\partial^2 I(w)}{\partial w_1 w_N}\\ \frac{\partial^2 I(w)}{\partial w_2 w_1} & \frac{\partial^2 I(w)}{\partial w_2^2} & \cdots & \frac{\partial^2 I(w)}{\partial w_2 w_N}\\ \cdots & \cdots & \cdots & \cdots\\ \frac{\partial^2 I(w)}{\partial w_N w_1} & \frac{\partial^2 I(w)}{\partial w_N w_2} & \cdots & \frac{\partial^2 I(w)}{\partial w_N^2}\end{bmatrix}, \quad (2.69)$$

$$g_k=\left(\frac{\partial I(w)}{\partial w_1} \quad \frac{\partial I(w)}{\partial w_2} \quad \cdots \quad \frac{\partial I(w)}{\partial w_N}\right)^T. \quad (2.70)$$

Ці похідні обчислюються у точці $w=w(k)$.

Якщо як критерій навчання використовується

$$I(w)=0{,}5\sum_{i=1}^{M}\left(y_i^*-y_i\right)^2=0{,}5\sum_{i=1}^{M}e_i^2, \quad (2.71)$$

то

$$g_k=\frac{\partial I(w)}{\partial w}=0{,}5\left[\frac{\partial\sum_{i=1}^{M}e_i^2}{\partial w_1} \quad \cdots \quad \frac{\partial\sum_{i=1}^{M}e_i^2}{\partial w_N}\right]^T=$$

$$=\left[\sum_{i=1}^{M}e_i\frac{\partial e_i}{\partial w_1} \quad \cdots \quad \sum_{i=1}^{M}e_i\frac{\partial e_i}{\partial w_N}\right]^T=J^T e, \quad (2.72)$$

де J — якобіан, що має вигляд

$$J=\begin{bmatrix}\frac{\partial e_1}{\partial w_1} & \frac{\partial e_1}{\partial w_2} & \cdots & \frac{\partial e_1}{\partial w_N}\\ \frac{\partial e_2}{\partial w_1} & \frac{\partial e_2}{\partial w_2} & \cdots & \frac{\partial e_2}{\partial w_N}\\ \cdots & \cdots & \cdots & \cdots\\ \frac{\partial e_M}{\partial w_1} & \frac{\partial e_M}{\partial w_2} & \cdots & \frac{\partial e_M}{\partial w_N}\end{bmatrix}. \quad (2.73)$$

Оскільки елементи матриці $H = \nabla_w^2 I(w)$ визначаються за формулою

$$[H]_{l,m} = \left[\nabla_w^2 I(w)\right]_{l,m} = \frac{\partial^2 I(w)}{\partial w_l \partial w_m} = \sum_{i=1}^{M}\left(\frac{\partial e_i}{\partial w_l}\frac{\partial e_i}{\partial w_m} + e_i \frac{\partial^2 e_i}{\partial w_l \partial w_m}\right),$$

то з урахуванням (2.69), (2.73) можна записати

$$H = J^T J + S,$$

де $S = \sum_{i=1}^{M} e_i \nabla_w^2 e_i$.

З наближенням до мінімуму елементи матриці S стають малими, і матриця Гессе може бути апроксимована

$$H \approx J^T J. \qquad (2.74)$$

Підставляючи (2.74) у (2.68), отримуємо такий алгоритм *методу Ньютона*:

$$w(k+1) = w(k) - \left[J_k^T J_k\right]^{-1} J_k^T e_k, \qquad (2.75)$$

де $e_k = (e_1, e_2, \ldots, e_m)^T$; індекс $k = 0, 1, 2, \ldots$ означає прискорений машинний час.

Для підвищення обчислювальної стійкості алгоритму навчання замість (2.74) використовують апроксимацію

$$H \approx J^T J + \mu I, \qquad (2.76)$$

де μ — досить мале додатне число.

Підстановка (2.76) в (2.68) призводить до *алгоритму Левенберга — Маркуардта*

$$w(k+1) = w(k) - \left[J_k^T J_k + \mu I\right]^{-1} J_k^T e_k. \qquad (2.77)$$

Ще один метод, що використовує обчислення градієнта й широко застосовуваний у процесі навчання багатошарових ШНМ — *метод зворотного поширення помилки* (*backpropagation*) — буде розглянутий у наступному розділі. Зазначимо тільки, що дельта-правило є оптимальним у певному сенсі для одношарових ШНМ, тому що з його допомогою визначається вагова матриця, що точно відтворює реакцію нейрона у випадку лінійно незалежних вхідних векторів. У багатошарових ШНМ це правило не застосовується, оскільки не зрозуміло, яким чином настоювати вагові коефіцієнти нейронів прихованих шарів, які б зменшували помилку вихідного сигналу мережі. Ця проблема називається «*credit assignment-проблеми*» і вирішується за допомогою алгоритму зворотного поширення помилки. Цей алгоритм може бути застосований до ШНМ, що має будь-яке число прихованих шарів. Відповідно до цього алгоритму шляхом мінімізації вихідної помилки спочатку настроюються ваги останнього (вихідного) шару, потім попереднього й т. д. до вхідного шару. Однак, якщо дельта-правило гарантує збіжність процесу навчання, то алгоритм зворотного поширення помилки такої гарантії не дає, і навчання може застрягнути в одному з локальних мінімумів.

2.4.7. Навчання з підкріплюванням

Цей вид навчання також заснований на оптимізації деякого критерію. Крім того, у випадку навчання з підкріплюванням, як ми вже зазначали раніше, також є присутнім учитель або зовнішній арбітр, що дає лише вказівку про зменшення або збільшення значень параметрів мережі. Оскільки для забезпечення адаптації мережі у всьому заданому просторі сигналів необхідні випадкові її вихідні сигнали, то для настроювання мережі звичайно використовують алгоритми, застосовувані для навчання стохастичних нейронів.

Так, у роботі [75] запропоновано алгоритм навчання з підкріплюванням, аналогічний (2.52) і що використовує для настроювання мережі послідовність навчальних пар вигляду $\{x(k), r(k)\}$, де $x(k)$ — вектори вихідних сигналів, а $r(k) \in \{-1,1\}$ — вказівки вчителя ($r(k) = 1$ свідчить про правильний напрямок у зміні ваг і про необхідність збільшення їхнього значення; $r(k) = -1$ сигналізує про помилковий напрямок і про необхідність зменшення значення ваг). Крім того, у цьому алгоритмі, на відміну від (2.52), використовується змінний коефіцієнт навчання $\alpha(r(k))$

$$w(k+1) = w(k) + \alpha(r(k))[y^*(k) - < y(k) >][1 - < y^2(k) >]x(k), \qquad (2.78)$$

де

$$\alpha(r(k)) = \begin{cases} \alpha^+ \\ \alpha^-, \end{cases} \quad y^*(k) = \begin{cases} y(k) & \text{при } r(k) = +1 \quad \text{(заохочування)} \\ -y(k) & \text{при } r(k) = -1 \quad \text{(штраф)}, \end{cases}$$

$\alpha^+ >> \alpha^- > 0$.

Вибір $\alpha^+ >> \alpha^-$ служить для забезпечення швидкого навчання й повільного забування.

У роботі [76] запропоновано модифікацію алгоритму (2.78) такого вигляду:

$$w(k+1) = w(k) + \alpha(r(k))[r(k)(y^*(k) - < y(k) >) - \\ - (1 - r(k))(y(k) + < y^2(k) >)]x(k), \quad (2.79)$$

де $r(k) \in [0,1]$.

На завершення зазначимо, що хоча цей вид навчання й досліджено у ряді робіт, він не належить до числа популярних. Тому надалі ми розглядатимемо перші два види навчання: навчання з учителем і навчання без учителя.

Докладніше методи й алгоритми навчання висвітлено в монографії [77].

Контрольні запитання та завдання

1. Яка структура штучного нейрона?
2. Наведіть приклади вхідних операторів.
3. Що таке функція активації? Наведіть приклади різних функцій активації.
4. Дайте характеристику нейрона Маккаллоха — Піттса.
5. У чому особливість нейрона Фукушіми?
6. Поясніть принцип функціонування нейрона Гопфілда.
7. Поясніть особливості нейрона Гроссберґа.
8. Наведіть приклади топологій ШНМ.
9. Які існують підходи до навчання ШНМ?
10. Поясніть правило навчання Гебба.
11. На чому засноване дельта-правило?
12. У чому суть конкурентного навчання?
13. Що є основою стохастичного навчання?
14. Наведіть приклади градієнтних методів навчання.

3. РАННІ АРХІТЕКТУРИ ШНМ

Перші ШНМ, розроблені у 50–60-х роках XX ст., мали просту одношарову архітектуру й могли вирішувати досить обмежене коло задач. Однак результати досліджень властивостей цих мереж виявилися настільки цікавими, що викликали цілий потік робіт, присвячених створенню мереж, які мають більш складну структуру й здатні вирішувати значно складніші задачі.

3.1. Одношарові ШНМ

3.1.1. Одношаровий персептрон

Термін «персептронний нейрон» введено у роботі *У. Маккаллоха* й *У. Піттса* [1]. У більшій частині їхньої роботи використовувалася модель (2.17) з активаційною функцією вигляду (2.10). Із досягненням зваженою сумою значення, більшого заданого порога θ, на виході нейрона з'являвся одиничний сигнал, якщо ж зважена сума була менше θ, сигнал був відсутній. Системи, що використовують дані моделі нейронів, отримали назву *персептронів*.

Значний інтерес до персептронів викликаний роботою *Ф. Розенблатта* [4], у якій він досліджував нейромережеву модель сітківки (RETINA) — *фотоперсептрон*. Згодом такий підхід широко використовувався для моделювання обробки оптичних сигналів. Фотоперсептрон зображений на рис. 3.1 і складається, відповідно до концепції Розенблатта, із трьох шарів, що послідовно здійснюють попередню обробку (розбивання) образу, оцінку його характеристик і розпізнавання:

— сітківка (RETINA);
— асоціативний шар;
— вихідний шар.

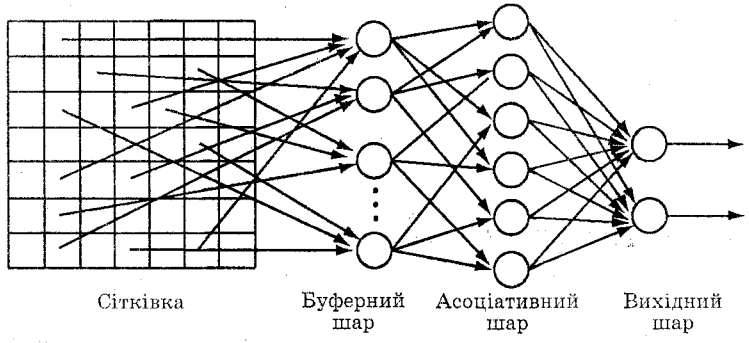


Рис. 3.1. Персептрон Розенблатта

Попередня обробка образу не залежить від його виду. Однак результат цієї обробки має забезпечити можливість розпізнавання образів на основі аналізу їхніх характеристик. Нарешті, вихідний шар (класифікатор) аналізує характеристики знову пропонованого образу й установлює його відповідність одному з раніше поданих.

Сигнали першого шару, сітківки, подані у двійковій формі, надходять на асоціативний шар, причому в загальному випадку не всі нейрони першого шару пов'язані з усіма нейронами другого шару. При встановленні цих зв'язків виникає можливість структуризації вхідних даних, тобто виділення й об'єднання в так звані *рецептивні поля* найбільш важливих ознак (областей).

У зв'язку з цим під рецептивним полем розуміють множину всіх нейронів вхідного шару, пов'язаних з одним нейроном асоціативного шару. Зв'язки між нейронами асоціативного й вихідного шарів варіабельні й можуть модифікуватися шляхом зміни вагових коефіцієнтів.

Нейрони асоціативного шару мають лінійні активаційні функції, тому під час надходження із сітківки вхідних (дратівних) сигналів вони посилають імпульси на вихідний шар, де й відбувається додавання зважених імпульсів. У вихідному шарі використовуються уні- або біполярна активаційні функції (2.10) або (2.11). Якщо сума зважених імпульсів перевищує деяке задане порогове значення, виробляється одиничний вихідний сигнал, якщо не перевищує — нульовий (для уніполярної) або –1 (для біполярної функції активації). У зв'язку з цим персептрон може розглядатися як двошарова ШНМ прямого поширення.

Можливості одношарового персептрона можна проаналізувати на прикладі реалізації булевих функцій двох змінних (табл. 3.1).

Таблиця 3.1

x_1	x_2	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Відповідний спрощений персептрон зображено на рис. 3.2.

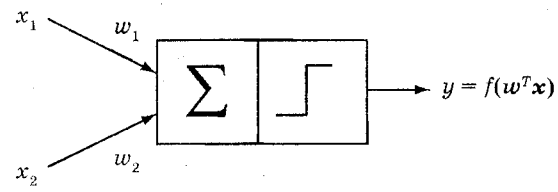


Рис. 3.2. Персептрон:
w_1 й w_2 — вагові коефіцієнти; $\theta > 0$ — поріг вихідного нейрона

Приклад 3.1. Для реалізації функції «І» необхідне виконання таких нерівностей:

Якщо $x_1 = 0, x_2 = 0$, $w_1x_1 + w_2x_2 = 0 < \theta$,

$x_1 = 0, x_2 = 1$, $w_1x_1 + w_2x_2 = w_2 < \theta$,

$x_1 = 1, x_2 = 0$, $w_1x_1 + w_2x_2 = w_1 < \theta$,

$x_1 = 1, x_2 = 1$, $w_1x_1 + w_2x_2 = w_1 + w_2 \geq \theta$.

Як видно з наведених співвідношень, існує нескінченна множина значень w_1 й w_2, при яких дані нерівності виконуються.

Особливістю персептрона є здатність до навчання його вагових коефіцієнтів. У режимі навчання йому подають образи — пари, що навчають $(\boldsymbol{x}, \boldsymbol{y}^*)$, на основі яких він настроює свої параметри так, щоб з появою деякого вхідного вектора $\boldsymbol{x}$ на його виходах з'являлися відповідні цьому входу сигнали $\boldsymbol{y}$. Процес навчання завершується, якщо всі пари $(\boldsymbol{x}, \boldsymbol{y}^*)$ асоціюються правильно. Алгоритм навчання персептрона реалізує спосіб навчання з учителем, при якому вихідна помилка мінімізується. Досліджуючи клас самонастроювальних мереж, правильніше мереж, що навчаються без учителя, Ф. Розенблатт довів, що за певних умов алгоритм навчання завжди збігається. Однак виявилося, що персептрони не здатні

навчатися вирішенню низки простих задач. Точний доказ цього факту був здійснений *М. Мінські* [8].

Найбільш вражаюча обмеженість найпростішого персептрона виявляється при спробі реалізації функції «що виключає АБО» (функції y_6 у табл. 3.1).

Приклад 3.2. Застосування персептрона, зображеного на рис. 3.2, для реалізації функції y_6 вимагає виконання таких умов:

$$\begin{aligned} x_1 = 0,\ x_2 = 0, &\qquad w_1x_1 + w_2x_2 = 0 < \theta,\\ x_1 = 0,\ x_2 = 1, &\qquad w_1x_1 + w_2x_2 = w_2 \ge \theta,\\ x_1 = 1,\ x_2 = 0, &\qquad w_1x_1 + w_2x_2 = w_1 \ge \theta,\\ x_1 = 1,\ x_2 = 1, &\qquad w_1x_1 + w_2x_2 = w_1 + w_2 < \theta. \end{aligned}$$

З перших трьох нерівностей випливає, що необхідно задати $\theta > 0$. Однак у цьому випадку отримуємо

$$w_1 + w_2 \ge 2\theta > \theta,$$

тобто остання умова не виконується. Отже, на найпростішому персептроні реалізувати функцію «що виключає АБО» неможливо.

Обмежені можливості найпростішого персептрона, виявлені й доведені Мінські, призвели до того, що дослідження в цій галузі на певний час припинилися. Це можна пояснити ще й тим, що в той час хоча й розуміли необхідність переходу до багатошарових персептронів, методи їхнього навчання ще не були відомі.

3.1.2. Навчання персептрона

Існують різні шляхи реалізації процесу навчання персептрона, однак у їхній основі лежить таке правило: вагові коефіцієнти персептрона змінюються тільки тоді, коли виникає розбіжність між його фактичною й бажаною реакціями.

Схему навчання персептрона наведено на рис. 3.3.

Передбачається, що активаційна функція має вигляд (2.11). Процес навчання полягає в послідовному поданні множини пар, що навчають $(\boldsymbol{x}_p, y_p^*)$, $p = \overline{1,P}$, де $\boldsymbol{x}_p$, y_p^* — $(N \times 1)$-вхідний вектор і бажаний вихідний сигнал p-ї пари, що навчає, відповідно, за допомогою яких визначається необхідний вектор вагових коефіцієнтів $\boldsymbol{w}^*$ такий, що

$$y_p = \operatorname{sgn}(\boldsymbol{w}^{*T}\boldsymbol{x}_p) = y_p^*,\ p = \overline{1,P}. \qquad (3.1)$$

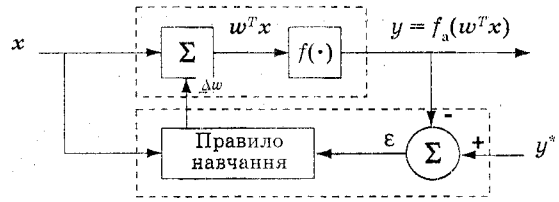


Рис. 3.3. Схема навчання персептрона

У цьому випадку вектор $\boldsymbol{w}^*$ забезпечить правильну класифікацію персептроном усіх пар, що навчають, із поданої множини (завершується цикл навчання).

З (3.1) випливає, що гіперплощина $\boldsymbol{w}^{*T}\boldsymbol{x}_p = 0$ ділить вхідний простір на два підпростори. Для $y_p^* = 1$ має виконуватися умова $\boldsymbol{w}^{*T}\boldsymbol{x}_p > 0$, а для $y_p^* = -1$ — умова $\boldsymbol{w}^{*T}\boldsymbol{x}_p < 0$.

Алгоритм навчання може бути записаний у такий спосіб:

$$\boldsymbol{w}_{p+1} = \boldsymbol{w}_p + \gamma e_p \boldsymbol{x}_p, \qquad (3.2)$$

де $e_p = y_p^* - y_p$ — помилка класифікації; γ — параметр, що впливає на швидкість збіжності алгоритму (тривалість процесу навчання). Корекція ваг відповідно до (3.2) може відбуватися в режимах *online* й *offline*.

У режимі *online* корекція відбувається при поданні кожної навчальної пари $(\boldsymbol{x}_p, y_p^*)$, $p = \overline{1,P}$.

У режимі *offline* у M-му циклі (епосі) навчання подаються всі пари $(\boldsymbol{x}_p, y_p^*)$ й обчислюється середнє значення помилки класифікації

$$\bar{e}_M = \frac{1}{P}\sum_{p=1}^{P}\left(y_{P,M}^* - y_{P,M}\right), \qquad (3.3)$$

що й використовується в алгоритмі навчання. Тут $y_{P,M}^*, y_{P,M}$ — бажані й реальні вихідні сигнали при пред'явленні p-ї пари, що навчає, в M-му циклі навчання відповідно.

Алгоритм навчання в M-му циклі приймає вигляд

$$\boldsymbol{w}_{M+1} = \boldsymbol{w}_M + \gamma \bar{e}_M \boldsymbol{x}_P. \qquad (3.4)$$

Реалізація даного навчання пов'язана з необхідністю попереднього обчислення значень вихідних змінних для всіх пар, що навчають.

3.1.3. Теорема збіжності для персептрона

Ця теорема, доведена Розенблаттом, стверджує, що якщо задача має розв'язок, то він може бути отриманий за кінцеве число тактів (ітерацій). З погляду класифікації це означає, що персептрон може навчитися правильно класифікувати подані йому образи на кінцевому числі пар, що навчають.

Це може бути показано в такий спосіб.

Нехай $y_k^* \in \{0,1\}$ і вхідні сигнали обмежені за величиною, тобто

$$\|x(k)\|^2 \le A, \tag{3.5}$$

де $\|x(k)\|^2 = x^T(k)x(k)$ — евклідова норма; k — деякий такт навчання.

Використовувана в алгоритмі (3.2) помилка $e = y^* - y$ може приймати значення ±1 або 0. При $e = 0$ (правильна класифікація) зміни ваг не відбувається. Позначимо розв'язок: вектор ваг, що може правильно класифікувати всі пропоновані образи, через w^*. Для цього вектора ваг маємо

$$\begin{aligned} &w^{*T} w(k) > \delta > 0, \quad \text{якщо} \quad y^*(k) = 1; \\ &w^{*T} w(k) < -\delta < 0, \quad \text{якщо} \quad y^*(k) = 0. \end{aligned} \tag{3.6}$$

Помноживши обидві частини (3.2) зліва на $w^T(k+1)$, отримаємо

$$\|w(k+1)\|^2 = \|w(k)\|^2 + 2\gamma e(k) w^T(k)x(k) + \gamma^2 e^2(k)\|x(k)\|^2. \tag{3.7}$$

Для простоти приймемо, що $w(0) = 0$. Оскільки

$$w^T(k)x(k) \begin{cases} > 0 \text{ для } e(k) = -1; \\ < 0 \text{ для } e(k) = +1 \end{cases} \tag{3.8}$$

(корекція коефіцієнтів відбуватиметься тільки в цьому випадку), то $2\gamma e(k) w^T(k)x(k) < 0$ і з урахуванням (3.5) можна записати

$$\begin{aligned} \|w(k+1)\|^2 &\le \|w(k)\|^2 + \gamma^2 e^2(k)\|x(k)\|^2 \le \|w(k)\|^2 + \gamma^2 A = \\ &= \gamma^2 \sum_{i=1}^{k} e^2(k)\|x(i)\|^2 \le \gamma^2 kA. \end{aligned} \tag{3.9}$$

Звідси отримуємо верхню межу для $\|w(k+1)\|$

$$\|w(k+1)\| \le \gamma (kA)^{1/2}. \tag{3.10}$$

З нерівності Коші — Шварца випливає, що

$$(w^{*T} w(k)) \le \|w^*\| \, \|w(k)\|.$$

Тому з урахуванням (3.6) отримуємо нижню межу довжини вектора ваг на k-й ітерації

$$\|w(k+1)\| \ge \frac{\left(w^{*T} w(k+1)\right)}{\|w^*\|} = \frac{\sum_{i=1}^{K} \gamma e(i) w^{*T} x(i)}{\|w^*\|} > \frac{\gamma k \delta}{\|w^*\|}. \tag{3.11}$$

Поєднуючи (3.10) і (3.11), отримуємо

$$k\gamma\delta \|w^*\|^{-1} \le \|w(k)\| \le \gamma (kA)^{0,5}. \tag{3.12}$$

Ліва частина нерівності (3.12) зростає лінійно зі збільшенням k і швидше правої частини. Тому число k обмежене. Таким чином, алгоритм навчання персептрона збігатиметься за кінцеве число ітерацій.

Приклад 3.3. Заданий тривходовий $x = (x_1\ x_2\ x_3)^T$ одношаровий персептрон з вектором ваг $w(0) = (0{,}5\ \ 0{,}1\ \ 1)^T$ й активаційною функцією виду $f(\cdot) = \operatorname{sgn}(\cdot)$. Слід змінити ваговий вектор так, щоб з появою вхідного сигналу $x = (-1\ -1\ 1)^T$ на виході персептрона з'являвся сигнал $y^* = -1$.

З наявними ваговими коефіцієнтами персептрон відпрацьовує вихідний сигнал

$$y(0) = \operatorname{sgn}(w^T(0)\,x) = \operatorname{sgn}\left[(0{,}5\ \ 0{,}1\ \ 1) \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix} \right] = \operatorname{sgn}(0{,}4) = 1,$$

що не відповідає необхідному.

Корекція ваг за алгоритмом (3.2) з $\gamma = 0{,}01$ дає

$$w(1) = w(0) - \gamma(y^* - y(1))x = \begin{pmatrix} 0{,}5 \\ 0{,}1 \\ 1 \end{pmatrix} - 0{,}01 \cdot 2 \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0{,}52 \\ 0{,}12 \\ 0{,}98 \end{pmatrix}.$$

У цьому випадку

$$y(1) = \operatorname{sgn}(w^T(1)x) = \operatorname{sgn}\left[(0{,}52\ \ 0{,}12\ \ 0{,}98) \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix} \right] = \operatorname{sgn}(0{,}34) = 1.$$

Через те, що реакція персептрона на вхідний сигнал неправильна, знову корегуються ваги

$$w(2) = w(1) - \gamma(y^* - y(1))x = \begin{pmatrix} 0{,}52 \\ 0{,}12 \\ 0{,}98 \end{pmatrix} - 0{,}01 \cdot 2 \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0{,}54 \\ 0{,}14 \\ 0{,}96 \end{pmatrix}.$$

При цьому значення вихідного сигналу персептрона дорівнюватиме

$$y(2) = \operatorname{sgn}(w^T(2)x) = \operatorname{sgn}\left[(0{,}54 \;\; 0{,}14 \;\; 0{,}96) \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix} \right] = \operatorname{sgn}(0{,}26) = 1.$$

Неправильна реакція персептрона свідчить про необхідність подальшої корекції ваг. Діючи за аналогією, отримуємо, що вектор ваг і вихідний сигнал персептрона змінюватимуться в такий спосіб:

$$w(4) = (0{,}58 \;\; 0{,}18 \;\; 0{,}92)^T; \;\; y(4) = 1;$$

$$w(5) = (0{,}60 \;\; 0{,}20 \;\; 0{,}90)^T; \;\; y(5) = 1;$$

$$w(6) = (0{,}62 \;\; 0{,}22 \;\; 0{,}88)^T; \;\; y(6) = 1;$$

$$w(7) = (0{,}64 \;\; 0{,}24 \;\; 0{,}86)^T; \;\; y(7) = -1.$$

Після сьомого такту $y^* - y(7) = 0$. А оскільки реакція персептрона на заданий вхідний сигнал зараз правильна, навчання завершене.

Нескладно побачити, що вибір параметра γ впливає на тривалість процесу навчання. Так, вибираючи в розглянутому прикладі $\gamma = 0{,}02$, отримаємо

$$w(1) = (0{,}6 \;\; 0{,}2 \;\; 0{,}9)^T; \;\; y(1) = 1;$$

$$w(2) = (0{,}70 \;\; 0{,}20 \;\; 0{,}90)^T; \;\; y(5) = 1;$$

тобто навчання завершується за 2 такти.

При виборі ж $\gamma = 0{,}05$ відразу отримуємо один із можливих розв'язків

$$w(1) = (0{,}90 \;\; 0{,}50 \;\; 0{,}60)^T; \;\; y(1) = -1.$$

Мейсом запропоновано модифікації правила настроювання персептрона з лінійною активаційною функцією $z = w^T x$, у яких враховувалася величина реакції (активації) мережі, визначена деяким порогом Ф, що апріорно задається. Перше таке модифіковане правило має вигляд

$$w(k+1) = \begin{cases} w(k) + 0{,}5\gamma e(k)x(k)\|x(k)\|^2 & \text{при } z(k) \geq \Phi; \\ w(k) + \gamma y^*(k)x(k)\|x(k)\|^2 & \text{при } z(k) < \Phi, \end{cases} \qquad (3.13)$$

де $e(k) = y^*(k) - y(k) = y^*(k) - f(w^T(k)x(k))$.

У другому запропонованому ним правилі враховується той факт, що під час виконання умови $(z(k) \geq \Phi) \wedge (e(k) = 0)$ ваги мережі не мають змінюватися. У випадку ж слабкої реакції мережі (при незначному ступені її активації), визначеної, як і раніше, порогом Ф або при виникненні помилки між поточною й необхідною реакціями мережі, здійснюється корекція ваг. Вищенаведене враховується в алгоритмі у такий спосіб:

$$w(k+1) = \begin{cases} w(k) \;\; \text{при } (z(k) \geq \Phi) \wedge (e(k) = 0); \\ w(\mathrm{k}) + \gamma e(k)x(k)\|x(k)\|^2 \;\; \text{в іншому випадку.} \end{cases} \qquad (3.14)$$

Якщо множина поданих персептрону образів $\{x(k), y^*(k)\}$ є лінійно роздільною, то обидва ці алгоритми збігаються до розв'язку за кінцеве число кроків при $\gamma \in (0,2)$. Хоча в деяких випадках алгоритм (3.14) збігається швидше, ніж алгоритм (3.13), теоретичного обґрунтування цьому немає. Слід зазначити, що якщо алгоритм (3.13) ідейно ближчий до стандартного алгоритму навчання персептрона, то алгоритм (3.14) має багато спільною з алгоритмом *Уідроу — Гоффа*, що буде розглянутий нижче.

Згодом був запропонований персептрон із сигмоїдальною активаційною функцією (рис. 3.4).

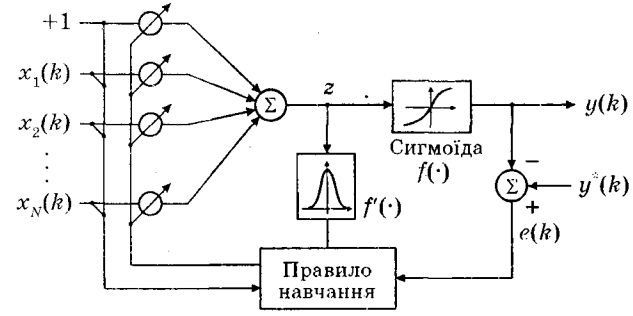


Рис. 3.4. Персептрон із сигмоїдальною активаційною функцією

Наявність у даному персептроні диференційованої активаційної функції дозволяє використати для його навчання градієнтні алгоритми

$$w(k+1) = w(k) - \gamma \nabla_w I(w). \qquad (3.15)$$

Якщо в якості мінімізувального функціонала вибрати квадратичний (2.47), а активаційна функція має вигляд (2.14), то

$$\nabla_w I(w) = -e(k)f'_{th}[z(k)]x(k) = -e(k)\alpha[1 - y^2(k)]x(k)$$

і алгоритм (3.15) записується в такий спосіб:

$$w(k+1) = w(k) + \gamma\alpha[1 - y^2(k)]e(k)x(k). \qquad (3.16)$$

Якщо ж активаційна функція обрана вигляду (2.13), то

$$w(k+1) = w(k) + \gamma\alpha y(k)[1 - y(k)]e(k)x(k). \qquad (3.17)$$

З наведених формул видно, що вибір сигмоїдальних функцій активації дозволяє отримати прості алгоритми навчання.

3.1.4. Адаліна

Першу лінійну нейронну мережу запропоновано *Б. Уідроу* й *М. Е. Гоффом* у роботі [6] 1960 року. Розроблену ними модель було названо *адаптивним нейроном*, а потім *ADALINE* — **ADA***ptive* **LI***near* **NE***uron*. З часом свою популярність як ШНМ вона втратила, і зараз *ADALINE* — це **ADA***ptive* **LIN***ear* **E***lement*. Адаліна мала структуру, аналогічну персептрону. Зважені вхідні сигнали $x_i \in \{-1,1\}$ підсумовувалися й перетворювалися біполярною активаційною функцією (2.11) у вихідний сигнал $y \in \{-1,1\}$. Зазначимо, що вхідні сигнали можуть бути також булевими змінними або реальними числами. Основна відмінність Адаліни від персептрона полягала у використовуваному алгоритмі навчання. Хоча навчання Адаліни також полягало в корекції її вагових коефіцієнтів на основі поданих пар, що навчають, і порівнянні реальних вихідних сигналів з бажаними (необхідними), Уідроу й Гофф, ґрунтуючись на працях Н. Вінера, пов'язаних з дослідженням фільтрів, використали для цього градієнтний алгоритм.

Алгоритм Уідроу — Гоффа є так званим *нормалізованим алгоритмом методу найменших квадратів* (МНК) або дельта-правилом і може бути отриманий у такий спосіб (рис. 3.5).

Подамо бажаний й реальний вихідні сигнали лінійної частини Адаліни на $(k+1)$-му такті навчання відповідно як

$$y^*(k) = w^{*T}x(k); \qquad (3.18)$$

$$u(k) = w^T(k-1)x(k), \qquad (3.19)$$

де w^*, $w(k)$ — оптимальний вектор вагових коефіцієнтів і його оцінка, отримана на попередньому k-му такті навчання.

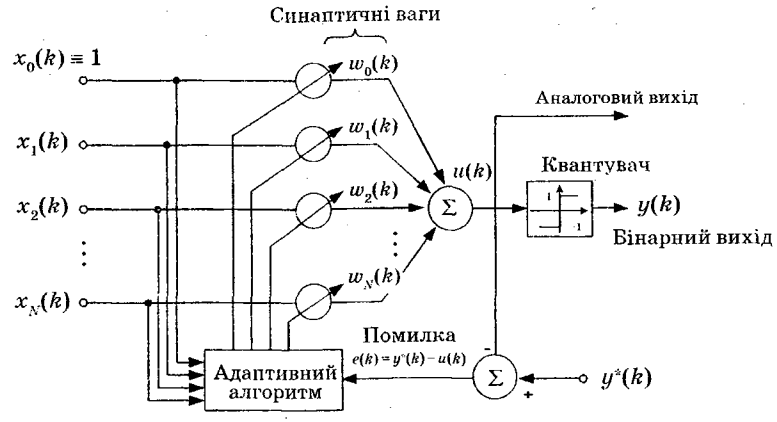


Рис. 3.5. Адаліна

Визначимо помилку реакції

$$e(k) = y^*(k) - u(k) = y^*(k) - w^T(k)x(k), \qquad (3.20)$$

яку використаємо для формування квадратичного критерію якості навчання

$$I(w,x) = e^2(k) = (y^*(k) - w^T(k)x(k))^2. \qquad (3.21)$$

Визначаючи шукані вагові коефіцієнти з умови $I(w,x) \to \min_w$, тобто з умови

$$\frac{\partial I(w,x)}{\partial w} = 0, \qquad (3.22)$$

і використовуючи градієнтний алгоритм (2.35), отримуємо таке правило корекції ваг:

$$w(k+1) = w(k) + \gamma(k)(y^*(k) - w^T(k)x(k))x(k). \qquad (3.23)$$

Тут γ — деякий позитивний параметр, що впливає на швидкість збіжності алгоритму (тривалість процесу навчання). Оптимальне значення γ, що забезпечує максимальну швидкість збіжності (3.18), визначається в такий спосіб.

Віднімемо з обох частин (3.21) w^*, тобто запишемо алгоритм стосовно помилок оцінювання $\theta(i) = w(i) - w^*$. Тоді з урахуванням (3.16)–(3.18)

$$\theta(k+1) = \theta(k) - \gamma(k)(\theta^T(k)x(k))x(k). \qquad (3.24)$$

Помножимо (3.24) ліворуч на $\theta^T(k)$

$$\|\theta(k+1)\|^2 = \|\theta(k)\|^2 - 2\gamma(k)(\theta^T(k)x(k)) + \gamma^2(k)(\theta^T(k)x(k))^2\|x(k)\|^2.$$

Визначаючи γ_k з умови мінімуму $\|\theta(k+1)\|^2$, тобто з умови

$$\frac{\partial\|\theta(k+1)\|^2}{\partial\gamma(k)} = -2\left(\theta^T(k)x(k)\right)^2 + 2\gamma(k)\left(\theta^T(k)x(k)\right)^2\|x(k)\|^2 = 0,$$

отримуємо такий вираз для $\gamma^{\text{опт}}(k)$ (*см.* (2.51)):

$$\gamma^{\text{опт}}(k) = \|x(k)\|^{-2}. \tag{3.25}$$

Підстановка (3.25) у (3.23) дає алгоритм

$$w(k+1) = w(k) + \frac{y^*(k) - w^T(k)x(k)}{\|x(k)\|^2}x(k), \tag{3.26}$$

що називається внаслідок використання в ньому нормалізації $x(k)\|x(k)\|^{-2}$ *нормалізованим алгоритмом МНК*. Зазначимо, що даний алгоритм уперше застосовано *Качмажем* [47] для розв'язання систем лінійних алгебраїчних рівнянь.

Записаний стосовно помилок оцінювання θ алгоритм (3.26) має вигляд

$$\theta(k+1) = \left(I - \frac{x(k)x^T(k)}{\|x(k)\|^2}\right)\theta(k). \tag{3.27}$$

Тут I — одинична матриця $N \times N$; $x(k)x^T(k)\|x(k)\|^{-2}$ — матриця проеціювання на вектор $x(k)$. Внаслідок використання операції проеціювання швидкість збіжності цього градієнтного алгоритму максимальна.

Алгоритм Уідроу — Гоффа має вигляд, аналогічний (3.26)

$$w(k+1) = w(k) + \alpha\frac{y^*(k) - w^T(k)\,x(k)}{\|x(k)\|^2}x(k), \tag{3.28}$$

де $\alpha \in (0,2)$.

Неважко показати, що оптимальне значення $\alpha = 1$ і $\alpha < 1$, якщо змінні вимірюються неточно.

Приклад 3.4. Розглянемо персептрон, заданий у прикладі 3.3, і здійснимо корекцію його параметрів за алгоритмом (3.26). Тоді

$$\|x(1)\|^2 = 3, \quad w^T(0)\,x(1) = (0{,}5 \quad 0{,}1 \quad 1)\begin{pmatrix}-1\\-1\\1\end{pmatrix} = 0{,}4,$$

$$w(1) = \begin{pmatrix}0{,}5\\0{,}1\\1\end{pmatrix} + \frac{(-1-0{,}4)}{3}\begin{pmatrix}-1\\-1\\1\end{pmatrix} = \begin{pmatrix}0{,}96\\0{,}56\\0{,}54\end{pmatrix};$$

і

$$y(1) = \operatorname{sgn}\left(w^T(1)x(1)\right) = \operatorname{sgn}\left[(0{,}96 \quad 0{,}56 \quad 0{,}54)\begin{pmatrix}-1\\-1\\1\end{pmatrix}\right] = \operatorname{sgn}(-0{,}98) = -1.$$

У роботі [79] наведено моделі Адаліни, що містять замість порогової активаційної функції сигмоїдальну (рис. 3.6), а також має як сигмоїдальну, так і порогову активаційні функції (рис. 3.7). Відповідні сигнали помилок використовуються для корекції вагових коефіцієнтів Адаліни.

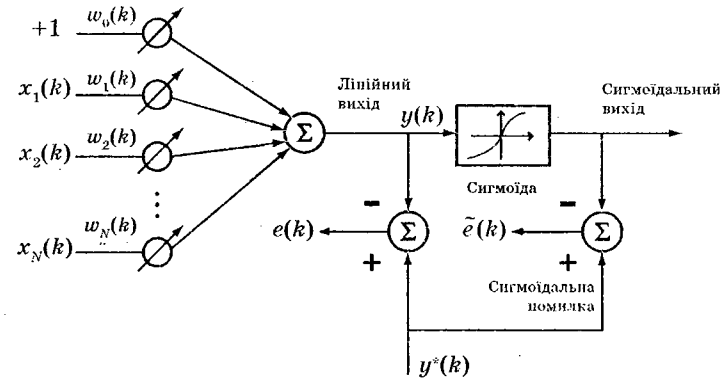


Рис. 3.6. Адаліна із сигмоїдальною функцією активації

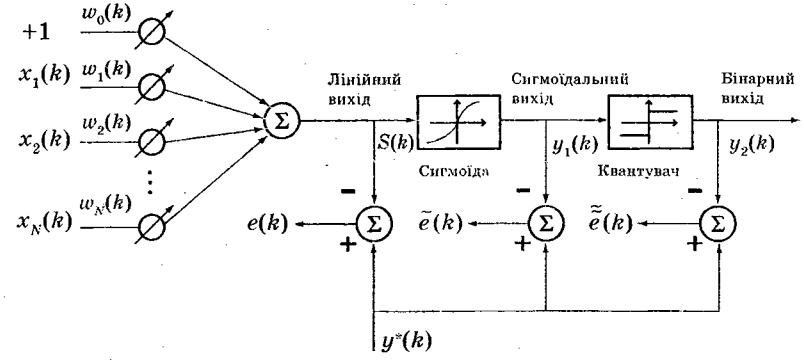


Рис. 3.7. Узагальнена Адаліна

3.1.5. Н-Адаліна

Н-Адаліна (N-ADALINE) є різновидом Адаліни, на вхід якої крім звичайних вхідних сигналів подаються їхні різні комбінації, тобто використовуються нелінійні перетворення вхідних сигналів. У ній реалізовано розроблений *О. Г. Івахненком* метод групового урахування аргументів, МГУА (*GMDH — Group Method of Data Handling*) [37, 46, 80, 81]. Н-Адаліну, вихідний сигнал якої обчислюється за формулою

$$y = w_0 + w_1 x_1 + w_2 x_1^2 + w_3 x_1 x_2 + w_4 x_2^2 + w_5 x_2, \qquad (3.29)$$

зображено на рис. 3.8.

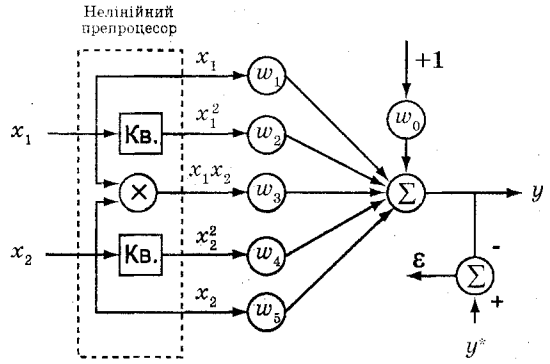


Рис. 3.8. Н-Адаліна

Як випливає з (3.29), вихід y є лінійною функцією відносно вагових коефіцієнтів w_i ($i = \overline{1,5}$) і нелінійною щодо вхідних сигналів x_1 й x_2. За аналогією можуть бути побудовані Н-Адаліни більш високих порядків. МГУА дозволяє вибрати оптимальну складність (кількість членів, степінь і т. д.) математичної моделі об'єкта.

Для навчання Адаліни використовують алгоритми навчання з учителем, наприклад, алгоритм Уідроу — Гоффа, що для Н-Адаліни, наведеної на рис. 3.8, має вигляд

$$w(k+1)(k) = w(k) + \alpha \frac{(y^*(k) - w^T(k) z(k))}{\|z(k)\|^2} z(k), \qquad (3.30)$$

де $w = (w_0, w_1, \ldots, w_5)^T$; $z = (1, x_1, x_1^2, x_1x_2, x_2^2, x_2)^T$; $y^*(k)$ — необхідний вихідний сигнал; α — параметр навчання.

3.1.6. Вхідна зірка Ґроссберґа

Вхідна зірка (*Instar*) *С. Ґроссберґа* є нейроном, що за структурою аналогічний Адаліні, призначений для розв'язання найпростіших задач розпізнавання образів і здійснює перетворення

$$y_j = f(w_j^T x + \theta_j), \qquad (3.31)$$

де

$$f(u_j) = \begin{cases} 1, & \text{якщо } u_j \geq 0, \\ 0 & \text{в іншому випадку.} \end{cases} \qquad (3.32)$$

Схему вхідної зірки зображено на рис. 3.10.

Нескладно побачити, що цей нейрон активізується (на виході з'являється 1) у випадку, якщо вектор вхідних сигналів $x(k)$ у деякому значенні близький до поточного вектора синаптичних ваг $w_i(k)$, тобто з виконанням умови

$$w_j^T(k) x(k) = \|w_j(k)\| \|x(k)\| \cos\varphi \geq \theta_j, \qquad (3.33)$$

де φ — кут між векторами $w_i(k)$ і $x(k)$; θ_i — сигнал зміщення, що задає поріг «близькості» векторів, що визначає спрацьовування вхідної зірки.

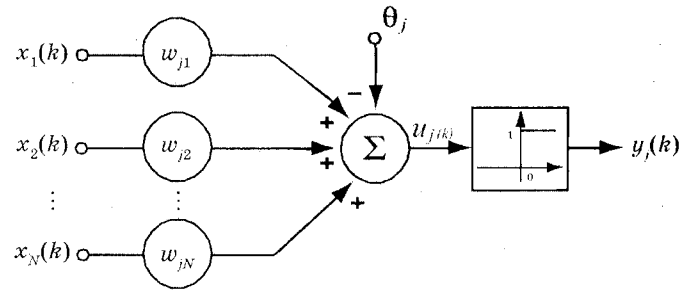


Рис. 3.10. Вхідна зірка

Якщо прийняти

$$\theta_j = \|w_j\| \|x\|, \qquad (3.34)$$

то зірка активується тільки у випадку, якщо вхідний сигнал співпадає з вектором синаптичних ваг, тобто розпізнається тільки

один образ. Чим менше значення θ_j, тим більше можливих образів можуть активувати нейрон, що стає при цьому все менш «розбірливим».

Навчання вхідної зірки здійснюється за допомогою модифікованого алгоритму (3.9), що набуває у цьому випадку вигляду

$$\boldsymbol{w}_j(k+1) = \boldsymbol{w}_j(k) + \gamma y_j(k)\boldsymbol{x}(k) - \alpha y_j(k)\boldsymbol{w}_j(k). \qquad (3.35)$$

Необхідність модифікації пов'язана з тим, що у випадку подачі на вхід нейрона послідовності $\boldsymbol{x}(k)$, що не активує зірку ($y_j(k) = 0$), відбувається поступове забування всієї накопиченої інформації. Дійсно, в цьому випадку алгоритм (3.9) набуває вигляду

$$\boldsymbol{w}_j(k+1) = (1-\alpha)\boldsymbol{w}_j(k). \qquad (3.36)$$

Характерною особливістю правила (3.35) є те, що самонавчання відбувається тільки в активованому стані, коли $y_j(k) = 1$.

Поклавши для простоти $\alpha = \gamma$, отримуємо так зване *стандартне правило самонавчання вхідної зірки*

$$\boldsymbol{w}_j(k+1) = \boldsymbol{w}_j(k) + \gamma y_j(k)(\boldsymbol{x}(k) - \boldsymbol{w}_j(k)), \qquad (3.37)$$

геометричну ілюстрацію якого наведено на рис. 3.10.

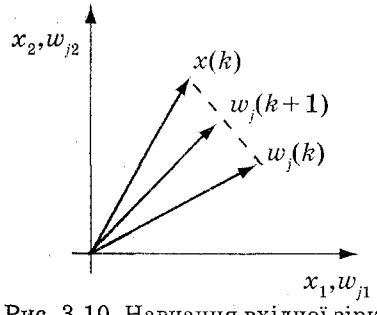


Рис. 3.10. Навчання вхідної зірки

При $y_j(k) = 0$ згідно з (3.37) навчання не відбувається, тобто $\boldsymbol{w}_j(k+1) = \boldsymbol{w}_j(k)$. При $y_j(k) = 1$ алгоритм природно набуває вигляду

$$\boldsymbol{w}_j(k+1) = \boldsymbol{w}_j(k) + \gamma(\boldsymbol{x}(k) - \boldsymbol{w}_j(k)) = (1-\gamma)\boldsymbol{w}_j(k) + \gamma\boldsymbol{x}(k), \qquad (3.38)$$

тобто вектор синаптичних ваг «підтягується» до вхідного образу на відстань, пропорційний параметру кроку γ. Чим більше γ, тим ближче $\boldsymbol{w}_j(k+1)$ до $\boldsymbol{x}(k)$ і при $\gamma = 1$ збігається з ним. Зазвичай у реальних задачах використовується змінне значення $\gamma(k)$, обумовлене відповідно до умов *Дворецького* [115]. Можна також зазначити, що для зручності обчислювання замість вектора $\boldsymbol{x}(k)$ частіше використовують його нормований аналог

$$\tilde{\boldsymbol{x}}(k) = \frac{\boldsymbol{x}(k)}{\|\boldsymbol{x}(k)\|}.$$

3.1.7. Вихідна зірка

Своєрідним антиподом вхідної зірки є *вихідна зірка* (*Outstar*), призначена для розв'язання задач відновлення образів, а її схему зображено на рис. 3.11.

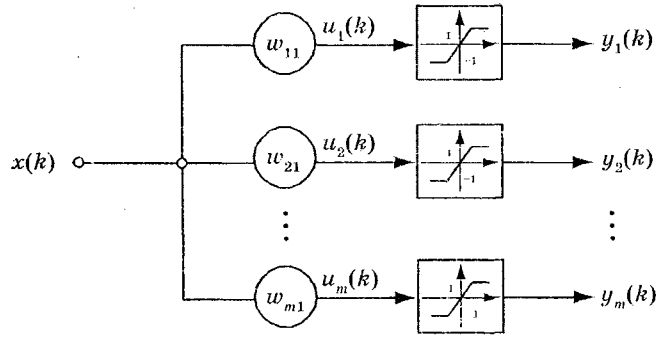


Рис. 3.11. Вихідна зірка

Цей нейрон має скалярний вхід і векторний вихід і здійснює перетворення

$$y_j = f(w_{j1}x), \; j = 1, 2, \ldots, m \qquad (3.39)$$

з функцією активації

$$f(u_j) = \begin{cases} u_j, & \text{якщо } -1 \le u_j \le 1, \\ 1, & \text{якщо } 1 < u_j, \\ -1, & \text{якщо } u_j < -1. \end{cases} \qquad (3.40)$$

Правило самонавчання вихідної зірки має вигляд

$$w_{ji}(k+1) = w_{ji}(k) + \gamma y_j(k)x_i(k) - \alpha x_i(k)w_{ji}(k), \qquad (3.41)$$

а при $\gamma = \alpha$ —

$$w_{ji}(k+1) = w_{ji}(k) + \gamma x_i(k)(y_j(k) - w_{ji}(k)), \qquad (3.42)$$

тобто настроювання синаптичних ваг відбувається тільки у випадку $x_i(k) \neq 0$. Тут, як видно, у процесі самонавчання синаптичні ваги «підтягуються» до вихідного вектора $y(k)$.

У векторній формі правило самонавчання має вигляд

$$\boldsymbol{w}_i(k+1) = \boldsymbol{w}_i(k) + \gamma \boldsymbol{x}_i(k)(\mathbf{y}(k) - \boldsymbol{w}_i(k)), \qquad (3.43)$$

де $\boldsymbol{w}_i(k)$ — i-й стовпець матриці коефіцієнтів $W(k+1)$.

3.2. Лінійна роздільність

Нехай N-вимірний простір X складається з двох підпросторів X_1 і X_2. Ці підпростори називаються *лінійно роздільними*, якщо є $(N+1)$-вимірний вектор $\boldsymbol{w}$, такий що

$$\sum_{i=1}^{N} w_i x_i = \begin{cases} \geq w_0 & \forall x = (x_1, \ldots, x_N) \in X_1; \\ < w_0 & \forall x = (x_1, \ldots, x_N) \in X_2. \end{cases} \qquad (3.44)$$

Якщо

$$\sum_i w_i x_i = \begin{cases} > w_0 & \forall x \in X_1; \\ < w_0 & \forall x \in X_2, \end{cases} \qquad (3.45)$$

то X_1 і X_2 називаються *абсолютно лінійно роздільними*.

Приклади розбивання на два класи наведено на рис. 3.12, 3.13.

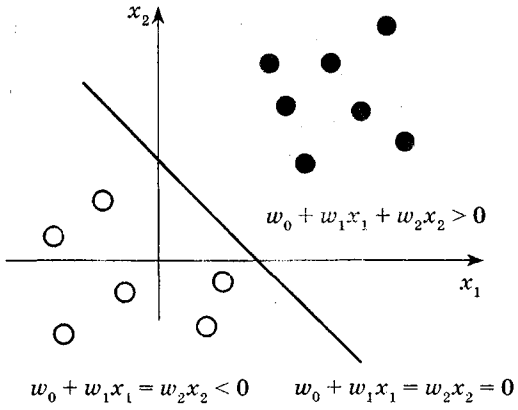


Рис. 3.12. Лінійно роздільні множини

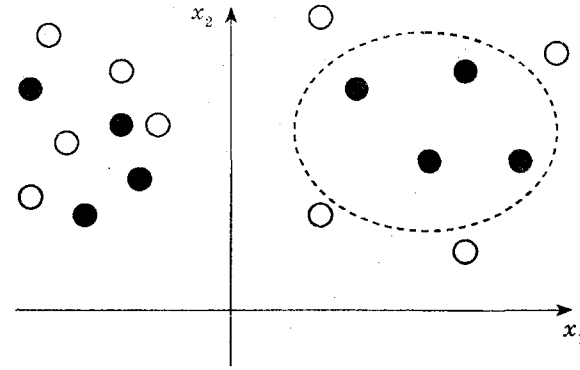


Рис. 3.13. Лінійно нероздільні множини

На рис. 3.14, 3.15 наведено графічну ілюстрацію реалізації найпростішим персептроном функцій «І» й «що виключає АБО». Темні кільця на рисунках відповідають значенню функції $f = 1$.

Як видно з рисунків, можна провести нескінченну безліч прямих, що відповідають різним значенням ваг w_1 й w_2, які відповідають нерівностям із прикладу 3.1 і реалізують функцію «І», і неможливо провести одну пряму, що розділяє площину X_1–X_2 так, щоб реалізовувалася функція «що виключає АБО» (приклад 3.2).

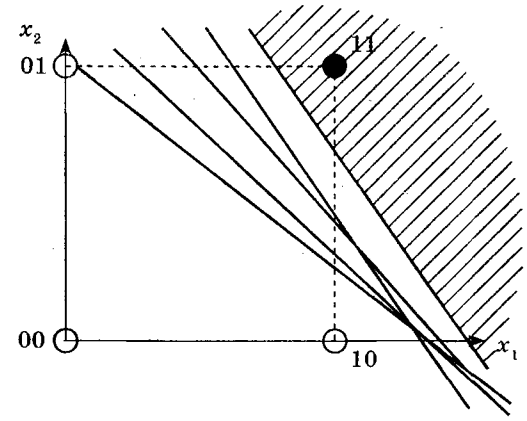


Рис. 3.14. Реалізація функції «І»

Слід зазначити, що вектор ваг $\boldsymbol{w}$ буде завжди ортогональним граничній площині.

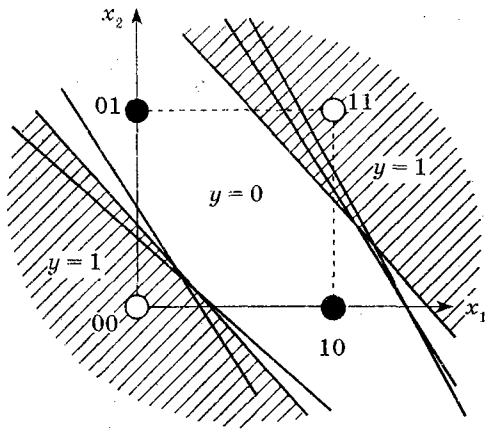


Рис. 3.15. Реалізація функції «що виключає АБО»

Приклад 3.5. Нехай є двовходовий персептрон (рис. 3.2) з вагами $w_1 = 0{,}5$, $w_2 = 1$ і зсувом $\theta = -1$. Тоді межа розв'язків є лінією, що описана рівнянням

$$0{,}5x_1 + 1x_2 - 1 = 0$$

і наведена на рис. 3.16.

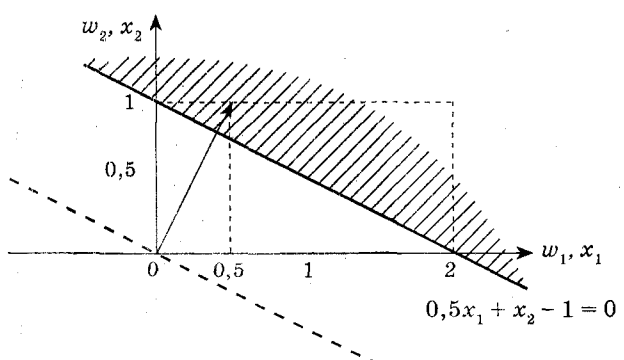


Рис. 3.16. Розташування межі розв'язків і вектора ваг

Нескладно перевірити, що область над межею відповідає $y = 1$, а під межею — $y = 0$.
Зобразивши на цьому ж графіку вектор $\boldsymbol{w} = (0{,}5 \;\; 1)^T$, неважко переконатися, що він буде ортогональний межі розв'язків.

Зазначимо, що це справедливо й для випадку $\theta = 0$ (пунктирна пряма на рис. 3.16).
Графічна ілюстрація дозволяє наочно уявити процес навчання персептрона.

Приклад 3.6. Нехай двовходовому персептрону з $\boldsymbol{w}_0 = (0{,}8 \; -0{,}6)^T$ (рис. 3.2) послідовно подаються такі пари, що навчають:
$(\boldsymbol{x}_1 = (1 \;\; 1{,}5)^T, y_1^* = 1)$, $(\boldsymbol{x}_2 = (-1 \;\; 2{,}5)^T, y_2^* = 0)$ $(\boldsymbol{x}_3 = (-0{,}5 \;\; -2)^T, y_3^* = 0)$.
На рис. 3.17 відповідні координати навчальних векторів позначені темними ($y^* = 1$) і світлими ($y^* = 0$) кільцями. Тут також наведено вихідний вектор ваг $\boldsymbol{w} = (0{,}8 \;\; -0{,}6)^T$.

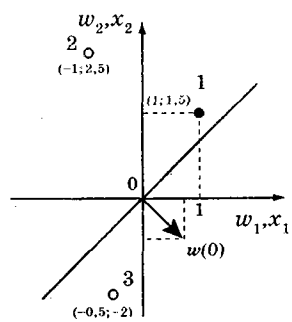


Рис. 3.17. Розташування початкового вектора ваг і векторів, що навчають

Подання вхідного вектора $\boldsymbol{x}_1 = (1 \;\; 1{,}5)^T$ викликає на виході персептрона сигнал

$$y_1 = f_a\left(\boldsymbol{w}_0^T \boldsymbol{x}_1\right) = f_a\left((0{,}8 \;\; -0{,}6)\begin{pmatrix}1\\1{,}5\end{pmatrix}\right) = f_a(-0{,}1) = 0,$$

тобто даний образ класифікований неправильно.
Дійсно, з рис. 3.17 видно, що перший і третій образи розташовано по одну сторону від межі розв'язків.
Корекція ваг за алгоритмом (3.2) з $\gamma = 1$ дає

$$\boldsymbol{w}_1 = \boldsymbol{w}_0 + \boldsymbol{x}_1 = \begin{pmatrix}0{,}8\\-0{,}6\end{pmatrix} + \begin{pmatrix}1\\1{,}5\end{pmatrix} = \begin{pmatrix}1{,}8\\0{,}9\end{pmatrix}.$$

Після подання $\boldsymbol{x}_2 = (-1 \;\; 2{,}5)^T$ отримуємо

$$y_2 = f_a\left(\boldsymbol{w}_1^T \boldsymbol{x}_2\right) = f_a\left((1{,}8 \;\; 0{,}9)\begin{pmatrix}-1\\2{,}5\end{pmatrix}\right) = f_a(0{,}445) = 1,$$

а оскільки класифікація здійснена неправильно, знову коректуємо ваги

$$\boldsymbol{w}_2 = \boldsymbol{w}_1 - \boldsymbol{x}_2 = \begin{pmatrix}1{,}8\\0{,}9\end{pmatrix} - \begin{pmatrix}-1\\2{,}5\end{pmatrix} = \begin{pmatrix}2{,}8\\-1{,}6\end{pmatrix}.$$

Подання $\boldsymbol{x}_3 = (-0{,}5 \;\; -2)^T$ дає

$$y_3 = f_a\left((2{,}8 \;\; -1{,}6)\begin{pmatrix}-0{,}5\\-2\end{pmatrix}\right) = f_a(0{,}8) = 1,$$

тому ваги коректуємо знову

$$\boldsymbol{w}_3 = \boldsymbol{w}_2 - \boldsymbol{x}_3 = \begin{pmatrix}2{,}8\\-1{,}6\end{pmatrix} - \begin{pmatrix}-0{,}5\\-2,\end{pmatrix} = \begin{pmatrix}3{,}3\\0{,}4\end{pmatrix}.$$

На рис. 3.18 показано послідовні розташування вектора коефіцієнтів після кожного подання персептрону пар, що навчають. Як видно з рисунка, після подання $(\boldsymbol{x}_3, y_3^*)$ персептрон остаточно навчився, тобто налаштував вагові коефіцієнти так, щоб подані йому вектори класифікувалися правильно. Про це свідчить і положення межі розв'язків — прямої, перпендикулярної w_3 і тієї, що ділить всю площину на два класи.

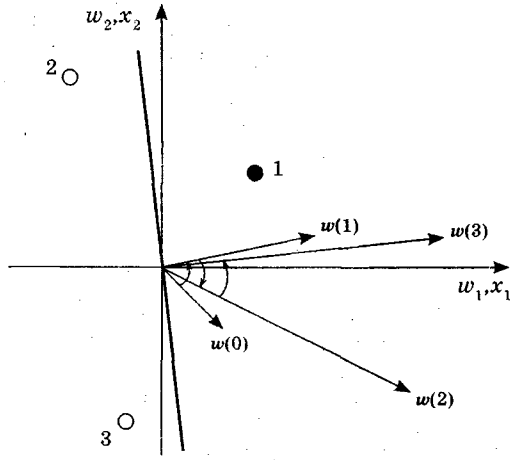


Рис. 3.18. Послідовна зміна розташування вектора ваг

Таким чином, як найпростіший (одношаровий) персептрон, так і Адаліна можуть класифікувати лише лінійно роздільні множини. На жаль, приклад, наведений на рис. 3.15, не єдиний, що свідчить про обмежену можливість одношарових мереж. Тому практичне застосування персептронних схем вимагає впевненості в тому, що розглянута функція є лінійно роздільною. Для загального випадку проблема лінійної роздільності не вирішена.

Р. Віднер досліджував цю проблему теоретично й визначив число лінійно роздільних функцій для n нейронів, входами яких є двійкові змінні. Оскільки в цьому випадку загальне число комбінацій вхідних сигналів становить 2^n, то загальне число вихідних комбінацій (булевих функцій) дорівнюватиме 2^{2^n} [34].

Таблиця 3.2. Лінійно роздільні функції

n	Число булевих функцій	Число лінійно роздільних функцій
1	4	4
2	16	14
3	256	104
4	65536	1772
5	$4{,}3 \cdot 10^9$	94572
6	$1{,}8 \cdot 10^{19}$	5028134

Як видно з таблиці, імовірність того, що випадково обрана функція виявиться лінійно роздільною, досить мала навіть для невеликого числа змінних.

Ефективність схем персептронного типу значно підвищується шляхом переходу до багатошарових персептронів.

3.3. Багатошарові ШНМ

3.3.1. Багатошаровий персептрон

Загальний вигляд багатошарового персептрона наведено на рис. 3.19.

Даний персептрон, що має N входів, M виходів й K шарів, з яких перший, що містить S нейронів, є вхідним, k-й, що складається з M нейронів, — вихідним, а інші — прихованими, описується рівнянням

$$y = f^{(k)}\left[w^{(k)}\ldots f^{(2)}\left[w^{(2)}f^{(1)}\left[w^{(1)}x + b^{(1)}\right] + b^{(2)}\right]\ldots + b^{(k)}\right],$$

де $w^{(i)}$ — матриця ваг i-го шару; $b^{(i)}$ — вектор зсувів i-го шару.

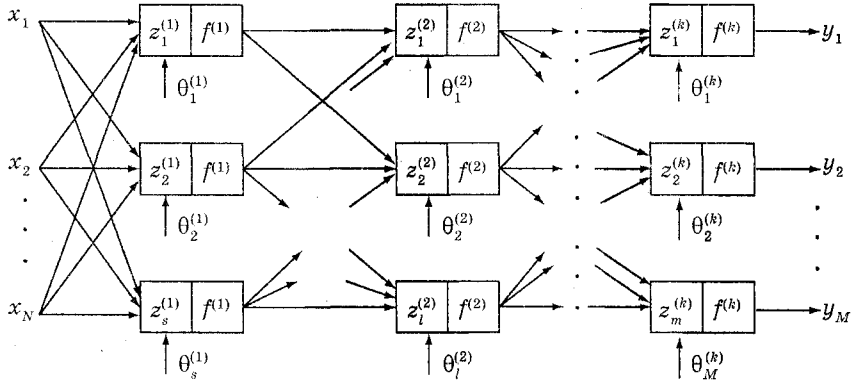


Рис. 3.19. Багатошаровий персептрон

За допомогою багатошарових персептронов можна вирішувати значно складніші задачі, зокрема можна реалізувати нерозв'язну за допомогою одношарового персептрона функцію «що виключає АБО».

Приклад 3.7. Одну з можливих реалізацій функції «що виключає АБО» наведено на рис. 3.20.

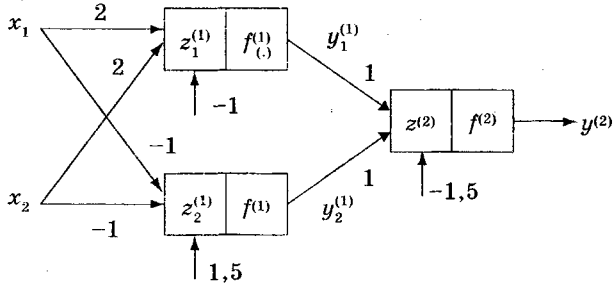


Рис. 3.20. Реалізація функції «що виключає АБО»

На рисунку позначено: $z_i^{(j)}$ — зважена сума входів i-го нейрона j-го шару; $y_i^{(j)}$ — вихідний сигнал i-го нейрона j-го шару; f — порогова активаційна функція.

Функціонування цієї мережі описується в табл. 3.3.

Таблиця 3.3

x_1	x_2	$y_1^{(1)}$	$y_2^{(1)}$	$y^{(2)}$
0	0	$2\cdot0+2\cdot0-1=0$	$(-1)\cdot0+(-1)\cdot0+1{,}5=1$	$1\cdot0+1\cdot1-1{,}5=0$
0	1	$2\cdot0+2\cdot1-1=1$	$(-1)\cdot0+(-1)\cdot1+1{,}5=1$	$1\cdot1+1\cdot1-1{,}5=1$
1	0	$2\cdot1+2\cdot0-1=1$	$(-1)\cdot1+(-1)\cdot0+1{,}5=1$	$1\cdot1+1\cdot1-1{,}5=1$
1	1	$2\cdot1+2\cdot1-1=1$	$(-1)\cdot1+(-1)\cdot1+1{,}5=0$	$1\cdot1+1\cdot0-1{,}5=0$

Кожний з нейронів першого шару здійснює лінійне розбивання на два класи (рис. 3.21, *а* й *б*), а нейрон другого шару реалізує функцію «І» (рис. 3.21, *в*).

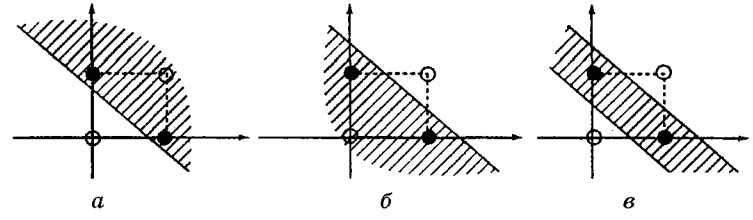


Рис. 3.21. Послідовний поділ на два класи

Цей приклад свідчить про те, що використання більшої кількості шарів дозволяє отримувати області розв'язків у вигляді багатокутників кожної наперед заданої форми.

Ці області завжди будуть опуклими, оскільки вони утворені з використанням операції «І» над областями, що задаються лініями.

Неопуклі області розв'язків можуть бути реалізовані на тришаровій мережі, коли опуклі багатокутники, що надходять на входи нейрона третього шару шляхом певних логічних комбінацій, здійснюваних цим нейроном, утворюють неопуклий багатокутник. Слід зазначити, що зі збільшенням кількості нейронів кількість сторін багатокутника необмежено зростає. Це дозволяє апроксимувати області будь-якої форми з необхідною точністю. На рис. 3.22 показано можливості персептронів різної структури [82].

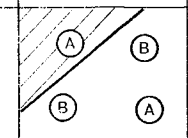
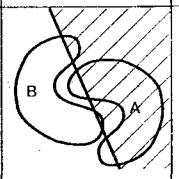
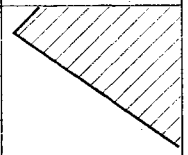
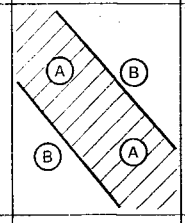
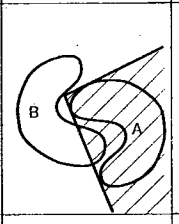
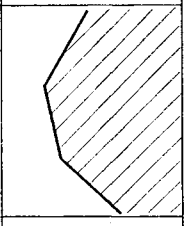
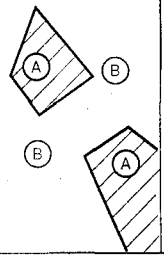
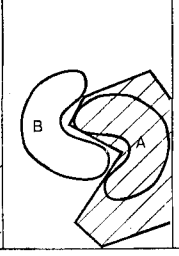
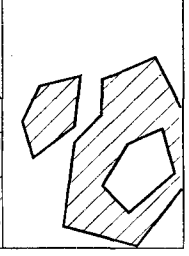
Структура персептрона	Область розв'язання	«Що виключає АБО»	Вільно розташовані області	Області найбільш загальної форми
Одношарова	Напівплощина, обмежена гіперплощиною			
Двошарова	Опуклі відкриті або закриті області			
Тришарова	Будь-якої складності (обмежені числом вузлів)			

Рис. 3.22. Можливості персептронів

Приклад 3.8. Розглянемо задачу стиснення інформації (*Encoding-Problem*). На вхід мережі подається n-розрядна двійкова послідовність, що містить тільки одну 1. Вихід багатошарового персептрона має повторити вхідний сигнал. При n нейронах вхідного й вихідного шарів прихований шар містить $(\log_2 n)$ нейронів. Таким чином, мережа має навчитися кодувати n-розрядний вхідний сигнал у $(\log_2 n)$-розрядний, а потім його знову декодувати.

На рис. 3.23 наведено тришаровий персептрон, що вирішує проблему 8-3-8.

У таблиці 3.4 наведено значення ваг (прихованого шару), отримані за допомогою градієнтного алгоритму навчання після 3150 поданих образів [56].

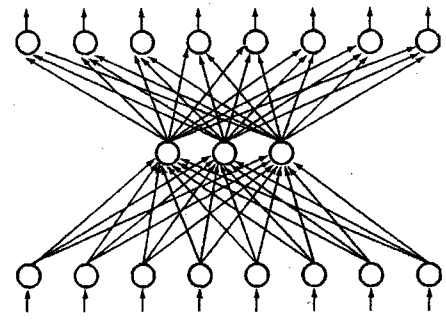


Рис. 3.23. Персептрон, що вирішує проблему 8-3-8

Таблиця 3.4

Вхідний образ	Прихований шар	Вихідний образ
10000000	0,0 1,0 0,6	10000000
01000000	1,0 1,0 1,0	01000000
00100000	0,2 0,6 0,0	00100000
00010000	1,0 1,0 0,2	00010000
00001000	1,0 0,0 0,1	00001000
00000100	0,0 0,0 0,3	00000100
00000010	1,0 0,0 1,0	00000010
00000001	0,0 0,3 1,0	00000001

На рис. 3.24 наведено графік залежності помилки мережі від кількості циклів навчання. Після завершення навчання помилка становила 8 %.

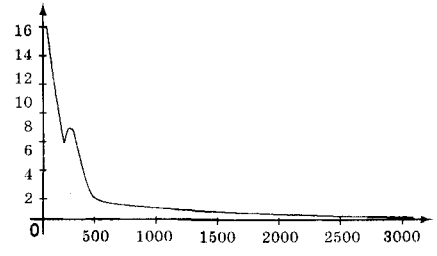


Рис. 3.24. Графік зміни помилки

3.3.2. Мадаліна

Мадаліна (*Many ADALINES*) є в загальному випадку багатошаровою мережею, утвореною багатьма Адалінами (рис. 3.25).

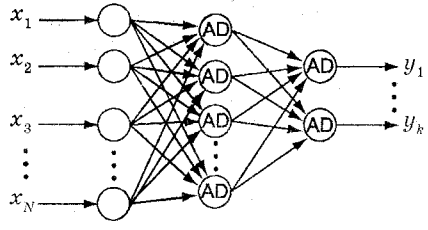


Рис. 3.25. Мадаліна

Цей тип мереж був покликаний усунути недоліки, властиві як Адаліні, так й одношаровому персептрону, і вирішувати задачі, які не могли бути вирішені з їхньою допомогою, зокрема здійснювати класифікацію лінійно нероздільних класів, вирішувати проблему «що виключає АБО» і т. д.

Приклад 3.9. Мадаліну, яка реалізує функцію «що виключає АБО», наведено на рис. 3.26.

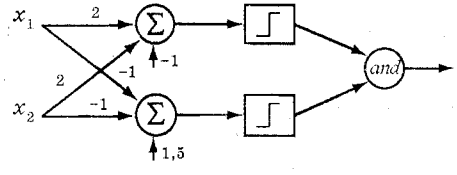


Рис. 3.26. Реалізація логічної функції «що виключає АБО» на двох Адалінах

На вхід мережі подаються сигнали x_1 й x_2, що приймають значення +1 й −1. Кожна з Адалін, показаних на рисунку, виробляє вихідний сигнал, що являє собою біполярне перетворення зважених сум вхідних сигналів. Ці вихідні сигнали надходять на вхід елемента «І», на вході якого, як нескладно переконатися, з'являється сигнал «+1», якщо значення вхідних сигналів x_1 й x_2 не збігаються, і «0», коли збігаються, тобто реалізується логічна функція «що виключає АБО».

Якщо первинні схеми Мадаліни використовували настроювання вагових параметрів Адалін першого шару й порогові (релейні) функції активації, то згодом були розроблені алгоритми настроювання ваг усіх шарів Мадаліни, що реалізують навчання з учителем, і використані неперервні нелінійні, зокрема, сигмоїдальні функції активації.

Адаптивний алгоритм навчання Мадаліни заснований на принципі, названому Уідроу «*принципом мінімального збурювання*», і полягає в тому, що корекція вагових коефіцієнтів при поданні нового образу, що забезпечує зменшення виникаючої вихідної помилки мережі, має бути мінімальною. Це пов'язано з тим, що наявні до моменту подання нового образу значення коефіцієнтів, обчислені на основі минулих образів, правильно відбивали роботу мережі. Робота алгоритму полягає в наступному.

При поданні нового образу у вхідному шарі вибирається нейрон, значення вихідного сигналу якого близьке до нуля, і змінюються його вагові коефіцієнти таким чином, щоб його бінарний стан змінився на протилежний. Далі простежується, як впливає ця зміна на стан нейронів наступних шарів і чи зменшується при цьому вихідна помилка: якщо зменшується, то внесені зміни ваг залишаються, якщо ні — значення коефіцієнтів залишаються попередніми й шукається інший нейрон вхідного шару. Цей процес триває доти, поки не будуть переглянуті всі нейрони першого шару. Після цього вибираються різні пари нейронів першого шару, для яких процедура зміни ваг повторюється й визначається вплив цієї зміни на вихідну помилку. Якщо помилка зменшується, зміни залишають, якщо ні — значення вагових коефіцієнтів пар залишають попередніми. Після аналізу всіх пар переходять до аналізу різних трійок, однак на практиці обмежуються звичайно парами.

Після цього переходять до нейронів другого шару й повторюють всю процедуру корекції ваг з урахуванням зміни ваг нейронів першого шару. Потім розглядають третій шар і т. д. аж до останнього (вихідного), вагові коефіцієнти якого корегують інакше, а саме з використанням дельта-правила або будь-якого іншого алгоритму навчання з учителем. Це можливо внаслідок того, що виявляються помилки кожної окремої Адаліни, що перебуває у вихідному шарі. Весь процес корекції вагових коефіцієнтів повторюється для кожного знову поданого образу.

Розглянутий адаптивний алгоритм навчання Мадаліни був згодом модифікований й удосконалений, і сьогодні існує багато його різних варіантів, які використовують, зокрема, крім порогової активаційної функції й сигмоїдальні, що надає мережі додаткові

позитивні властивості, а сам алгоритм стає практично еквівалентним найбільш відомому й досконалому *алгоритму зворотного поширення помилки*, який розглядатиметься нижче.

3.3.3. ШНМ, що заснована на МГУА

За аналогією з Мадаліною, елементами якої є Адаліни, може бути побудована мережа, що складається з *Н-Адалін*.

В алгоритмах МГУА аргументи й проміжні змінні поєднуються попарно, тобто в групи, що складаються із двох аргументів (тому МГУА, очевидно, більш точно може бути названий методом попарного урахування аргументів). У такий спосіб мережа, що складається з m шарів, може реалізовувати поліном степеня 2^m.

Дана мережа, запропонована в [46], називається *мережею, заснованою на МГУА* (*GMDH Neural Network*) (рис. 3.27).

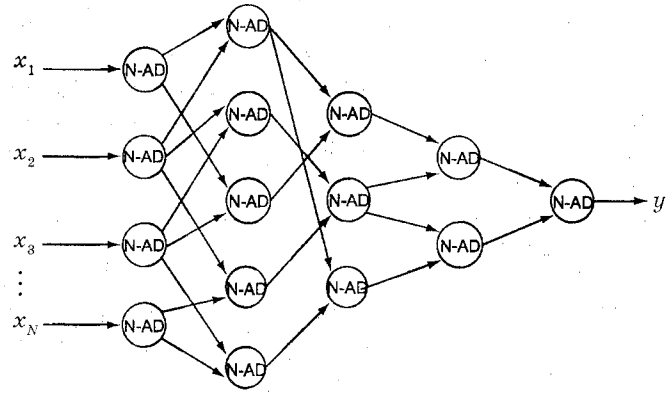


Рис. 3.27. ШНМ, заснована на МГУА

Основним критерієм оптимізації в МГУА є *критерій мінімуму середньоквадратичної помилки* (СКП), а питання, пов'язані зі знаходженням цього мінімуму, вирішуються за допомогою перебирання варіантів, що використовують навчальні й перевіркові послідовності даних. Для вибору кращих варіантів використовується пороговий принцип.

Критерій мінімуму СКП використовується в алгоритмах МГУА принаймні в чотирьох випадках:

1) для обчислення вагових коефіцієнтів кожної Н-Адаліни;

2) для выбору кращих комбінацій пар аргументів, тобто входів Н-Адалін;

3) для визначення величини порогів, на основі яких відбувається відбір кращих варіантів;

4) для визначення ступеня нелінійності використовуваного опису й вибору функцій, що його утворюють.

Якщо у звичайних методах, як правило, з умови мінімуму СКП знаходять вагові коефіцієнти *повного* рівняння, що описує об'єкт, то МГУА забезпечує такий вибір вагових коефіцієнтів *окремих* рівнянь, що становлять повне, при якому досягається мінімум СКП у просторі цих коефіцієнтів. Коефіцієнти ж повного рівняння визначаються за допомогою виключення проміжних змінних із окремих рівнянь. Крім того, для зменшення СКП у МГУА здійснюється повний або частковий перебір можливих комбінацій пар аргументів (вхідних сигналів Н-Адалін), виконуваний також за критерієм мінімуму СКП. У результаті вибираються комбінації, для яких помилка мінімальна.

Кількість враховуваних пар, змінних у кожному шарі багатошарової мережі визначається порогами, що обчислюють також за мінімумом СКП.

Повний опис МГУА, у тому числі поділ даних на послідовності, що навчають і перевіряють, оптимізацію порогів, правила селекції змінних і т. д., наведено в монографії [81]. Тут зазначимо лише, що оскільки ця мережа призначена для моделювання й прогнозування складних процесів, вхідні й вихідні сигнали яких є випадковими, перед початком навчання мережі здійснюється стандартизація (центрування й нормування) змінних за формулами

$$\hat{x}_i = \frac{x_i - \overline{x}_i}{\sigma_{x_i}};\ \ \hat{y}_i = \frac{y_i - \overline{y}_i}{\sigma_{y_i}}, \qquad (3.46)$$

де x_i, y_i, $\overline{x}_i$, $\overline{y}_i$ — вхідні й вихідні змінні та їхні середні значення відповідно; σ_{x_i}, σ_{y_i} — середньоквадратичні відхилення x_i й y_i.

Потім формуються різні пари вхідних сигналів Н-Адалін першого шару та здійснюється корекція їхніх ваг за правилом Уідроу — Гоффа.

Після досягнення мінімального значення сумарної СКП першого шару корекція ваг цього шару завершується, тобто вагам привласнюються відповідні значення, які надалі не змінюються. Процедура навчання (селекція сигналів, корекція тощо) повторюється для другого шару й т. д. до досягнення всією мережею необхідного або припустимого значення СКП.

3.4. Алгоритм зворотного поширення помилки

Алгоритм зворотного поширення помилки, згодом названий просто *алгоритмом зворотного поширення* (*Backpropagation, BP*), що являє собою розширене дельта-правило, був запропонований у роботі [17] і застосований для навчання ШНМ у роботах [18, 19]. Він реалізує градієнтний метод мінімізації опуклого (звичайного квадратичного) функціонала помилки в багатошарових мережах прямого поширення, що використовують моделі нейронів з диференціальними функціями активації. Застосування сигмоїдальних функцій активації, що є монотонно зростаючими і що мають відмінні від нуля похідні на всій області визначення, забезпечує правильне навчання й функціонування мережі. Процес навчання полягає у послідовному поданні мережі пар $(\boldsymbol{x}(i), \boldsymbol{y}^*(i))$ $i=\overline{1,P}$, що навчають, де $\boldsymbol{x}(i)$ і $\boldsymbol{y}^*(i)$ — вектор вхідних і бажаних вихідних сигналів мережі відповідно, вивченні реакції на них мережі й корекції відповідно до реакції вагових параметрів (елементів вагової матриці).

Перед початком навчання всім вагам привласнюються невеликі різні випадкові значення (якщо задати всі значення однакові, а для правильного функціонування мережі знадобляться нерівні значення, мережа не навчатиметься).

Для реалізації алгоритму зворотного поширення необхідно:

1. Вибрати із заданої навчальної множини чергову пару $(\boldsymbol{x}(i), \boldsymbol{y}^*(i))$, $i=\overline{1,P}$, що навчає, і подати на вхід мережі вхідний сигнал $\boldsymbol{x}(i)$.
2. Обчислити реакцію мережі $\boldsymbol{y}(i)$.
3. Порівняти отриману реакцію $\boldsymbol{y}(i)$ з необхідною $\boldsymbol{y}^*(i)$ і визначити помилку $\boldsymbol{y}^*(i) - \boldsymbol{y}(i)$.
4. Скорегувати ваги так, щоб помилка була мінімальною.
5. Кроки 1–4 повторити для всієї множини пар $(\boldsymbol{x}(i), \boldsymbol{y}^*(i))$ $i=\overline{1,P}$, що навчають, доти, поки на заданій множині помилка не досягне необхідної величини.

Таким чином, у процесі навчання мережі подача вхідного сигналу й обчислення реакції відповідає *прямому* проходу сигналу від вхідного шару до вихідного, а обчислення помилки й корекція вихідних параметрів — *зворотному*, коли сигнал помилки поширюється по мережі від її виходу до входу. При зворотному проході здійснюється пошарова корекція ваг, починаючи з вихідного шару. Якщо корекція ваг вихідного шару здійснюється за допомогою модифікованого дельта-правила порівняно просто, оскільки

необхідні значення вихідних сигналів відомі, то корекція ваг прихованих шарів відбувається трохи складніше, оскільки для них не відомі необхідні вихідні сигнали.

Алгоритм зворотного поширення застосовують також щодо мереж з будь-якою кількістю шарів: як мереж прямого поширення, так і таких, що містять зворотні зв'язки. Ми обмежимося розглядом випадку навчання двох шарів мережі прямого поширення. Фрагмент такої мережі зображено на рис. 3.28.

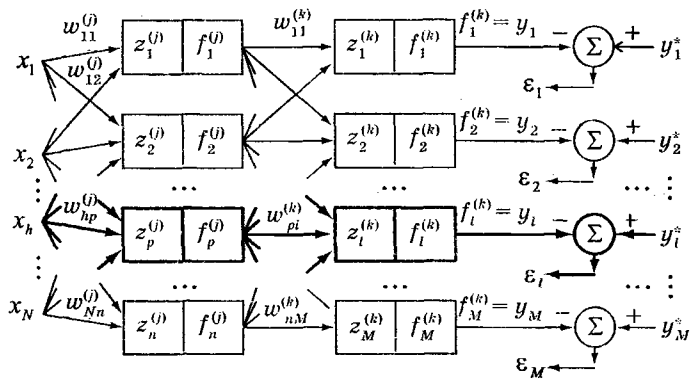


Рис. 3.28. Фрагмент мережі прямого поширення

На рисунку позначено: $w_{pi}^{(k)}$ — вага зв'язку p-го нейрона попереднього шару (j-го) з l-м нейроном наступного (k-го) шару; $f_p^{(j)}$ — активаційна функція p-го нейрона i-го шару; $z_p^{(j)}$ — зважена сума вихідних сигналів попереднього (i-го) шару, що надходять на вхід p-го нейрона наступного (j-го) шару.

3.4.1. Обчислення ваг нейронів вихідного шару

На рис. 3.29 використовуються ті позначення, що й на рис. 3.28. Розглянемо l-й нейрон вихідного (k-го) шару. Вихід даного нейрона є виходом мережі, тому його сигнал $y_l = f_l^{(k)}$ порівнюється з необхідним y_l^* й обчислюється помилка

$$\zeta_l = y_l^* - f_l^{(k)}. \qquad (3.47)$$

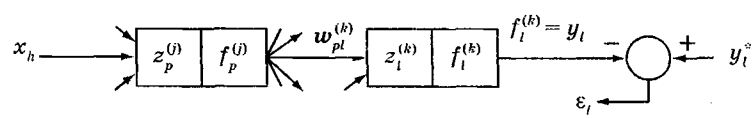


Рис. 3.29. Фрагмент багатошарової мережі

Використання квадратичного критерію якості навчання

$$\zeta_l^2 = (y_l^* - f_l^{(k)})^2 \tag{3.48}$$

дозволяє отримати градієнтний алгоритм корекції ваг, що у цьому випадку набуває вигляду

$$\Delta w_{pl}^{(k)} = -\gamma_{pl} \frac{\partial \zeta_l^2}{\partial w_{pl}^{(k)}}, \tag{3.49}$$

де γ_{pl} — коефіцієнт, що впливає на швидкість навчання.

Обчислення складової (3.49) частинної похідної здійснюється за правилом

$$\frac{\partial \zeta_l^2}{\partial w_{pl}^{(k)}} = \frac{\partial \zeta_l^2}{\partial f_l^{(k)}} \frac{\partial f_l^{(k)}}{\partial z_l^{(k)}} \frac{\partial z_l^{(k)}}{\partial w_{pl}^{(k)}}. \tag{3.50}$$

З урахуванням (3.48), виду активаційних функцій і того, що

$$z_l^{(k)} = \sum_{p=1}^{n} w_{pl}^{(k)} f_p^{(j)},$$

отримуємо

$$\frac{\partial \zeta_l^2}{\partial f_l^{(k)}} = -2(y_l^* - y_l) = -2\zeta_l; \tag{3.51}$$

$$\frac{\partial f_l^{(k)}}{\partial Z_l^{(k)}} = \alpha f_l^{(k)}(1 - f_l^{(k)}); \tag{3.52}$$

$$\frac{\partial z_l^{(k)}}{\partial w_{pl}^{(k)}} = f_p^{(k)}. \tag{3.53}$$

Підстановка (3.51)–(3.53) у (3.50) дає

$$\Delta w_{pl}^{(k)} = 2\alpha\gamma_{pl}\zeta_l f_l^{(k)}(1 - f_l^{(k)}) f_p^{(k)} \tag{3.54}$$

або

$$w_{pl}^{(k)}(i+1) = w_{pl}^{(k)}(i) + \gamma_{pl}\delta_{pl}^{(k)} f_p^{(k)}, \tag{3.55}$$

де

$$\delta_{pl}^{(k)} = 2\alpha\zeta_l f_l^{(k)}(1 - f_l^{(k)}), \tag{3.56}$$

i — номер ітерації.

Аналогічно обчислюються ваги інших нейронів вихідного шару. Всі величини, що входять до складу алгоритму (3.55), є відомими, тому його реалізація труднощів не викликає. Присутня в (3.54), (3.55) помилка ζ_l відмінна від нуля внаслідок того, що нейрони вихідного шару виробляють помилкові вихідні сигнали,

тому що, по-перше, їм привласнені випадкові вагові коефіцієнти й, по-друге, нейрони прихованих шарів також виробляють помилкові сигнали.

Як ми вже зазначали, наявна помилка ζ_l поширюється назад на попередні приховані шари й використовується для корекції ваг цих шарів.

3.4.2. Обчислення ваг нейронів прихованого шару

Фрагмент мережі для обчислення ваг нейронів прихованого шару наведено на рис. 3.30.

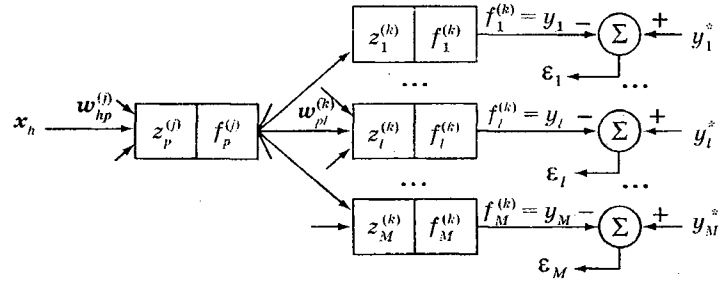


Рис. 3.30. Фрагмент мережі

Розглянемо p-й нейрон прихованого (j-го) шару. Для обчислення ваги $w_{hp}^{(j)}$ даного нейрона також використовується дельта-правило, аналогічне (3.48). Але оскільки вихідний сигнал цього нейрона надходить на виходи всіх нейронів вхідного шару, алгоритм настроювання записується в такий спосіб:

$$\Delta w_{hp}^{(j)} = -\gamma_{hp} \frac{\partial \zeta^2}{\partial w_{hp}^{(j)}} = -\gamma_{hp} \sum_{i=1}^{M} \frac{\partial \zeta_l^{(2)}}{\partial w_{hp}^{(j)}}. \tag{3.57}$$

Тут ζ_l — помилка на i-му виході; γ_{hp} — параметр, що виконує ту саму роль, що й γ_{pl} у (3.55).

Оскільки квадратичний критерій якості в цьому випадку приймає вигляд

$$\zeta = \sum_{l=1}^{M} \left[y_l^* - f_l^{(k)} \right]^2, \tag{3.58}$$

частинна похідна, використовувана в (3.57), обчислюється так:

$$\frac{\partial \zeta^2}{\partial w_{hp}^{(j)}} = \sum_{l=1}^{M} \frac{\partial \zeta_l^2}{\partial f_l^{(k)}} \frac{\partial f_l^{(k)}}{\partial z_l^{(k)}} \frac{\partial z_l^{(k)}}{\partial f_p^{(j)}} \frac{\partial f_p^{(j)}}{\partial z_p^{(j)}} \frac{\partial z_p^{(j)}}{\partial w_{hp}^{(j)}}. \tag{3.59}$$

Для розглянутого випадку за аналогією з вищевикладеним отримуємо

$$\frac{\partial \zeta_l^2}{\partial f_l^{(k)}} = -2[y_l^* - f_l^{(k)}] = -2\zeta_l; \tag{3.60}$$

$$\frac{\partial f_i^{(k)}}{\partial z_i^{(k)}} = \alpha f_i^{(k)}(1 - f_i^{(k)}); \tag{3.61}$$

$$\frac{\partial z_l^{(k)}}{\partial f_p^j} = w_{pl}^{(k)}; \tag{3.62}$$

$$\frac{\partial f_p^{(j)}}{\partial z_p^{(j)}} = \alpha f_p^{(j)}(1 - f_p^{(j)}); \tag{3.63}$$

$$\frac{\partial z_p^{(j)}}{\partial w_{hp}^{(j)}} = x_h. \tag{3.64}$$

Тут враховано, що

$$z_l^{(k)} = \sum_{p=1}^{m} w_{pl}^{(k)} f_p^{(j)};$$

$$z_l^{(j)} = \sum_{h=1}^{N} w_{hp}^{(j)} x_h,$$

а функції активації є сигмоїдальними.

Таким чином,

$$\frac{\partial \zeta^2}{\partial w_{hp}^{(j)}} = \sum_{l=1}^{M}(-2)\alpha\zeta_l\left[f_l^{(k)}(1 - f_l^{(k)})\right] w_{pl}^{(k)} \alpha\left[f_p^{(j)}(1 - f_p^{(j)})\right] x_h =$$

$$= -\sum_{l=1}^{M} \delta_{pl}^{(k)} w_{pl}^{(k)} \frac{\partial f_p^{(j)}}{\partial z_p^{(j)}} x_h, \tag{3.65}$$

де δ_{pl}^{k} визначається виразом (3.56).

Позначимо

$$\delta_{hp}^{(j)} = \delta_{pl}^{(k)} w_{pl}^{(k)} \frac{\partial f_p^{(j)}}{\partial z_p^{(j)}}. \tag{3.66}$$

Тоді

$$\frac{\partial \zeta^2}{\partial w_{hp}^{(j)}} = -\sum_{l=1}^{M} \delta_{hp}^{(j)} x_h \tag{3.67}$$

і алгоритм корекції ваг приймає вигляд

$$\Delta w_{hp}^{(j)} = -\gamma_{hp} x_h \sum_{l=1}^{M} \delta_{hp}^{(j)} \tag{3.68}$$

або

$$w_{hp}^{(j)}(i+1) = w_{hp}^{(j)}(i) + \gamma_{hp} x_h \sum_{l=1}^{M} \delta_{hp}^{(j)}. \tag{3.69}$$

Приклад 3.10. Розглянемо корекцію вагових коефіцієнтів мережі прямого поширення, зображеної на рис. 3.31, за допомогою алгоритму зворотного поширення помилки. Для простоти приймемо, що всі нейрони мають однакову функцію активації з $\alpha = 1$, а коефіцієнт навчання, використовуваний в алгоритмах настроювання ваг, $\gamma = 0{,}5$. Бажаний вихід нейрона E $y^* = 0{,}2$.

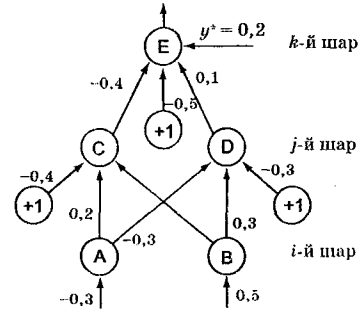


Рис. 3.31. Приклад мережі

Ваги між шарами j й k корегуються за правилом (3.55), а між шарами i й j — за правилом (3.69).

Спочатку обчислимо виходи кожного нейрона з огляду на вигляд функції активації $f = (1 + e^{-\alpha x})^{-1}$ при $\alpha = 1$:

$z_C = -0{,}3 \cdot 0{,}2 + 0{,}5 \cdot 0{,}1 + 1 \cdot (-0{,}4) = -0{,}06 + 0{,}05 - 0{,}4 = -0{,}41;$

$y_C = f_C = 0{,}399$

$z_D = -0{,}3 \cdot (-0{,}3) + 0{,}5 \cdot 0{,}3 + 1 \cdot (-0{,}3) = 0{,}09 + 0{,}15 - 0{,}3 = -0{,}06;$

$y_D = f_D = 0{,}485$

$z_E = 0{,}399 \cdot (-0{,}4) + 0{,}485 \cdot 0{,}1 + 1 \cdot (-0{,}5) = -0{,}16 + 0{,}485 - 0{,}5 = -0{,}17;\ y_E = f_E = 0{,}352$

Підстановка цих й інших значень у вираз (3.54) і (3.68) дає

$\Delta w_{CE} = -0{,}5 \cdot (-2) \cdot 1 \cdot (0{,}2 - 0{,}352) \cdot 0{,}352 \cdot (1 - 0{,}352) \cdot 0{,}399 = -0{,}01381;$

$\Delta w_{DE} = (0{,}2 - 0{,}352) \cdot 0{,}352 \cdot (1 - 0{,}352) \cdot 0{,}485 = -0{,}01679;$

$\Delta w_{1E} = (0{,}2 - 0{,}352) \cdot 0{,}352 \cdot (1 - 0{,}352) \cdot 1 = -0{,}03462;$

$\Delta w_{AC} = (0{,}2 - 0{,}352) \cdot 0{,}352 \cdot (1 - 0{,}352) \cdot (-0{,}4) \cdot 1 \cdot 0{,}399 \cdot (1 - 0{,}399) \cdot (-0{,}3) = -0{,}001;$

$\Delta w_{AD} = (0{,}2 - 0{,}352) \cdot 0{,}352 \cdot (1 - 0{,}352) \cdot 0{,}1 \cdot 1 \cdot 0{,}485 \cdot (1 - 0{,}485) \cdot (-0{,}3) = 0{,}00026;$

$$\Delta w_{SC} = (0,2 - 0,352) \cdot 0,352 \cdot (1 - 0,352) \cdot (-0,4) \cdot 1 \cdot 0,399 \cdot (1 - 0,399) \cdot 0,5 = -0,00166;$$

$$\Delta w_{BD} = (0,2 - 0,352) \cdot 0,352 \cdot (1 - 0,352) \cdot 0,1 \cdot 1 \cdot 0,485 \cdot (1 - 0,485) \cdot 0,5 = -0,00043;$$

$$\Delta w_{1C} = (0,2 - 0,352) \cdot 0,352 \cdot (1 - 0,352) \cdot (-0,4) \cdot 1 \cdot 0,399 \cdot (1 - 0,399) \cdot 1 = 0,00332;$$

$$\Delta w_{AD} = (0,2 - 0,352) \cdot 0,352 \cdot (1 - 0,352) \cdot 0,1 \cdot 1 \cdot 0,485 \cdot (1 - 0,485) \cdot 1 = -0,00086.$$

Додавши ці прирости до існуючих ваг, тобто скориставшись формулами (3.55) і (3.68), отримуємо такі нові значення ваг:

$$\Delta w_{CE} = -0,4 - 0,01381 = -0,414;$$

$$\Delta w_{DE} = 0,1 - 0,01679 = 0,083;$$

$$\Delta w_{1E} = -0,5 - 0,03462 = -0,535;$$

$$\Delta w_{AC} = 0,2 - 0,001 = 0,199;$$

$$\Delta w_{AD} = -0,3 + 0,00026 = -0,03;$$

$$\Delta w_{BC} = 0,1 + 0,00166 = 0,102;$$

$$\Delta w_{BD} = 0,3 - 0,00043 = -0,03;$$

$$\Delta w_{1C} = -0,4 + 0,00332 = -0,397;$$

$$\Delta W_{AD} = -0,3 - 0,00086 = -0,301.$$

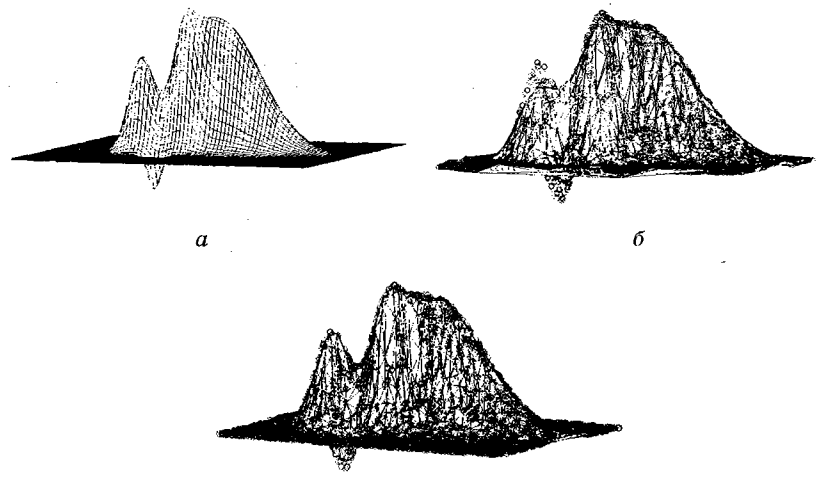
Приклад 3.11. На рис. 3.32, 3.33 наведено результати апроксимації функції (рис. 3.32, *а*)

$$f(x,y) = 0,5\exp\left[-\frac{(9x-2)^2 + (9y-2)^2}{4}\right] + 0,75\exp\left[-\frac{(9x+1)^2}{49} - \frac{(9y+1)^2}{10}\right] +$$

$$+ 0,5\exp\left[-\frac{(9x-7)^2 + (9y-3)^2}{4}\right] - 0,2\exp\left[-(9x-4)^2 - (9y-7)^2\right],$$

багатошаровим персептроном з одним (рис. 3.32,*б*) і двома (рис. 3.33) прихованими шарами. У першому випадку прихований шар містив 30 нейронів, у другому — перший прихований шар — 10, а другий — 6 нейронів, тобто двошаровий персептрон мав вигляд 2-10-6-1. Ця функція була промодельована із

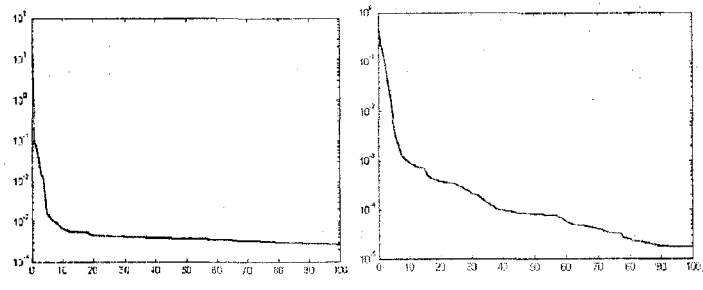
кроком дискретизації 0,1. Під час навчання багатошарового персептрона за допомогою алгоритму зворотного поширення було згенеровано 5000 двовимірних випадкових величин. Самі пари, що навчають, показано на рисунку точками. Задовільна збіжність процесу навчання була досягнута після 100 тактів.



а *б* *в*

Рис. 3.32. Апроксимація функції $f(x, y)$ за допомогою багатошарового персептрона

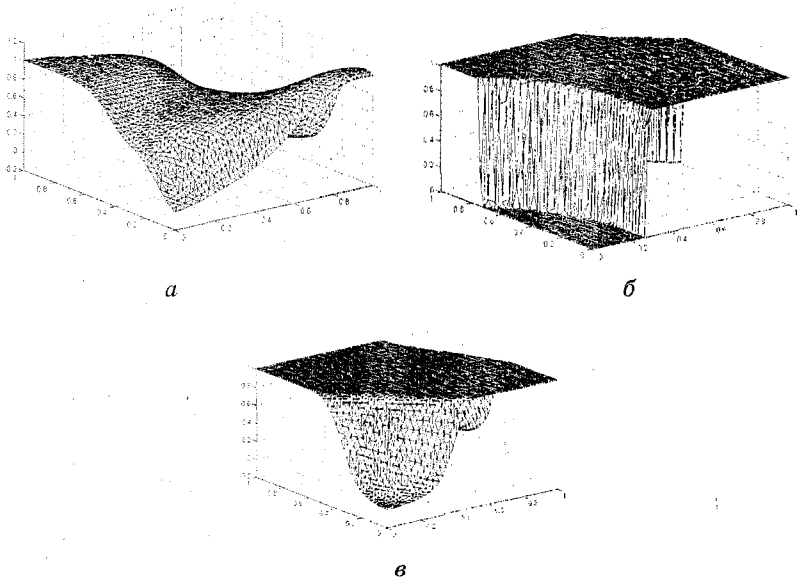
Зміну величин помилок апроксимації зображено на рис. 3.33.



а *б*

Рис. 3.33. Зміна помилок для персептрона, що містить: *а* — один і *б* — два прихованих шари

Приклад 3.12. На рис. 3.34 наведено результати моделювання логічної функції «що виключає АБО» на персептроні 2-7-2-1, що має два прихованих шари, причому рисунки *а*, *б* та *в* відбивають реалізації даних функцій на персептронах з активаційними функціями (2.7), (2.13) і (2.14) відповідно.



а *б* *в*

Рис. 3.34. Реалізація функції «що виключає АБО» на персептроні 2-7-2-1

Слід зазначити, що призначення випадкових початкових значень вагових параметрів призводить до того, що при повторному моделюванні з іншими початковими значеннями форма поверхонь може бути іншою.

Контрольні запитання та завдання

1. Що являє собою одношаровий персептрон?
2. Поясніть теорему збіжності для персептрона.
3. Які можливості одношарового персептрона?
4. У чому відмінність Адаліни від персептрона?
5. Які існують різновиди Адаліни?

6. У чому суть алгоритму Уідроу — Гоффа?
7. Що таке лінійна роздільність?
8. Наведіть приклади багатошарових ШНМ.
9. Розгляньте способи реалізації на багатошаровому персептроні різних булевих функцій.
10. Поясніть метод зворотного поширення помилки. Які труднощі виникають у процесі його реалізації?
11. Що являє собою вхідна зірка Ґроссберґа?
12. Яким способом здійснюється навчання вхідної зірки?
13. Яка структура вихідної зірки?
14. Запишіть правило навчання вихідної зірки.

4. ШНМ-КОМПАРАТОР

Дана мережа, що призначена для виділення з усієї сукупності вхідних сигналів сигналу, який має максимальне значення, запропонована *Т. Мартіном* [78]. На відміну від інших типів ШНМ, ця мережа використовує постійні значення ваг, що призначені до початку функціонування мережі й незмінні в процесі її роботи.

Структуру багатошарового ШНМ-компаратора подано на рис. 4.1.

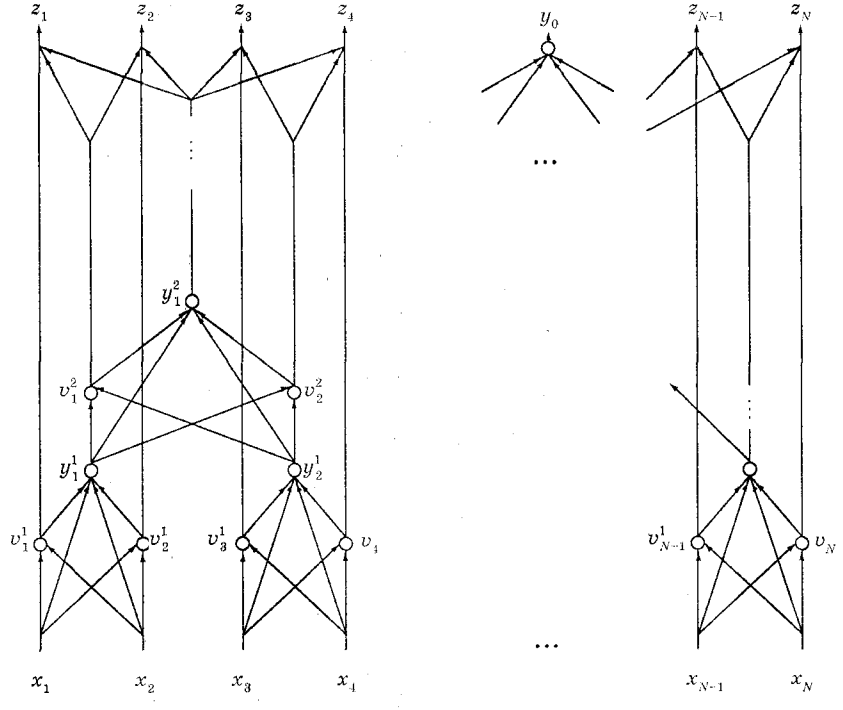


Рис. 4.1. Структура багатошарової мережі-компаратора

Число шарів K залежить від розмірності вхідного сигналу N і визначається в такий спосіб:

$$K =]\log_2 N[, \qquad (4.1)$$

де $]\cdot[$ означає округлення вбік найближчого більшого цілого числа.

Мережа складається із з'єднаних певним чином двовходових ШНМ-компараторів, структуру яких зображено на рис. 4.2.

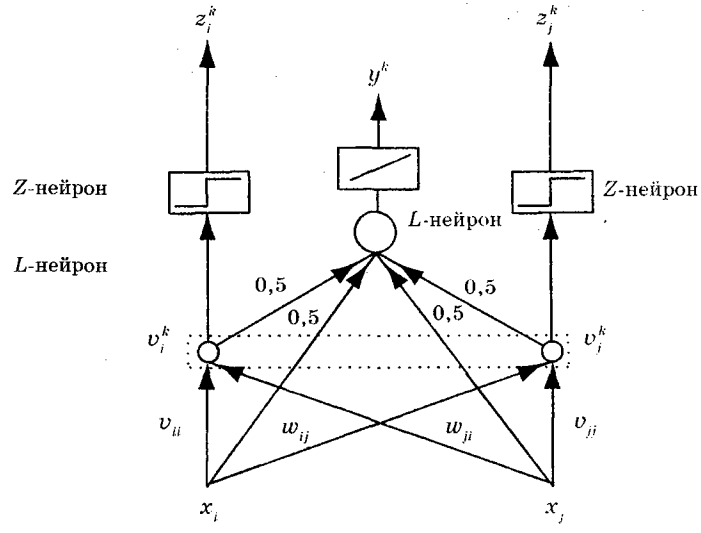


Рис. 4.2. Двовходовий ШНМ-компаратор

Мережу утворюють два типи нейронів: L-нейрони з лінійною функцією активації й Z-нейрони з пороговою функцією активації. Перші служать для обчислення значень вихідних сигналів прихованих шарів мережі v_i^k, y_i^k (тут k — номер прихованого шару, $k = \overline{1, K}$) і значення сигналу y_0 у вихідному шарі, що є середнім значенням вхідних сигналів. Вихідними сигналами Z-нейронів прихованих шарів є «+1» й «−1».

Як ми вже зазначали, ваги всіх нейронів призначаються до початку функціонування й залишаються постійними. При цьому ваги L-нейронів приймаються рівними 0,5, а ваги Z-нейронів — $w_{ij} = 1$ при $i = j$ й $w_{ij} = -1$ при $i \neq j$.

При надходженні на вхід мережі вектора вхідних сигналів $\boldsymbol{x}$ обчислюються вихідні сигнали y_j L-нейронів прихованого шару

$$v_i^k = \sum w_{ij} y_i^{k-1}, \qquad (4.2)$$

є вхідними сигналами віповідних Z-нейронів.

Виходи Z-нейронів усіх шарів обчислюються в такий спосіб:

$$z_i = \begin{cases} 1, & \text{якщо } y_i > \Theta; \\ -1, & \text{якщо } y_i \le \Theta, \end{cases} \qquad (4.3)$$

де Θ — пороговий сигнал (зсув).

На відміну від прихованих шарів, у вихідному шарі тільки на виході одного k-го Z-нейрона, що відповідає максимальному вхідному сигналу $x_k = \max\limits_i x_i$, з'являється сигнал «+1». На виходах всіх інших Z-нейронів будуть сигнали «−1».

L-нейрон вихідного шару формує сигнал

$$y_0 = \sum_{j=1}^{n}\sum_{i=1}^{n} w_{ij} x_i + \sum_{j=1}^{n}\sum_{i=1}^{n} w_{ij} y_i^k = \frac{1}{N}\sum_{i=1}^{N} x_i, \qquad (4.4)$$

величина якого є середнім значенням вхідних сигналів.

Приклад 4.1. Розглянемо побудову ШНМ-компаратора для визначення максимального компонента вектора.

Нехай $x = (0\ 1\ 2\ -1\ 3\ 4\ -2\ 1)$.

Число вихідних Z-нейронів дорівнює числу компонент, тобто $N = 8$. Приймемо

$$\Theta = \begin{cases} 0 \text{ для } Z\text{-нейронів прихованих шарів;} \\ 2 \text{ для } Z\text{-нейронів вихідного шару.} \end{cases}$$

Оскільки компаратор виконує порівняння попарно, а $8 = 2^3$, ШНМ матиме 3 шари.

Вихідні сигнали v_j^1 $(j = \overline{1,8})$ першого прихованого шару обчислюються згідно з (4.2) у такий спосіб:

$v_1^1 = 1 \cdot x_1 + (-1) \cdot x_2 = -1;$ $\quad v_2^1 = (-1) \cdot x_1 + 1 \cdot x_2 = 1;$

$v_3^1 = 1 \cdot x_3 + (-1) \cdot x_4 = 3;$ $\quad v_4^1 = (-1) \cdot x_3 + 1 \cdot x_4 = -3;$

$v_5^1 = 1 \cdot x_5 + (-1) \cdot x_6 = -1;$ $\quad v_6^1 = (-1) \cdot x_5 + 1 \cdot x_6 = 1;$

$v_7^1 = 1 \cdot x_7 + (-1) \cdot x_8 = -3;$ $\quad v_8^1 = (-1) \cdot x_7 + 1 \cdot x_8 = 3.$

Допоміжні сигнали z_j^1 $(j = \overline{1,8})$, необхідні для формування вихідних сигналів Z-нейронів, обчислюються відповідно до (4.3):

$z_1^1 = -1;\ z_2^1 = 1;\ z_3^1 = 1;\ z_4^1 = -1;\ z_5^1 = -1;\ z_6^1 = 1;\ z_7^1 = -1;\ z_8^1 = 1.$

Вихідні сигнали y_j^1 $(j = \overline{1,4})$ L-нейронів першого прихованого шару обчислюються за формулою (4.4):

$y_1^1 = 0{,}5(x_1 + x_2 + v_1^1 + v_2^1) = 0{,}5;$ $\quad y_2^1 = 0{,}5(x_3 + x_4 + v_3^1 + v_4^1) = 0{,}5;$

$y_3^1 = 0{,}5(x_5 + x_6 + v_5^1 + v_6^1) = 3{,}5;$ $\quad y_4^1 = 0{,}5(x_7 + x_8 + v_7^1 + v_8^1) = -0{,}5.$

Дані вихідні сигнали Z- і L-нейронів першого прихованого шару є вхідними для нейронів другого прихованого шару. Виконуючи аналогічні обчислення для всіх шарів, отримуємо схему ШНМ-компаратора, наведену на рис. 4.3.

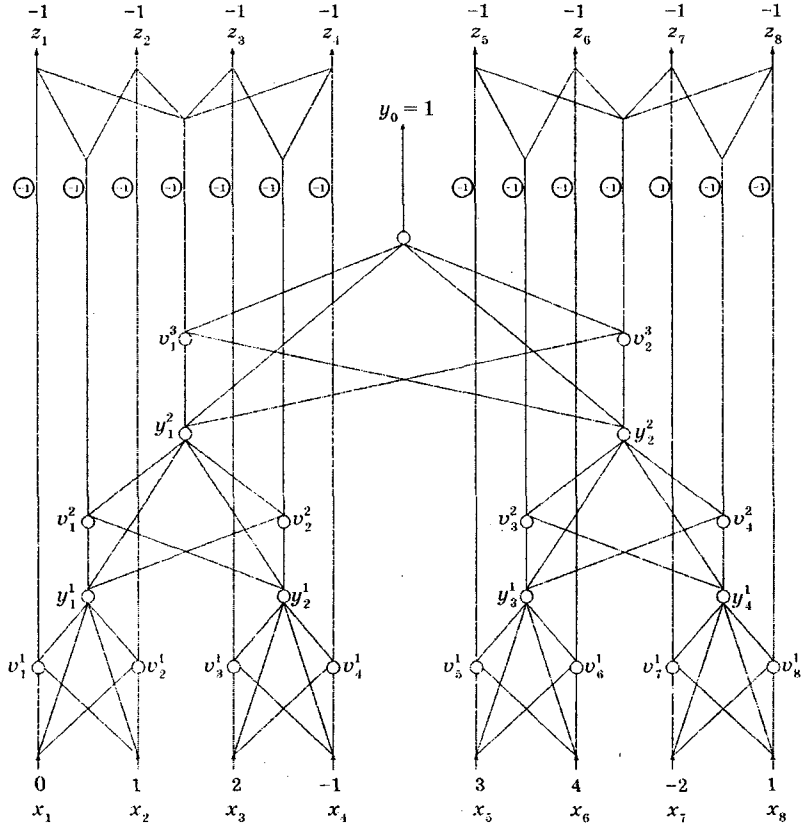


Рис. 4.3. Синтезований восьмивходовий ШНМ-компаратор

Як видно з рисунка, на виході тільки Z-нейрона Z_6 з'являється сигнал «+1», що відповідає максимальному компоненту вхідного сигналу $x_6 = 4$. Величина сигналу на виході L-нейрона вихідного шару

$$y_0 = 0{,}5(y_1^2 + y_2^2 + v_1^3 + v_2^3) = 1$$

дорівнює середньому значенню вхідних сигналів

$$x_{\text{сер}} = \frac{1}{8}\sum_{i=1}^{8} x_i = 1.$$

Контрольні запитання

1. Яка структура багатошарового ШНМ-компаратора?
2. Як визначається необхідна кількість шарів багатошарового ШНМ-компаратора?
3. Чим відрізняються L- і Z-нейрони, що утворюють структуру мережі?
4. Яким способом визначаються ваги мережі?

5. АСОЦІАТИВНА ПАМ'ЯТЬ

Вивчаючи природу зв'язків у мозку, *Ф. Лемент* [50] виділив прецизійні й асоціативні з'єднання. Перші характерні для невеликої кількості клітин і є впорядкованими. Сила зв'язків других, характерних для значно більшої кількості клітин, неоднакова, залежить від типу розглянутої мережі й відрегульована «на основі досвіду» так, що ті провідні шляхи, які часто активуються спільно, якимось чином підсилюються. Саме ці з'єднання й утворюють асоціації, тобто зв'язки між явищами такі, що виклик одного з них призведе до виклику й іншого. Пам'ять людини асоціативна. З асоціаціями ми постійно зустрічаємося в повсякденному житті. Так, глянувши на обличчя людини, ми згадуємо його ім'я (хоча, на жаль, не завжди). Асоціаціями широко користуються під час навчання. Наприклад, запам'ятовування таблиці множення полягає у встановленні зв'язків (асоціацій) між парами співмножників і відповідними цим парам добутками. За необхідності згадати добуток — ці асоціації з пам'яті викликаються.

На асоціативному навчанні базуються й знамениті експерименти на собаках фізіолога *І. Павлова*.

На відміну від звичайної пам'яті, у якій для пошуку інформації використовується адреса, в асоціативній пам'яті пошук інформації здійснюється за допомогою ключа — послідовності бітів, порівнюваної з усіма ключами інформації, що зберігається. Такий вид пам'яті називається *пам'яттю, що адресується за змістом* (*CAM — content adressable memory*).

Здатність пам'яті до відтворення асоціацій дозволяє вирішувати таке важливе завдання, як відновлення спотвореної або зашумленої інформації, наприклад візуальної. Ми в змозі згадати людину, навіть якщо побачили частину її обличчя.

Найбільш бурхливий період розвитку ШНМ даного типу припадає на 70–80-ті роки XX ст. Значний внесок у вирішення проблеми асоціативного навчання внесли *Дж. Андерсон* [15, 16], *С. Гроссберг*

[20, 21], *Т. Когонен* [10–14]. Незважаючи на простоту розглянутих у даному розділі мереж і на обмежені можливості вирішення практичних задач, вони є базисом для створення потужніших мереж, що використовують крім асоціацій конкурентне навчання (як, наприклад, мережа Когонена, що описана в розділі 5).

5.1. Асоціативна мережа прямого поширення

Проста лінійна модель містить, як і персептрон, один або два шари, однак на відміну від нього використовує іншу парадигму навчання. Лінійна модель у загальному випадку використовується не для класифікації образів, а для їхнього асоціативного пошуку й відновлення. Тому вона, на відміну від персептрона, має не один, а багато виходів.

Як й у персептроні, на вхід лінійної моделі подається деякий образ, що у випадку роботи даної мережі як асоціатора є ключем для відповідного вихідного сигналу.

Найпростіша асоціативна мережа прямого поширення, наведена на рис. 5.1, перетворить N вхідних сигналів у M вихідних. Подані мережі асоціативні пари $\{(\boldsymbol{x}_p, \boldsymbol{y}_p),\ p=\overline{1,P}\}$, кожна з яких характеризується своєю ваговою матрицею W_p розмірності $N \times M$, запам'ятовуються, і завдання мережі полягає у видачі на виході сигналу (образу) $\boldsymbol{y}_p$, асоційованого з $\boldsymbol{x}_p$, якщо на її вхід надходить сигнал (образ) $\boldsymbol{x}$, подібний $\boldsymbol{x}_p$, тобто віддалений від $\boldsymbol{x}_p$ на мінімальній відстані Хеммінга

$$d(\boldsymbol{x}, \boldsymbol{x}_p) = \min_k (\boldsymbol{x}, \boldsymbol{x}_k). \qquad (5.1)$$

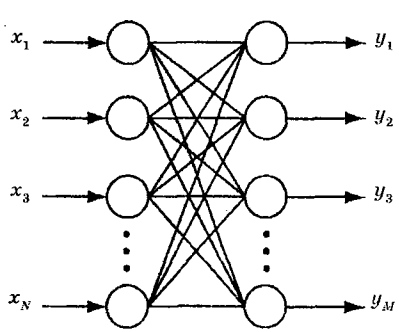


Рис. 5.1. Найпростіша асоціативна мережа прямого поширення

Розрізнюють два види асоціативної пам'яті: автоасоціативна й гетероасоціативна. Задача автоасоціативної пам'яті, що зберігає P образів, але не асоціативних пар, — видача сигналу $\boldsymbol{y}$, що відповідає $\boldsymbol{x}$, тобто $\boldsymbol{y} = \boldsymbol{x}$. Гетероасоціативна пам'ять, що зберігає асоціативні пари, як правило, формує сигнал $\boldsymbol{y} \neq \boldsymbol{x}$.

Найпростішою асоціативною мережею є *лінійний асоціатор*, що являє собою одношарову мережу й використовує тільки операцію скалярного добутку векторів

$$y_j = \sum_{i=1}^{N} w_{ij} x_i = \boldsymbol{w}_j \boldsymbol{x}, \qquad j = \overline{1, M}, \qquad (5.2)$$

де $\boldsymbol{w}_j$ — j-й рядок матриці ваг W.

Однак, як було зазначено вище для персептрона, введення додаткових шарів призводить до набуття мережею нових позитивних властивостей, це не стосується лінійного асоціатора. Справа в тому, що якщо одношаровий лінійний асоціатор описується лінійним рівнянням з ваговою матрицею W, тобто $\boldsymbol{y} = W\boldsymbol{x}$, то додавання другого шару, що характеризується ваговою матрицею V, дає $\boldsymbol{z} = V\boldsymbol{y} = VW\boldsymbol{x} = T\boldsymbol{x}$, де $T = VW$, тобто знову виходить лінійне перетворення. Таким чином, багатошаровий лінійний асоціатор еквівалентний одношаровому.

Характерною рисою лінійної моделі є її здатність відновлювати частково зруйнований образ. На рис. 5.2 наведено приклад відновлення даною мережею спотвореної букви «А», що подається в бінарному вигляді на вхід моделі. Модель, що функціонує в такий спосіб, називається *асоціативною пам'яттю*.

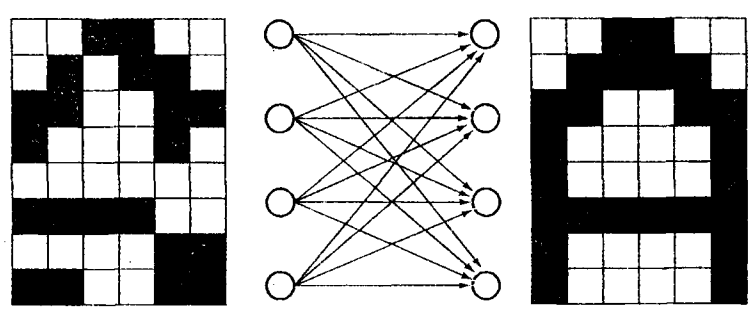


Рис. 5.2. Відновлення мережею спотворень букви

Слід зазначити, що лінійна нейромережа відводить для кожного образу не деяке певне місце, а розподіляє інформацію про всі

збережені образи у вагових коефіцієнтах мережі. Тому при подачі інформації на вхід мережі відбувається не порівняння пропонованого образу зі збереженим, а лінійне перетворення вхідного сигналу, що використовується для обчислення вихідного сигналу.

Ємність асоціативної пам'яті обмежена її фізичними розмірами й видом використовуваного перетворення (лінійним).

5.2. Алгоритми навчання асоціативної пам'яті

Асоціації, або пам'ять мережі, реалізуються у ваговій матриці W, тому завдання навчання мережі полягає у визначенні таких значень елементів вагової матриці W, які забезпечували б точне відтворення образу, що викликається. Сьогодні для навчання використовується правило Гебба, дельта-правило й метод найменших квадратів, що нерідко називається *методом псевдоінверсії*.

5.2.1. Правило навчання Гебба

Як зазначалося раніше, правило Гебба засноване на тому факті, що зміна синаптичної сили (ваги) нейрона пропорційно зміні пре- і постсинаптичної сил, що є відповідно вхідним та вихідним сигналами нейрона, і може бути записане в такий спосіб:

$$w_{ij} = \gamma x_i y_j, \quad i = \overline{1,N} \quad j = \overline{1,M} \tag{5.3}$$

або в матричному вигляді

$$W = \gamma \boldsymbol{y}\boldsymbol{x}^T, \tag{5.4}$$

де γ — коефіцієнт пропорційності (навчання).

Зазначимо, що оскільки співвідношеннями (5.3), (5.4) описується кожна з P пар $(\boldsymbol{x}_p, \boldsymbol{y}_p)$, що навчають, добре було б це подати у такий спосіб:

$$W_p = \gamma y_p x_p^T, \quad p = \overline{1,P}. \tag{5.5}$$

Тоді матриця ваг W, що відбиває зберігання в пам'яті всіх P образів, матиме вигляд:

$$W = \sum_{p=1}^{P} W_p. \tag{5.6}$$

Оскільки асоціативна пам'ять заснована на запам'ятовуванні предметів, спостережуваних одночасно, співвідношення (5.3) і (5.4) припускають одночасно появу $\boldsymbol{x}$ й $\boldsymbol{y}$.

У записі (5.3) вага w_{ij}, що є синапсом між входом x_i і виходом y_j, відбиває той факт, що якщо додатний (від'ємний) x_i викликає додатний (від'ємний) y_j, то синаптичний зв'язок підсилюється. Ця зміна синаптичного зв'язку може бути записана у такий спосіб:

$$W(k+1) = W(k) + \gamma \boldsymbol{y}(k)\boldsymbol{x}^T(k), \tag{5.7}$$

де індекс k відповідає k-м вхідним і вихідним векторам.

Правило (5.7), що містить фактичні значення входів і виходів мережі й не використовує ніякої інформації, що стосується необхідної мети, називається *неконтрольованим правилом навчання Гебба*. Відповідно до цього правила зміна ваги пропорційна виникненню активності на будь-якій стороні синапса, тобто вага збільшується не тільки коли $\boldsymbol{x}$ й $\boldsymbol{y}$ додатні, але й коли обидві змінні від'ємні. Інакше кажучи, правило (5.3) засновано на обчисленні кореляції між вхідними й вихідними сигналами мережі.

Співвідношення (5.7) може бути переписане у вигляді

$$W(k) = W(0) + \gamma \sum_{i=1}^{k} \boldsymbol{y}(i)\boldsymbol{x}^T(i), \tag{5.8}$$

звідки видно, що при тривалому поданні входів, тобто із зростанням k значення ваг можуть стати як завгодно великими.

Приклад 5.1. Якщо мережі подаються вхідні образи $\boldsymbol{x}^T(1) = (1\ 1)$ $\boldsymbol{x}^T(2) = (1\ 1)$..., що навчають, а вихідний сигнал формується пороговою функцією

$$y = \begin{cases} 1, & \text{якщо } \boldsymbol{w}^T\boldsymbol{x} > 0; \\ 0, & \text{якщо } \boldsymbol{w}^T\boldsymbol{x} \le 0. \end{cases}$$

Розглянемо настроювання ваг за алгоритмом (5.7) з $\gamma = 1$. Нехай $\boldsymbol{w}^T(0) = (1\ 1)$. Для послідовних моментів часу маємо

$k = 1$ $\quad \boldsymbol{w}(0)^T\boldsymbol{x}(1) = 1\cdot 1 + 1\cdot 1 = 2, \quad y(1) = 1,$

$k = 2$ $\quad \boldsymbol{w}(1)^T\boldsymbol{x}(2) = (2\ \ 2)\begin{pmatrix}1\\1\end{pmatrix} = 4, \quad y(2) = 1,$

$$\boldsymbol{w}(1) = \begin{pmatrix}1\\1\end{pmatrix} + 1\cdot\begin{pmatrix}1\\1\end{pmatrix} = \begin{pmatrix}2\\2\end{pmatrix};$$

$$\boldsymbol{w}(2) = \boldsymbol{w}(1) + y(2)\boldsymbol{x}(2) = \begin{pmatrix}2\\2\end{pmatrix} + 1\begin{pmatrix}1\\1\end{pmatrix} = \begin{pmatrix}3\\3\end{pmatrix}.$$

Таким чином, зі збільшенням кількості подавань образів вагові коефіцієнти зростають і з часом можуть стати як завгодно великими. Це не погоджено з біологічними принципами, тому що синаптичні ваги не можуть зростати нескінченно, і є істотним недоліком неконтрольованого правила (5.7).

Іншим недоліком, тісно пов'язаним з першим, є відсутність у (5.7) механізму зменшення ваг, що приводить знову-таки до зростання ваг мережі та її неправильної реакції під час впливу шумів на входи й виходи мережі.

Для усунення зазначених недоліків в алгоритмі (5.7) використовують механізм забування, що має вигляд

$$W(k) = W(k-1) + \gamma \mathbf{y}(k)\mathbf{x}^T(k) - \alpha W(k-1) = \\ = (1-\alpha)W(k-1) + \gamma \mathbf{y}(k)\mathbf{x}^T(k), \qquad (5.9)$$

де $\alpha \in (0,1)$ — коефіцієнт забування.

При $\alpha \to 0$ алгоритм (5.9) перетворюється в (5.7), а при $\alpha \to 1$ алгоритм забуває стару інформацію й пам'ятає тільки найновішу, усуваючи тим самим нескінченне зростання вагових коефіцієнтів. Максимальне значення $w_{ij}^{\max}$ визначається величинами α і γ. Оскільки максимізація швидкості навчання відбувається при $x_i = y_i = 1$, для сталого стану, коли ваги не змінюються, можна записати

$$w_{ij}^{\max} = (1-\alpha)w_{ij}^{\max} + \gamma x_i y_j = (1-\alpha)w_{ij}^{\max} + \gamma, \qquad (5.10)$$

звідки

$$w_{ij}^{\max} = \frac{\gamma}{\alpha}. \qquad (5.11)$$

Приклад 5.2. Нехай мережі подають образи, розглянуті в прикладі 5.2. Використання алгоритму (5.10) з $\gamma = 1$ й $\alpha = 0{,}1$ дає

$$k=1 \quad y(1)=1 \qquad w(1) = (1-0{,}1)\begin{pmatrix}1\\1\end{pmatrix} + 1\cdot\begin{pmatrix}1\\1\end{pmatrix} = \begin{pmatrix}1{,}9\\1{,}9\end{pmatrix};$$

$$k=2 \quad y(2)=1 \qquad w(2) = (1-0{,}1)\begin{pmatrix}1{,}9\\1{,}9\end{pmatrix} + 1\cdot\begin{pmatrix}1\\1\end{pmatrix} = \begin{pmatrix}2{,}71\\2{,}71\end{pmatrix},$$

тобто зростання значень вагової матриці відбувається повільніше, а максимальне значення її елементів, визначене відповідно до (5.11), дорівнює

$$w^{\max} = \frac{1}{0{,}1} = 10.$$

Правило (5.10) гарантує, що малі зміни вагових параметрів, що виникають внаслідок впливу шуму, з часом зникнуть, тобто шум не призведе до появи помилкових асоціацій.

З іншого боку, за відсутності нових сигналів, що навчають, збережені мережею асоціації губляться. Так, при $x_i = 0$ з (5.7) отримуємо

$$w_{ij}(k) = (1-\alpha)\, w_{ij}(k-1), \qquad (5.12)$$

а оскільки $\alpha \in (0,1)$, то $w_{ij}(k) < w_{ij}(k-1)$.

Таким чином, використання правила навчання із забуванням вимагає повторення подання мережі послідовностей, що навчають.

У зв'язку з цим доцільною є така модифікація алгоритму (5.7), що дозволяла б використати механізм забування (обмеження зростання елементів вагової матриці), тільки якщо нейрон активний, тобто коли $y \neq 0$. Ця ідея реалізується у такий спосіб:

$$w_{ij}(k) = w_{ij}(k-1) + \gamma y_i(k) x_j(k) - \alpha y_i(k)\, w_{ij}(k-1), \qquad (5.13)$$

тобто переходом до алгоритму (3.35).

Дійсно, при $y_i(k) = 0$ $w_{ij}(k) = w_{ij}(k-1)$, тобто стирання інформації не відбувається.

Якщо в (5.13) вибрати $\gamma = \alpha$, отримаємо співвідношення

$$w_{ij}(k) = w_{ij}(k-1) + \alpha y_i(k)\,(x_j(k) - w_{ij}(k-1)), \qquad (5.14)$$

що називається *правилом вхідного зв'язування* або *стандартним правилом навчання вхідної зірки* (3.37), графічну ілюстрацію корекції ваг відповідно до якого наведено на рис. 3.11.

Особливістю правила (5.14) є те, що якщо вектор вхідних сигналів нормалізований, вихідні вектори також будуть нормалізованими.

При контрольованому навчанні правило Гебба (5.5) використовує замість фактичного значення виходу $\mathbf{y}$ необхідне $\mathbf{y}^*$

$$W_p = \gamma \mathbf{y}_p^* \mathbf{x}_p^T. \qquad (5.15)$$

Особливості цього правила розглянемо для випадку $\gamma = 1$. Помножимо обидві частини (5.15) зліва на $\mathbf{x}_p$

$$W_p \mathbf{x}_p = \mathbf{y}_p^* \mathbf{x}_p^T \mathbf{x}_p. \qquad (5.16)$$

З (5.16) випливає, що при ортонормованих вхідних сигналах, $\|x_p\|^2 = 1$, матриця (5.15) відразу ж дає бажаний вихідний сигнал

$$W_p x_p = y_p^*. \tag{5.17}$$

Оскільки мережа має зберігати всі P образів, матриця мережі W визначається згідно з (5.6). Під час подання мережі вхідного образу x_p вона має видати асоційований з x_p вихідний сигнал y_p. Фактичний вихідний сигнал при поданні деякого образу x_k обчислюється у такий спосіб:

$$y_k = W x_k = \left(\sum_{p=1}^{P} y_p^* x_p^T \right) x_k = y_k^* x_k^T x_k + \sum_{p \neq k}^{P} y_p^* x_p^T x_k. \tag{5.18}$$

Якщо вхідні сигнали ортонормовані

$$x_p^T x_k = \begin{cases} 1 & \text{при } p = k; \\ 0 & \text{при } p \neq k \end{cases} \tag{5.19}$$

й ортогональні, то $y_k = y_k^*$, тобто мережа безпомилково відтворює асоційований із заданим вхідним образом вихідний. Якщо вхідні вектори не ортонормовані, вихідний сигнал міститиме деяку помилку, величина якої залежить від ступеня їхньої лінійної залежності або корельованості. Це справедливо незалежно від того, збігається чи ні розмірність векторів входу й виходу мережі.

Приклад 5.3. Нехай на вхід лінійної мережі надходить послідовність ($N = M = 3$), що навчає

$$x_1^T = (-0{,}6 \quad 0 \quad 0{,}8); \quad (y_1^*)^T = (1 \quad 1 \quad 0);$$

$$x_2^T = (0 \quad 1 \quad 0); \quad (y_2^*)^T = (0 \quad 1 \quad 1);$$

$$x_1^T = (0{,}8 \quad 0 \quad 0{,}6); \quad (y_3^*)^T = (1 \quad 0 \quad 1).$$

Обчислимо матриці ваг

$$W_1 = y_1^* x_1^T = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} (-0{,}6 \quad 0 \quad 0{,}8) = \begin{pmatrix} -0{,}6 & 0 & 0{,}8 \\ -0{,}6 & 0 & 0{,}8 \\ 0 & 0 & 0 \end{pmatrix};$$

$$W_2 = y_2^* x_2^T = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} (0 \quad 1 \quad 0) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix};$$

$$W = \sum_{i=1}^{3} W_i = \begin{pmatrix} 0{,}2 & 0 & 1{,}4 \\ -0{,}6 & 1 & 0{,}8 \\ 0{,}8 & 1 & 0{,}6 \end{pmatrix}.$$

За умов подачі на вхід образів, які запам'ятовано, мережа їх правильно відтворює

$$y_1 = W x_1 = \begin{pmatrix} 0{,}2 & 0 & 1{,}4 \\ -0{,}6 & 1 & 0{,}8 \\ 0{,}8 & 1 & 0{,}6 \end{pmatrix} \begin{pmatrix} -0{,}6 \\ 0 \\ 0{,}8 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = y_1^*;$$

$$y_2 = W x_2 = \begin{pmatrix} 0{,}2 & 0 & 1{,}4 \\ -0{,}6 & 1 & 0{,}8 \\ 0{,}8 & 1 & 0{,}6 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = y_2^*;$$

$$y_3 = W x_3 = \begin{pmatrix} 0{,}2 & 0 & 1{,}4 \\ -0{,}6 & 1 & 0{,}8 \\ 0{,}8 & 1 & 0{,}6 \end{pmatrix} \begin{pmatrix} 0{,}8 \\ 0 \\ 0{,}6 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = y_3^*.$$

Приклад 5.4. Нехай на вхід мережі подаються пари $\{x_i, y_i^*\}$ ($i = 1,2$), що навчають, причому вхідні образи є ортонормованими ($N = 4$; $M = 2$)

$$x_1^T = (-0{,}5 \quad 0{,}5 \quad 0{,}5 \quad -0{,}5); \quad (y_1^*)^T = (-1 \quad 1);$$

$$x_2^T = (-0{,}5 \quad -0{,}5 \quad 0{,}5 \quad 0{,}5); \quad (y_2^*)^T = (1 \quad 1).$$

Визначимо матриці ваг

$$W_1 = y_1^* x_1^T = \begin{pmatrix} -1 \\ 1 \end{pmatrix} (-0{,}5 \quad 0{,}5 \quad 0{,}5 \quad -0{,}5) = \begin{pmatrix} 0{,}5 & -0{,}5 & -0{,}5 & 0{,}5 \\ -0{,}5 & 0{,}5 & 0{,}5 & -0{,}5 \end{pmatrix};$$

$$W_2 = y_2^* x_2^T = \begin{pmatrix} 1 \\ 1 \end{pmatrix} (-0{,}5 \quad -0{,}5 \quad 0{,}5 \quad 0{,}5) = \begin{pmatrix} -0{,}5 & -0{,}5 & 0{,}5 & 0{,}5 \\ -0{,}5 & -0{,}5 & 0{,}5 & 0{,}5 \end{pmatrix};$$

$$W = \sum_{i=1}^{2} W_i = \begin{pmatrix} 0 & -1 & 0 & 1 \\ -1 & 0 & 1 & 0 \end{pmatrix}$$

Обчислюючи вихідні сигнали, отримуємо

$$Wx_1 = \begin{pmatrix} 0 & -1 & 0 & 1 \\ -1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} -0{,}5 \\ 0{,}5 \\ 0{,}5 \\ -0.5 \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \end{pmatrix} = y_1^*;$$

$$Wx_2 = \begin{pmatrix} 0 & -1 & 0 & 1 \\ -1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} -0{,}5 \\ -0{,}5 \\ 0{,}5 \\ 0{,}5 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} = y_2^*.$$

Мережа правильно відтворила асоційовані образи.

Приклад 5.5. Нехай на вхід мережі надходять пари, що навчають:

$x_1^T = (-1\ \ 1\ \ 1);\ (y_1^*)^T = (-1\ \ 1);$

$x_2^T = (-1\ \ 1\ \ -1);\ (y_1^*)^T = (1\ \ 1).$

Легко перевірити, що образи x_1 й x_2 не ортонормовані. Обчислимо вагові матриці

$$W_1 = y_1^* x_1 = \begin{pmatrix} -1 \\ 1 \end{pmatrix} (-1\ \ 1\ \ 1) = \begin{pmatrix} 1 & -1 & -1 \\ -1 & 1 & 1 \end{pmatrix};$$

$$W_2 = y_2^* x_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} (-1\ \ 1\ \ -1) = \begin{pmatrix} -1 & 1 & -1 \\ -1 & 1 & -1 \end{pmatrix};$$

$$W = \sum_{i=1}^{2} W_i = \begin{pmatrix} 0 & 0 & -2 \\ -2 & 2 & 0 \end{pmatrix}.$$

Застосуємо дану матрицю до пропонованих образів

$$Wx_1 = \begin{pmatrix} 0 & 0 & -2 \\ -2 & 2 & 0 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -2 \\ 4 \end{pmatrix} \neq y_1^*;$$

$$Wx_2 = \begin{pmatrix} 0 & 0 & -2 \\ -2 & 2 & 0 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix} \neq y_2^*.$$

Таким чином, хоча мережа й класифікує подані неортогональні образи, її фактичний вихідний сигнал відрізнятиметься від необхідного.

5.2.2. Дельта-правило

Дане правило, що назване також *правилом Уідроу — Гоффа*, реалізує градієнтну процедуру мінімізації квадратичного функціонала від помилки за всіма пропонованими мережі образами. На відміну від неконтрольованого правила Гебба, розглянутого в 5.2.1, дане правило реалізує *контрольоване навчання*, тому що використовує інформацію про бажаний вихідний сигнал мережі y^*.

Якщо для p-ї пари (x_p, y_p^*), $p = \overline{1,P}$, що навчає, вихідний сигнал y_j поданий співвідношенням (5.2), квадратичний критерій навчання має вигляд

$$I = \sum_{p=1}^{P} \sum_{j=1}^{M} \left(y_{pj}^* - \sum_{i=1}^{N} w_{ij} x_{pi} \right)^2, \tag{5.20}$$

і вагові параметри корегуються за правилом

$$\Delta w_{ij} = -\gamma \frac{\partial I}{\partial w_{ij}}, \tag{5.21}$$

яке з урахуванням (5.13) може бути записане у такий спосіб:

$$\Delta w_{ij} = \gamma (y_{pj}^* - y_{pj}) x_{pi} \tag{5.22}$$

або в матричній формі

$$\Delta W = \gamma \sum_{p=1}^{P} (y_p^* - W x_p) x_p^T. \tag{5.23}$$

Перевага дельта-правила полягає в тому, що воно дозволяє поновлювати елементи вагової матриці після кожного подання вхідного образу, тобто забезпечує адаптацію мережі до умов, що змінюються.

Приклад настроювання ваг за алгоритмом (5.17) наведено у розділі 3 (приклад 3.3).

5.2.3. Алгоритми навчання, що використовують операцію псевдообернення матриці вхідних сигналів

Якщо вхідні образи не ортогональні, правило Гебба настроює елементи вагової матриці з помилками. У цьому випадку більш ефективними є алгоритми навчання, що використовують операцію

псевдообернення вхідних сигналів. Як й у випадку застосування дельта-правила, ці алгоритми реалізують контрольоване навчання, що формально полягає у мінімізації квадратичного функціонала (5.20).

Запишемо рівняння (5.2) у матричній формі

$$WX = Y^*, \tag{5.24}$$

де $Y^* = (y_1, y_2, \ldots, y_p)$ — матриця необхідних вихідних сигналів $M \times P$; $X^* = (x_1, x_2, \ldots, x_p)$ — матриця вхідних сигналів $N \times P$; W — матриця ваг $M \times N$.

Якщо матриця є квадратною й оборотною, то розв'язок має вигляд

$$W = Y^* X^{-1}. \tag{5.25}$$

На практиці ж така ситуація зустрічається надзвичайно рідко. Звичайно вхідні вектори x, що є стовпцями матриці X, вибираються незалежними, але їхня розмірність не збігається з їхньою кількістю, тобто матриця X буде прямокутною. Як відомо [70, 83], до такої матриці операція звичайного обернення незастосовна.

Тому в загальному випадку розв'язання, що мінімізує квадратичний функціонал (оцінка найменших квадратів)

$$\mathrm{I}(W) = \|Y^* - WX\|^2, \tag{5.26}$$

де $\|\cdot\|$ — матрична норма, що записується у вигляді

$$W = \lim_{\lambda \to 0}\left[Y^* X^T \left(XX^T + \lambda I\right)^{-1}\right] = Y^* X^+, \tag{5.27}$$

де X^+ — матриця, псевдообернена до X, тобто матриця, що має властивості [70]

$$\begin{aligned} XX^+X &= X; \\ X^+XX^+ &= X^+; \\ X^+X &= (X^+X)^T; \\ XX^+ &= (XX^+)^T. \end{aligned} \tag{5.28}$$

Якщо матриця X є квадратною, то $X^+ = X^{-1}$ й оцінка (5.27) приймає вигляд (5.25). Якщо ж кількість рядків матриці X (розмірність вектора x) більше кількості її стовпців (кількості вхідних образів), то псевдообернена матриця X^+ обчислюється в такий спосіб:

$$X^+ = (X^T X)^{-1} X^T. \tag{5.29}$$

Приклад 5.6. Нехай на вхід мережі надходять пари, що навчають, задані в прикладі 5.4, тобто

$$x_1^T = (-1\ \ 1\ \ 1);\ (y_1^*)^T = (-1\ \ 1);$$

$$x_2^T = (-1\ \ 1\ -1);\ (y_2^*)^T = (1\ \ 1).$$

Матриці вхідних і вихідних образів, що входять у рівняння (5.24), мають вигляд

$$X = \begin{pmatrix} -1 & -1 \\ 1 & 1 \\ 1 & -1 \end{pmatrix};\quad Y^* = \begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix}.$$

Псевдообернена матриця X^+ обчислюється згідно з (5.29)

$$X^+ = (X^T X)^{-1} X^T = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}^{-1} \begin{pmatrix} -1 & 1 & 1 \\ -1 & 1 & -1 \end{pmatrix} =$$

$$= \begin{pmatrix} 0{,}375 & -0{,}125 \\ -0{,}125 & 0{,}375 \end{pmatrix} \begin{pmatrix} -1 & 1 & 1 \\ -1 & 1 & -1 \end{pmatrix} = \begin{pmatrix} -0{,}25 & 0{,}25 & 0{,}5 \\ -0{,}25 & 0{,}25 & -0{,}5 \end{pmatrix}.$$

Матриця ваг визначається зі співвідношення (5.21)

$$W = \begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} -0{,}25 & 0{,}25 & 0{,}5 \\ -0{,}25 & 0{,}25 & -0{,}5 \end{pmatrix} = \begin{pmatrix} 0 & 0 & -1 \\ -0{,}5 & 0{,}5 & 0 \end{pmatrix}.$$

Застосовуючи отриману матрицю до вхідних образів, визначаємо

$$Wx_1 = \begin{pmatrix} 0 & 0 & -1 \\ -0{,}5 & 0{,}5 & 0 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \end{pmatrix} = y_1^*;$$

$$Wx_2 = \begin{pmatrix} 0 & 0 & -1 \\ -0{,}5 & 0{,}5 & 0 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} = y_2^*.$$

Таким чином, алгоритм, що використовує псевдообернення, дозволяє, на відміну від правила Гебба, точно відтворити асоційовані образи.

Хоча на перший погляд здається, що відповідно до даного правила ваги обчислюються однократно після подання мережі всіх пар, що навчають, можливе й послідовне (рекурентне) навчання

мережі, аналогічне дельта-правилу. Це питання більш докладно розглядатиметься у розділі 14.

5.2.4. Розпізнавання спотвореного образу

Якщо на вхід мережі подається збурений образ $\tilde{x}_p = x_p + \Delta x_p$, то на її виході з'являється сигнал

$$\tilde{y}_p = y_p + \Delta y_p, \tag{5.30}$$

де Δy_p — завадова складова, що може бути виражена як лінійна комбінація запам'ятованих векторів, що утворять ортонормальный базис

$$\Delta y_p = \sum_i \lambda_i x_i. \tag{5.31}$$

У цьому випадку вихідний сигнал мережі обчислюється у такий спосіб:

$$\begin{aligned} W\tilde{x}_p = W(x_p + \Delta y_p) = y_p^* + W\Delta y_p = y_p^* + \sum_i \sum_j \lambda_i y_i^* x_i^T x_j = \\ = y_p^* + \sum_i \lambda_i y_i^*. \end{aligned} \tag{5.32}$$

Отже, вихідний сигнал крім відповідного правильного (незбуреного) образу y_p^* міститиме деякі частини інших неспотворених образів, що зберігаються в мережі. Вплив завад підсилюється, якщо образи, що зберігаються, не ортогональні.

Кожний вхідний образ x відповідає найбільшою мірою тому збереженому образу x_p, значення скалярного добутку (кореляція) з яким буде максимальним. Тому для кожного вхідного сигналу визначається кореляція зі збереженими сигналами й викликається той образ, кореляція з яким максимальна. Таким чином, сигнал, відмінний від нуля, з'являється на виході нейрона i, номер якого визначається з умови

$$i = \arg\max_k (x_k^T x). \tag{5.33}$$

Приклад 5.7. Нехай задана лінійна ШНМ з матрицею із прикладу 5.3, на вхід якої надходить вектор $\tilde{x}$, що є спотвореним x_2

$$\tilde{x} = \begin{bmatrix} 0{,}3 \\ 0{,}9 \\ 0{,}0 \end{bmatrix} = x_2 + \begin{bmatrix} 0{,}3 \\ -0{,}1 \\ 0{,}0 \end{bmatrix} = x_2 - 0{,}18x_1 - 0{,}1x_2 + 0{,}24x_3.$$

Цьому $\tilde{x}$ відповідає вихідний сигнал

$$\tilde{y} = W\tilde{x} = \begin{bmatrix} 0{,}06 \\ 0{,}72 \\ 1{,}14 \end{bmatrix} = y_2^* + \begin{bmatrix} 0{,}06 \\ -0{,}28 \\ 0{,}14 \end{bmatrix} = y_2^* - 0{,}18y_1^* - 0{,}1y_2^* + 0{,}24y_3^*.$$

Пронормувавши компоненти векторів $\tilde{x}$ і $\tilde{y}$, визначимо відповідну кореляцію:

$\tilde{x}$	x_1	x_2	x_3	$\tilde{y}$	y_1^*	y_2^*	y_3^*
	−0,19	0,95	0,25		0,21	0,51	0,33

Таким чином, мережа сприймає образ $\tilde{x}$ як x_2, вихідний сигнал найбільш сильно корелює з y_2^*.

Оскільки сигнал завади Δy_p значно слабкіше, ніж асоційований y_p, правильне розв'язання можна отримати автоматично, якщо всі компоненти вихідного сигналу перед їхньою видачею пропустити через оператор, що реалізує деякий поріг S. Якщо поріг обраний правильно, то тільки найбільш вагомі компоненти з'являться на виході мережі. Це основна ідея використання в лінійному асоціаторі граничних елементів. При використанні порога номер активованого нейрона визначається з модифікованої умови (5.33)

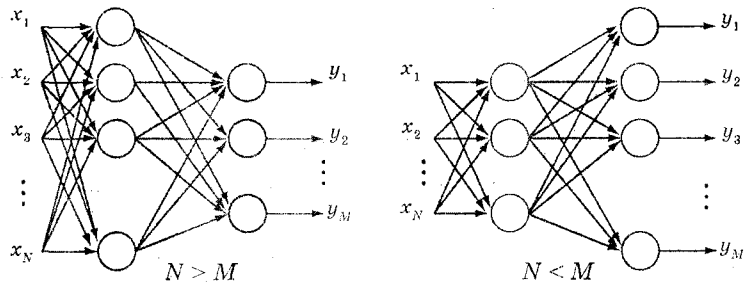
$$i = \arg(\max_k x_k^T x - S_k), \tag{5.34}$$

де S_k — заданий поріг.

5.3. Гетероасоціативна пам'ять

Гетероасоціативна пам'ять (ГАП) є мережею прямого поширення, що зберігає пари $\{x_p, y_p\}$, причому в загальному випадку $x_p \neq y_p$. Асоційовані пари x_p й y_p $(p = \overline{1,P})$ є векторами розмірності $N \times 1$ й $M \times 1$ відповідно (рис. 5.1), причому $N \neq M$. Схеми, що реалізують ГАП, наведені на рис. 5.3.

Мережу, зображену на рис. 5.3, *а*, застосовують у процесі розв'язання задач розпізнавання й прогнозування, коли необхідно визначити, до якого класу відноситься пропонований образ, а мережу на рис. 5.3, *б* — під час розв'язання задач відновлення, коли за номером класу (деяким кодом) необхідно відновити образ.



а *б*

Рис. 5.3. Гетероасоціативна пам'ять

Для навчання мережі використовується кожний з наведених вище алгоритмів.

Приклад 5.8. Розглянемо мережу, наведену на рис. 5.3, *а* з $N = 5$, $M = 3$ і граничною (релейною) функцією активації, на неї подаються такі пари, що навчають, які представлені біполярними сигналами:

$$\boldsymbol{x}_1^T = (\ 1\ -1\ -1\ \ 1\ \ 1); \quad (\boldsymbol{y}_1^*)^T = (\ 1\ \ 1\ -1);$$
$$\boldsymbol{x}_2^T = (\ 1\ \ 1\ \ 1\ -1\ -1); \quad (\boldsymbol{y}_2^*)^T = (\ 1\ -1\ \ 1);$$
$$\boldsymbol{x}_3^T = (-1\ \ 1\ \ 1\ \ 1\ \ 1); \quad (\boldsymbol{y}_3^*)^T = (-1\ \ 1\ \ 1).$$

При використанні порогової функції активації компоненти вектора вихідного сигналу мережі обчислюються в такий спосіб:

$$y_j = \operatorname{sgn}\left(\sum_{i=1}^{N} w_{ij}x_i\right) = \begin{cases} 1 & \text{при} \quad \sum_{i=1}^{N} w_{ij}x_i \geq 0; \\ -1 & \text{при} \quad \sum_{i=1}^{N} w_{ij}x_i < 0. \end{cases}$$

Зазначимо, що вхідні образи не є ні ортогональними, ні нормалізованими. Незважаючи на це, вагова матриця, що обчислюється за правилом Гебба, дає правильний результат. Обчислимо відповідно до (5.5) частки вагової матриці, прийнявши $\gamma = 1$

$$W_1 = \boldsymbol{y}_1^* \boldsymbol{x}_1^T = \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} \begin{pmatrix} 1 & -1 & -1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & 1 \\ -1 & 1 & 1 & -1 & -1 \end{pmatrix};$$

$$W_2 = \boldsymbol{y}_2^* \boldsymbol{x}_2^T = \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & -1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 & -1 \end{pmatrix};$$

$$W_3 = \boldsymbol{y}_3^* \boldsymbol{x}_3^T = \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} -1 & 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Скориставшись (5.6), отримуємо

$$W = \sum_{i=1}^{3} W_i = \begin{pmatrix} 3 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 3 & 3 \\ -1 & 3 & 3 & -1 & -1 \end{pmatrix}.$$

Для знаходження асоційованих образів з урахуванням того, що використовується порогова функція активації, обчислимо

$$W\boldsymbol{x}_1 = \begin{pmatrix} 3 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 3 & 3 \\ -1 & 3 & 3 & -1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ 7 \\ -9 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} = \boldsymbol{y}_1.$$

Аналогічно отримаємо, що $W\boldsymbol{x}_2 = \boldsymbol{y}_2$, $W\boldsymbol{x}_3 = \boldsymbol{y}_3$. Отже, мережа навчилася, тобто правильно створила й відтворила асоціації $\boldsymbol{y}_1 = \boldsymbol{y}_1^*$, $\boldsymbol{y}_2 = \boldsymbol{y}_2^*$, $\boldsymbol{y}_3 = \boldsymbol{y}_3^*$. Подивимося, що відбудеться, якщо на навчену мережу подається спотворений образ.

Нехай за якихось причин спотворився образ $\boldsymbol{x}_2$, тобто замість $\boldsymbol{x}_2$ на вхід надходить образ $\tilde{\boldsymbol{x}}_2^T = (1\ \ 1\ \ \underline{-1}\ \ -1\ \ -1)$ (спотворений біт підкреслений). У цьому випадку

$$W\boldsymbol{x}_2 = \begin{pmatrix} 3 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 3 & 3 \\ -1 & 3 & 3 & -1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 5 \\ -7 \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix} = \boldsymbol{y}_2.$$

Таким чином, мережа правильно відновила спотворений образ.

5.4. Автоасоціативна пам'ять

Автоасоціативна пам'ять (ААП) є окремим випадком гетероасоціативної при $N = M$ (рис. 5.4).

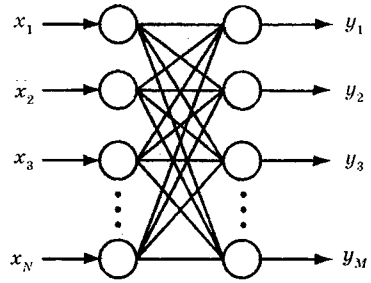


Рис. 5.4. Автоасоціативна пам'ять

Вона зберігає подані пари (образи) $(\boldsymbol{x}, \boldsymbol{y}^*)$, що навчають, з $\boldsymbol{y}^* = \boldsymbol{x}$. Завдання, розв'язуване цією пам'яттю, полягає у відновленні деякого зашумленого або спотвореного образу $\boldsymbol{x}_p$, а не у створенні його асоціації з іншим образом. Це є результатом одношарової структури асоціативної пам'яті, у якій вектор вихідних сигналів з'являється на виходах тих самих нейронів, на які надходить вектор вхідних сигналів. Дана пам'ять має всі властивості ГАП і забезпечує точне відтворення спотвореного образу при нормалізованих (ортонормованих) вхідних сигналах. Ємність пам'яті, тобто максимальне число збережених образів, дорівнює N. На практиці ж звичайно використовується число ортогональних збережених образів, менше, ніж N, оскільки при N асоціаціях кожен стан мережі є стійким, і мережа втрачає здатність відновлювати спотворені образи.

Як й у випадку з ГАП, навчання мережі здійснюється за допомогою кожного з розглянутих вище методів.

Приклад 5.9. Розглянемо ААП, зображену на рис. 5.4 з $N = 5$, що має порогову функцію активації. Нехай на вхід мережі подаються образи

$$\boldsymbol{x}_1^T = (-1\ -1\ -1\ -1\ \ 1);$$
$$\boldsymbol{x}_2^T = (\ 1\ -1\ \ 1\ -1\ \ 1);$$
$$\boldsymbol{x}_3^T = (-1\ -1\ \ 1\ \ 1\ -1);$$
$$\boldsymbol{x}_4^T = (-1\ -1\ -1\ \ 1\ \ 1).$$

Скориставшись тим, що вагові матриці для образів $\boldsymbol{x}_i$ ($i = 1, 2, 3, 4$) визначаються за формулами $W_i = \boldsymbol{x}_i \boldsymbol{x}_i^T$, отримуємо

$$W_1 = \boldsymbol{x}_1 \boldsymbol{x}_1^T = \begin{pmatrix} 1 & 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & -1 & 1 \end{pmatrix};$$

$$W_2 = \boldsymbol{x}_2 \boldsymbol{x}_2^T = \begin{pmatrix} 1 & -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & 1 \end{pmatrix};$$

$$W_3 = \boldsymbol{x}_3 \boldsymbol{x}_3^T = \begin{pmatrix} 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 & 1 \\ -1 & -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 \end{pmatrix};$$

$$W_4 = \boldsymbol{x}_4 \boldsymbol{x}_4^T = \begin{pmatrix} 1 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 \end{pmatrix}.$$

Результуючу вагову матрицю мережі W обчислюємо за формулою (4.6)

$$W = \sum_{i=1}^{4} W_i = \begin{pmatrix} 4 & 2 & 2 & -2 & 0 \\ 2 & 4 & 0 & 0 & -2 \\ 2 & 0 & 4 & 0 & -2 \\ -2 & 0 & 0 & 4 & -2 \\ 0 & -2 & -2 & -2 & 4 \end{pmatrix}.$$

Для визначення будь-якого образу, збереженого в пам'яті, обчислюємо добуток $W\boldsymbol{x}_i$ ($i = 1, 2, 3, 4$). Наприклад, при поданні $\boldsymbol{x}_3$ маємо

$$W\boldsymbol{x}_3 = \begin{pmatrix} 4 & 2 & 2 & -2 & 0 \\ 2 & 4 & 0 & 0 & -2 \\ 2 & 0 & 4 & 0 & -2 \\ -2 & 0 & 0 & 4 & -2 \\ 0 & -2 & -2 & -2 & 4 \end{pmatrix} \begin{pmatrix} -1 \\ -1 \\ 1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} -6 \\ -4 \\ 4 \\ 6 \\ -6 \end{pmatrix};$$

$$\boldsymbol{y}_3^T = \operatorname{sgn}\left(W\boldsymbol{x}_3\right)^T = \begin{pmatrix} -1 & -1 & 1 & 1 & -1 \end{pmatrix},$$

тобто мережа правильно відтворила збережений образ $\boldsymbol{x}_3$. Аналогічні результати здобуваються і для інших образів.

Нехай на вхід мережі поданий спотворений у четвертому біті образ $\boldsymbol{x}_3$

$$\tilde{\boldsymbol{x}}_3^T = \begin{pmatrix} -1 & -1 & 1 & \underline{\underline{-1}} & -1 \end{pmatrix}.$$

Застосовуючи матрицю W до $\tilde{\boldsymbol{x}}_3$, отримуємо

$$W\tilde{\boldsymbol{x}}_3 = \begin{pmatrix} 4 & 2 & 2 & -2 & 0 \\ 2 & 4 & 0 & 0 & -2 \\ 2 & 0 & 4 & 0 & -2 \\ -2 & 0 & 0 & 4 & -2 \\ 0 & -2 & -2 & -2 & 4 \end{pmatrix} \begin{pmatrix} -1 \\ -1 \\ 1 \\ -1 \\ -1 \end{pmatrix} = \begin{pmatrix} -2 \\ -4 \\ 4 \\ 0 \\ -2 \end{pmatrix};$$

$$\tilde{\boldsymbol{y}}_3^T = \operatorname{sgn}\left(W\tilde{\boldsymbol{x}}_3^T\right) = \begin{pmatrix} -1 & -1 & 1 & 1 & -1 \end{pmatrix}.$$

Отже, мережа правильно відновила спотворений образ.

5.5. Двоспрямована асоціативна пам'ять

Тип мереж, що називається *двоспрямованою асоціативною пам'яттю* (ДАП) (*BAM — bidirectional associative memory*), запропоновано і досліджено *Б. Коско* [84, 85]. Мережа ДАП являє собою двошарову гетероасоціативну пам'ять, у якій обидва шари є і вхідними, і вихідними (рис. 5.5). Образи, що подаються на входи $\boldsymbol{x}$, зображені N-вимірними векторами, на входи $\boldsymbol{y}$ — M-вимірними.

Зв'язок між обома шарами є двоспрямованим (звідси й назва мережі), тобто сигнали можуть проходити в обох напрямках. Як і для будь-якого типу асоціативної пам'яті, асоціації формуються у ваговій матриці W, що має розмірність $N \times M$.

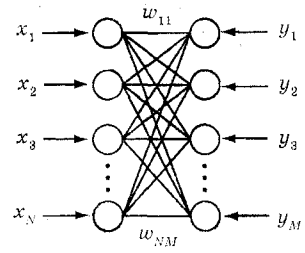


Рис. 5.5. Двоспрямована асоціативна пам'ять

Таким чином, ємність ДАП не може перевищувати меншого із чисел N або M. Більш того, реально вона менше меншого із цих чисел і визначається числом лінійно незалежних рядків (стовпців) матриці W. Елемент матриці w_{ij} $\left(i = \overline{1,N};\ j = \overline{1,M}\right)$, що зв'язує i-й нейрон образу $\boldsymbol{x}$ з j-м нейроном образу $\boldsymbol{y}$, використовується під час аналізу сигналів, переданих в обох напрямках. Сигнали можуть бути як дискретними, поданими у двійковій формі (бінарні або біполярні), так і неперервними, що реалізують дискретні й неперервні ДАП. Відповідно до цього активаційні функції нейронів можуть вибиратися релейними або сигмоїдальними. Нагадаємо, що сигмоїдальні функції з величиною λ, близькою до одиниці, описують нейрони із плавною й неперервною реакцією, багато в чому аналогічною реакції їхніх біологічних прототипів. За умови використання релейної активаційної функції у дискретній ДАП, як показав Коско, більш ефективним є біполярне кодування сигналів.

Можлива *синхронна* й *асинхронна* робота ДАП. Дана мережа має багато спільного з ART і мережею Гопфілда. Як і мережа Гопфілда, робота ДАП може бути проаналізована з енергетичної точки зору. Ввівши поняття двоспрямованої стійкості мережі й узагальнивши теорему Коена — Гроссберґа, Коско довів, що енергія ДАП є обмеженою й під час роботи мережі не зростає. Збіжність мережі до стійкого стану, при якому стани нейронів не змінюються, характеризує її рух до енергетичного мінімуму, що відповідає певному асоційованому образу. Якщо ж кількість збережених образів перевищує максимально припустиме, мережа видає помилковий сигнал.

Для навчання мережі, що полягає у визначенні вагової матриці W, застосовуються розглянуті вище методи. Наприклад, при використанні правила Гебба (5.4) матриця W матиме вигляд (5.6).

Для відтворення образу на входи x або y ДАП подається сигнал й обчислюється реакція мережі на протилежній стороні (наприклад, при подачі сигналу x за формулою (5.2) визначається y_j). Потім сигнал, що є добутком матриці W^T на отриманий вектор реакції, посилається у зворотному напрямку. Цей процес повторюється доти, поки мережа не опиниться в усталеному стані. При пороговій функції активації компоненти вихідного сигналу (для нашого прикладу y_j) у момент часу $(k+1)$ визначаються за формулою

$$y_j(k+1)=\begin{cases}+1, & \text{якщо } w_{ij}x_i > \theta_j;\\ y_j(k), & \text{якщо } |w_{ij}x_i| \le \theta_j;\\ -1, & \text{якщо } w_{ij}x_i < -\theta_j;\end{cases}$$

де θ_j — значення порога j-го нейрона шару y.

Аналогічно обчислюється вихідний сигнал на стороні мережі x, якщо вхідний сигнал надходить із боку y.

Приклад 5.10. Розглянемо застосування правила Гебба (4.4) з $\gamma = 1$ для навчання ДАП. Нехай мережі подаються пари $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, що навчають, $i = 1, 2$

$\boldsymbol{x}_1^T = (-1\ -1\ \ 1\ -1\ \ 1\ \ 1)$; $\quad \boldsymbol{y}_1^T = (-1\ \ 1\ \ 1\ -1)$;

$\boldsymbol{x}_2^T = (\ 1\ -1\ \ 1\ -1\ -1\ \ 1)$; $\quad \boldsymbol{y}_2^T = (-1\ -1\ \ 1\ \ 1)$.

Обчислимо матриці W_1 й W_2 відповідно до (5.5)

$$W_1 = \boldsymbol{y}_1\boldsymbol{x}_1^T = \begin{pmatrix} 1 & 1 & -1 & 1 & -1 & -1\\ -1 & -1 & 1 & -1 & 1 & 1\\ -1 & -1 & 1 & -1 & 1 & 1\\ 1 & 1 & -1 & 1 & -1 & -1\end{pmatrix};$$

$$W_2 = \boldsymbol{y}_2\boldsymbol{x}_2^T = \begin{pmatrix} -1 & 1 & -1 & 1 & 1 & -1\\ -1 & 1 & -1 & 1 & 1 & -1\\ 1 & -1 & 1 & -1 & -1 & 1\\ 1 & -1 & 1 & -1 & -1 & 1\end{pmatrix}.$$

Тоді

$$W = W_1 + W_2 = \begin{pmatrix} 0 & 2 & -2 & 2 & 0 & -2\\ -2 & 0 & 0 & 0 & 2 & 0\\ 0 & -2 & 2 & -2 & 0 & 2\\ 2 & 0 & 0 & 0 & -2 & 0\end{pmatrix}.$$

При подачі сигналу $\boldsymbol{x}_1$ з урахуванням порогової функції активації отримуємо

$$W\boldsymbol{x}_1 = \begin{pmatrix} 0 & 2 & -2 & 2 & 0 & -2\\ -2 & 0 & 0 & 0 & 2 & 0\\ 0 & -2 & 2 & -2 & 0 & 2\\ 2 & 0 & 0 & 0 & -2 & 0\end{pmatrix}\begin{pmatrix}-1\\-1\\1\\-1\\1\\1\end{pmatrix} = \begin{pmatrix}-8\\4\\8\\-4\end{pmatrix} \Rightarrow \begin{pmatrix}-1\\1\\1\\-1\end{pmatrix},$$

тобто вихідний сигнал збігається з $\boldsymbol{y}_1$.

Подаємо отриманий сигнал у зворотному напрямку, тобто на ДАП з обчисленою ваговою матрицею W подаємо сигнал y_1

$$W^T\boldsymbol{y}_1 = \begin{pmatrix} 0 & -2 & 0 & 2\\ 2 & 0 & -2 & 0\\ -2 & 0 & 2 & 0\\ 2 & 0 & -2 & 0\\ 0 & 2 & 0 & -2\\ -2 & 0 & 2 & 0\end{pmatrix}\begin{pmatrix}-1\\1\\1\\-1\end{pmatrix} = \begin{pmatrix}-4\\-4\\4\\-4\\4\\4\end{pmatrix} \Rightarrow \begin{pmatrix}-1\\-1\\1\\-1\\1\\1\end{pmatrix},$$

що відповідає безпомилковому відтворенню образу $\boldsymbol{x}_1$.

Аналогічні обчислення для $\boldsymbol{x}_2$ й $\boldsymbol{y}_2$ дають

$$W\boldsymbol{x}_2 = \begin{pmatrix} 0 & 2 & -2 & 2 & 0 & -2\\ -2 & 0 & 0 & 0 & 2 & 0\\ 0 & -2 & 2 & -2 & 0 & 2\\ 2 & 0 & 0 & 0 & -2 & 0\end{pmatrix}\begin{pmatrix}1\\-1\\1\\-1\\-1\\1\end{pmatrix} = \begin{pmatrix}-8\\-4\\8\\4\end{pmatrix} \Rightarrow \begin{pmatrix}-1\\-1\\1\\1\end{pmatrix} = \boldsymbol{y}_2;$$

$$W^T\boldsymbol{y}_2 = \begin{pmatrix} 0 & -2 & 0 & 2\\ 2 & 0 & -2 & 0\\ -2 & 0 & 2 & 0\\ 2 & 0 & -2 & 0\\ 0 & 2 & 0 & -2\\ -2 & 0 & 2 & 0\end{pmatrix}\begin{pmatrix}-1\\-1\\1\\1\end{pmatrix} = \begin{pmatrix}4\\-4\\4\\-4\\-4\\4\end{pmatrix} \Rightarrow \begin{pmatrix}1\\-1\\1\\-1\\-1\\1\end{pmatrix} = \boldsymbol{x}_2.$$

Таким чином, мережа правильно видає асоційовані образи.

Нехай тепер замість $\boldsymbol{x}_1$ мережі подано спотворений образ

$$\tilde{\boldsymbol{x}}_1^T = (-1 \quad -1 \quad \underline{-1} \quad -1 \quad 1 \quad 1).$$

Тоді

$$W\tilde{\boldsymbol{x}}_1 = \begin{pmatrix} 0 & 2 & -2 & 2 & 0 & -2 \\ -2 & 0 & 0 & 0 & 2 & 0 \\ 0 & -2 & 2 & -2 & 0 & 2 \\ 2 & 0 & 0 & 0 & -2 & 0 \end{pmatrix} \begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -4 \\ 4 \\ 4 \\ -4 \end{pmatrix} \Rightarrow \begin{pmatrix} -1 \\ 1 \\ 1 \\ -1 \end{pmatrix} = \tilde{\boldsymbol{y}}_1 = \boldsymbol{y}_1;$$

$$W^T\tilde{\boldsymbol{y}}_1 = \begin{pmatrix} 0 & -2 & 0 & 2 \\ 2 & 0 & -2 & 0 \\ -2 & 0 & 2 & 0 \\ 2 & 0 & -2 & 0 \\ 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 \end{pmatrix} \begin{pmatrix} -1 \\ 1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} -4 \\ -4 \\ 4 \\ -4 \\ 4 \\ 4 \end{pmatrix} \Rightarrow \begin{pmatrix} -1 \\ -1 \\ 1 \\ -1 \\ 1 \\ 1 \end{pmatrix} = \boldsymbol{x}_1,$$

тобто мережа правильно відновила спотворений образ.

Контрольні запитання та завдання

1. Дайте визначення асоціативної пам'яті.
2. Які існують алгоритми навчання асоціативної пам'яті? У чому їхні особливості? Дайте їхню порівняльну характеристику.
3. У чому відмінність між авто- і гетероасоціативною пам'яттю?
4. Що являє собою двоспрямована асоціативна пам'ять?

6. МЕРЕЖА ГОПФІЛДА

Новий сплеск досліджень у галузі ШНМ викликаний статтями американського фізика *Дж. Гопфілда* [23, 24], що вийшли в 1982 й 1984 рр., у яких розвивалися ідеї, засновані на результатах досліджень Маккаллоха й Піттса, Ґроссберґа, Андерсона, Когонена й інших учених. Гопфілд використав свою мережу для моделювання спінових слідів. Під цим розуміють на увазі матеріали, атоми яких мають магнітний диполь. Під час моделювання кожному диполю відповідав нейрон, орієнтація диполя в магнітному полі здійснювалася порушенням відповідного нейрона, а мережа описувала магнітні взаємодії полів. Розглянуті в його статтях застосування містять *асоціативну пам'ять*, *аналого-цифрове перетворення*, *розв'язання задачі оптимізації* (*про комівояжера*). Підкреслена практична спрямованість досліджень Гопфілда призвела до того, що внаслідок його тривалого співробітництва з AT&T Bell Laboratories уже в 1987 р. створено нейромережевий чип, заснований на розробленій ним мережі.

6.1. Модель Гопфілда

Розроблена Гопфілдом модель асинхронної ШНМ має такі ознаки:

1. Мережа є одношаровою й містить N нейронів, число яких є одночасно числом входів і виходів мережі.
2. Кожен нейрон мережі пов'язаний з усіма іншими нейронами, а також має один вхід, на який подається вхідний сигнал.
3. Жоден нейрон не має власного зворотного зв'язку ($w_{ij} = 0$).
4. Ваги мережі є симетричними, тобто вага зв'язку між i-м й j-м нейронами дорівнює вагам зв'язку між j-м й i-м нейронами $w_{ij} = w_{ji}$.
5. Кожен нейрон має порогову функцію активації.
6. Вхідними є двійкові сигнали.

Структурну схему мережі Гопфілда наведено на рис. 6.1.

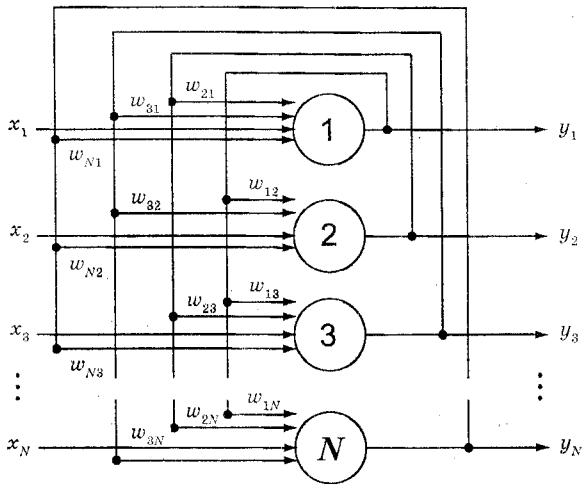


Рис. 6.1. Структурна схема мережі Гопфілда

Слід зазначити, що оскільки «мережева» архітектура мозку містить зворотні зв'язки, мережа Гопфілда щодо цього найбільше відповідає природному процесу обробки інформації.

Внаслідок симетрії мережі Гопфілда іноді використовують інше її подання, наведене на рис. 6.2 для мережі з чотирма нейронами.

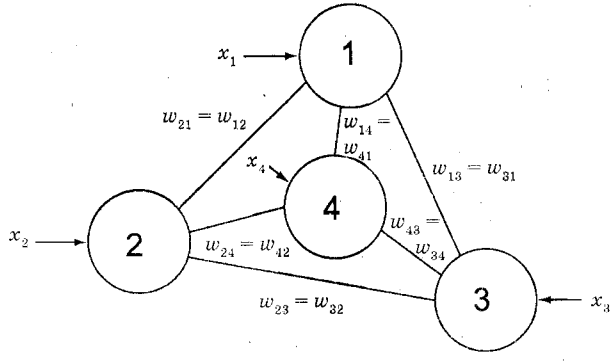


Рис. 6.2. Графічне зображення мережі Гопфілда

Дана мережа не використовує, строго кажучи, ані навчання з учителем, ані навчання без учителя. Вагові коефіцієнти в ній розраховуються тільки перед початком функціонування мережі на основі інформації про оброблювані дані, і все навчання мережі зводиться саме до цього розрахунку. З одного боку, подання апріорної інформації можна розцінювати як допомогу вчителя, а з іншого — мережа фактично просто запам'ятовує образи до того, як на її вхід надходять реальні дані, і не може змінювати свою поведінку, тому говорити про зворотний зв'язок із учителем не доводиться.

Слід зазначити, що в мережах зі зворотними зв'язками стан нейронів обчислюється доти, поки вони не виявляться сталими, не змінюваними згодом. Можна сказати, що мережа Гопфілда за певних умов збігається до сталого стану за кінцевий час.

Мережі подаються P образів, описуваних N-вимірними (N — кількість нейронів) векторами $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)^T$ з компонентами, що приймають двійкові значення, наприклад «−1» й «+1». Позначимо вектор вхідних сигналів, що описує p-й образ, як $\boldsymbol{x}^p$ ($p = \overline{1,P}$). Коли мережа розпізнає поданий образ, вона сформує вектор вихідних сигналів, компоненти якого співпадають з відповідними компонентами образу, тобто $\boldsymbol{y} = \boldsymbol{x}^p$, де $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)^T$ — вектор вихідних сигналів мережі. В іншому випадку вихідний сигнал не збігатиметься з жодним із поданих.

Розрізнюють три стадії (фази) функціонування мережі:

— ініціалізація;

— подання вхідного образу;

— обчислення стану нейронів.

Ініціалізація. На цій стадії мережі встановлюються такі значення вагових коефіцієнтів:

$$w_{ij} = \begin{cases} \sum\limits_{p=0}^{P} x_i^p x_j^p, & i \neq j; \\ 0, & i = j. \end{cases} \tag{6.1}$$

Тут $x^p{}_i$ й $x^p{}_j$ — i-та й j-та компоненти вектора $\boldsymbol{x}^p$.

Цю стадію можна розглядати як стадію навчання мережі, після завершення якої вона здатна правильно відтворювати подані їй образи. Кажуть, що мережа здатна до самоасоціативної роботи.

Подання мережі вхідного образу фактично здійснюється безпосередньою початковою (нульовою) установкою компонентів вихідних сигналів

$$y_i(0) = x_i, \quad i = \overline{1,N}, \tag{6.2}$$

тому позначення на схемі мережі вхідних сигналів у явному вигляді носить суто умовний характер.

Обчислення станів нейронів. За формулою

$$z_j(k+1) = \sum_{i=1}^{N} w_{ij} y_i(k) + x_j \qquad (6.3)$$

визначаються послідовні стани нейронів, на підставі чого розраховуються відповідні значення вихідних сигналів

$$y_j(k+1) = f_a\left(z_j(k+1)\right) = \begin{cases} 1, & \text{якщо } z_j(k+1) > \theta_j; \\ 0, & \text{якщо } z_j(k+1) < \theta_j; \\ y_i(k) & \text{в інших випадках,} \end{cases} \qquad (6.4)$$

де $f_a = (\cdot)$ — порогова активаційна функція; θ_j — заданий поріг.

Якщо вихідні значення змінилися, то за формулами (6.3) і (6.4) знову розраховуються стани й вихідні сигнали мережі, якщо не змінилися — робота мережі завершується (мережа перебуває в стійкому стані). У цьому випадку на виходах мережі сформувався вектор, що якнайкраще відповідає поданому образу.

Приклад 6.1. Розглянемо, до чого призведе порушення умов $w_{ii} = 0$. Припустимо, що є мережа, яка складається з двох нейронів, з матрицею ваг

$$\boldsymbol{w} = \begin{vmatrix} -1 & -1 \\ -1 & -1 \end{vmatrix}$$

і порогами $\theta_i = -1$ $(i = 1, 2)$. Нехай у деякий момент часу k при подачі сигналу $x_i = 0$ $(i = 1, 2)$ нейрони перебували в стані $z_i(k) = 1$ $(i = 1, 2)$, тобто на виході мережі були сигнали (*див.* (6.4)) $y_i = 1$ $(i = 1, 2)$. Цю мережу наведено на рис. 6.3.

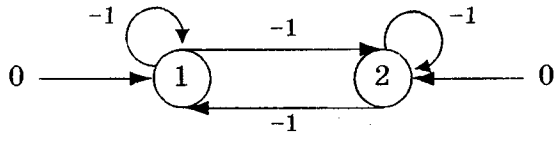


Рис. 6.3. Порушення умови $w_{ii} = 0$

У наступний, $(k + 1)$-й, момент часу, стани й вихідні сигнали нейронів, що обчислюють за формулами (6.3) і (6.4) відповідно, дорівнюватимуть

$$z_i(k+1) = (-1)\cdot 1 + (-1)\cdot 1 + 0 = -2;$$

$$y_i(k+1) = 0, \; i = 1, 2.$$

Аналогічно для $(k + 2)$-го моменти часу отримаємо

$$z_i(k+2) = (-1)\cdot 0 + (-1)\cdot 0 + 0 = 0;$$

$$y_i(k+1) = 1, \; i = 1, 2.$$

Далі процес повторюється.

Таким чином, при порушенні умови $w_{ii} = 0$ мережа осцилює, тобто не досягає стійкого стану.

Приклад 6.2. Розглянемо наслідки порушення умови $w_{ij} = w_{ji}$ на прикладі мережі, що складається із двох нейронів і має матрицю ваг

$$\boldsymbol{w} = \begin{vmatrix} 0 & -2 \\ 2 & 0 \end{vmatrix}$$

і поріг $\theta_i = 0$.

Нехай у k-й момент часу при подачі на вхід нейронів сигналів $x_1 = 1$, $x_2 = -1$ на їхніх виходах були сигнали $y_1 = 1$, $y_2 = 0$. Цю мережу зображено на рис. 6.4.

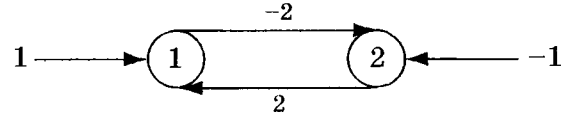


Рис. 6.4. Порушення умови $w_{ij} = w_{ji}$

Скориставшись формулами (6.3) і (6.4), визначимо, що в наступні моменти часу стану мережі й значення її вихідних сигналів зміняться у такий спосіб:

$k + 1$

$z_1(k+1) = -2\cdot 0 + 1 = 1$ $\quad y_1(k+1) = 1$

$z_2(k+1) = 2\cdot 1 - 1 = 1$ $\quad y_2(k+1) = 1$

$k + 2$

$z_1(k+2) = -2\cdot 1 + 1 = -1$ $\quad y_1(k+2) = 0$

$z_2(k+2) = 2\cdot 1 - 1 = 1$ $\quad y_2(k+2) = 1$

$k + 3$

$z_1(k+3) = -2\cdot 1 + 1 = -1$ $\quad y_1(k+3) = 0$

$z_2(k+3) = 2\cdot 0 - 1 = -1$ $\quad y_2(k+3) = 0$

$k+4$

$z_1(k+4) = -2\cdot 0 + 1 = 1 \qquad y_1(k+4) = 1$

$z_2(k+4) = 2\cdot 0 - 1 = -1 \qquad y_2(k+4) = 0.$

Мережа повернулася в початковий стан, і далі процес повторюється.

Таким чином, при порушенні умови $w_{ij} = w_{ji}$ мережа також не досягає стійкого стану, тобто осцилює.

Активація мережі Гопфілда може здійснюватися *асинхронно*, коли на кожному такті змінює свій стан тільки одним випадковим способом обраний нейрон, і *синхронно*, коли всі нейрони змінюють свій стан одночасно.

6.2. Навчання в мережі Гопфілда

Робота мережі Гопфілда може бути пояснена термінами енергетичного ландшафту. Є ландшафт, що являє собою гористу місцевість, на вершині якої перебуває куля. Потім куля котиться по схилу, поки не зупиниться в якій-небудь низині (западині). Ці низини відбивають стійкі стани мережі, і кожна з них відповідає певному поданому образу (образу навчання). У такому поданні куля, що має велику потенційну енергію, котиться в низину з меншою потенційною енергією, досягаючи локального мінімуму. Щоб знову опинитися в початковому стані, вона повинна здійснити у фізичному значенні роботу, тобто витратити енергію. Таким чином, робота мережі Гопфілда може бути охарактеризована деякою енергетичною функцією.

Якщо в процесі аналізу персептрона вибір такої функції (функціонала) особливих ускладнень не викликав, оскільки відомі реальні й бажані значення виходів мережі, то в цьому випадку ситуація дещо інша. По-перше, структура мережі Гопфілда не дозволяє заздалегідь визначити шлях вирішення, тобто бажану або необхідну послідовність станів нейронів. По-друге, наявність зворотних зв'язків призводить до того, що виходи мережі в будь-який момент часу надають набір входів. Крім того, має бути врахована відсутність у нейронів власних зворотних зв'язків. Зазначені особливості мережі відображуються в енергетичній функції вигляду

$$E = -\frac{1}{2}\sum_i \sum_{j\neq i} w_{ij} y_i y_j - \sum_i y_i x_i + \sum_i y_i \theta_i, \qquad (6.5)$$

де θ_i — значення порога i-го нейрона; w_{ij} — ваги зв'язку i-го й j-го нейронів; x_i — зовнішній вхідний сигнал i-го нейрона (постійна величина для спостережуваного моменту часу); y_i — значення вихідного сигналу i-го нейрона.

Гопфілд показав, що при активізації мережі функція (6.5) не зростає й досягає локального мінімуму в деякому сталому стані. А оскільки кількість таких стійких станів обмежена, мережа за певних умов досягає одного з них за кінцеву кількість ітерацій. При цьому, як уже зазначалося, низини енергетичного ландшафту (енергетичні мінімуми) відповідають збереженим образам. Щоб мережа Гопфілда правильно класифікувала образи, ці низини не мають перекриватися.

6.2.1. Накопичення образів у мережі Гопфілда

Функціонал (6.5) можна розглядати як критерій помилки (чим далі від вирішення, тим його значення більше). Щоб зберігати пропоновані мережі образи, необхідно мінімізувати значення енергетичної функції для кожного з них, тобто визначити локальні мінімуми енергетичного ландшафту. Однак додавання нових образів не має знищувати вже наявну інформацію. А оскільки вся інформація щодо образів, які зберігаються, міститься у ваговій матриці, задача навчання зводиться до визначення значень ваг, що доставляють мінімум енергетичній функції (6.5).

При мінімізації даного функціонала необхідно враховувати таке. Як видно з (6.5), мінімум даного виразу перебуває у від'ємній області. Для виконання умови $\sum_i y_i \theta_i < 0$ необхідно, щоб компоненти y_i й θ_i постійно мали протилежні знаки. При апріорно заданому θ_i це неможливо, тому доцільно прийняти $\theta_i = 0$, що призводить до критерію вигляду

$$E = -\frac{1}{2}\sum_i \sum_{j\neq i} w_{ij} y_i y_j - \sum_i x_i y_i. \qquad (6.6)$$

Позначимо за аналогією з вищевикладеним x_m, y_m — компоненти m-го образу. Тоді матрицю ваг w_{ij} можна розбити на дві підматриці, перша з яких (w^1_{ij}) відображає зв'язки всіх образів, крім m-го, а друга (w_{im}) — тільки m-й образ

$$E = -\frac{1}{2}\sum_i\sum_{j\neq i} w_{ij}^1 x_i x_j - \sum_{i\neq m} x_i y_i - \frac{1}{2}\sum_{i\neq m} w_{im} y_i y_m - x_m y_m. \qquad (6.7)$$

Перші два доданки (6.7) можна розглядати як збурення, що виражає загальний вплив усіх образів, крім m-го, на функціонал, а другі — як корисний сигнал. Отже, щоб мінімізувати функціонал (6.7) для заданого m-го образу, необхідно розглянути другий доданок, тобто

$$E_m = -\frac{1}{2}\sum_{i\neq m} w_{im} y_i y_m - x_m y_m. \qquad (6.8)$$

Мінімізація (6.8) еквівалентна максимізації

$$\sum_{i\neq m} w_{im} y_i y_m + x_m y_m. \qquad (6.9)$$

Розглянемо перший доданок. Оскільки $y_m \in \{-1, 1\}$, то завжди $(y_m)^2 > 0$. Якщо вибрати $w_{im} = y_i y_m$, то

$$\sum_{i\neq m} w_{im} y_i y_m = \sum_{i\neq m} (y_i)^2 (y_m)^2. \qquad (6.10)$$

Отже, вибір $w_{im} = y_i y_m$ забезпечує максимум функціоналові (6.9). Неважко побачити, що при нульових початкових умовах подача на вхід образу x призводить відповідно до (6.3) і (6.4) до появи на виході мережі вектора $y = x$. Цим пояснюється установка вагових коефіцієнтів на стадії ініціалізації у вигляді (6.1).

6.2.2. Виклик образу

Після того, як усі подані образи будуть запам'ятовані мережею (побудований енергетичний ландшафт), інформація про них може бути отримана шляхом мінімізації функціонала помилки на фазі навчання. Використання енергетичного ландшафту дозволяє визначити вплив окремої вершини на енергію, а мінімізація енергії — шлях переходу мережі в будь-який стійкий стан.

Нехай енергетична функція мережі Гопфілда в момент часу k визначається відповідно до (6.5) виразу

$$E(k) = -\frac{1}{2}\sum_i\sum_{j\neq i} w_{ij} x_i(k) x_j(k) + \sum_i x_i(k)\theta_i. \qquad (6.11)$$

Вхідні сигнали мережі в розглянутий момент часу не змінюються. Збудження нейронів мережі в момент часу $(k + 1)$ призводить до зміни її енергетичної функції, що обчислюється у такий спосіб:

$$\Delta E(k+1) = E(k+1) - E(k) = \left[-\frac{1}{2}\sum_i\sum_{j\neq i} w_{ij} x_i(k+1) x_j(k+1) + \sum_i x_i(k+1)\theta_i\right] - \left[-\frac{1}{2}\sum_i\sum_{j\neq i} w_{ij} x_i(k) x_j(k) + \sum_i x_i(k)\theta_i\right]. \qquad (6.12)$$

Припустимо, що в момент часу $(k + 1)$ змінюється лише стан одного m-го нейрона (з нульового переходить в одиничний або навпаки), все-таки інші нейрони свого стану не міняють. Внаслідок цього в (6.12) усі доданки з $i = m$ й $j = m$ однакові. Враховуючи те, що в мережі Гопфілда $w_{ij} = 0$ и $w_{im} = w_{mi}$ для $i = j \neq m$, вираз (6.12) можна переписати у такий спосіб:

$$\Delta E(k+1) = \left[-\sum_{j\neq m} w_{im} x_i(k+1)\, x_m(k+1) + \theta_m x_m(k+1)\right] - \left[-\sum_{i\neq m} w_{im} x_i(k)\, x_m(k) + \theta_m x_m(k)\right] \qquad (6.13)$$

або

$$\Delta E(k+1) = \left[-\sum_{i\neq m} w_{im} x_i(k+1) + \theta_m\right] x_m(k+1) - \left[-\sum_{i\neq m} w_{im} x_i(k) + \theta_m\right] x_m(k). \qquad (6.14)$$

Внаслідок того, що для всіх нейронів $i \neq m$ $x_i(k+1) = x_i(k)$, отримуємо

$$\Delta E(k+1) = \left[-\sum_{i\neq m} w_{im} x_i(k+1) + \theta_m\right] \cdot \left[x_m(k+1) - x_m(k)\right] = -\left[z_m(k+1) - \theta_m\right] \Delta x_m(k+1), \qquad (6.15)$$

де $\Delta x_m(k+1) = x_m(k+1) - x_m(k)$.

Під час аналізу (6.15) можливі два випадки.

1. $z_m(k+1) > \theta_m$.

Це означає, що m-й нейрон буде активований, тобто на попередньому такті він був пасивним. Таким чином, $\Delta x_m(k+1) > 0$ й $\Delta E(k+1) < 0$.

2. $z_m(k+1) < \theta_m$.

У цьому випадку нейрон перейде в нульовий стан з одиничного, тобто $\Delta x_m(k+1) < 0$, а оскільки й $z_m(k+1) - \theta_m < 0$, то знову $\Delta E(k+1) < 0$.

Отже, при збудженні m-го нейрона асинхронної мережі Гопфілда енергетична функція на кожному такті зменшується. Виклик образу з мережі відповідає проходженню через послідовність станів, кожен наступний із яких має більш низьку енергію порівняно з попереднім. Цей спад у стани з більш низьким енергетичним рівнем триває до досягнення мережею енергетичного мінімуму, тобто до відтворення мережею образу, якого викликано. Даний процес є аналогом градієнтного алгоритму, що збігається до локального екстремуму, і пояснює використовуваний у мережі механізм навчання (6.3) і (6.4).

Необхідно зазначити таке. Звичайно в літературі передбачається, що вхідний сигнал x_i діє короткий час й у розрахунках не враховується, тобто замість (6.5) розглядається функціонал

$$E = -\frac{1}{2}\sum_i \sum_{j \neq i} w_{ij} y_i y_j + \sum_i y_i \theta_i. \qquad (6.16)$$

Таким чином, алгоритм функціонування мережі Гопфілда полягає у такому.

1. *Навчання.*

Подаються образи x^p $(p = \overline{1,P})$. За формулою (6.1) обчислюються елементи w_{ij} матриці ваг і для активних нейронів (що перебувають у стані «+1») на основі їх поточного й попереднього стану обчислюється енергія кожного образу

$$E = -\frac{1}{2}\sum_{i=1}^{N} x_i^p(k+1)\sum_{j=1}^{N} w_{ij} x_j^p(k).$$

2. *Розпізнавання.*

Подаються образи, можливо, спотворені й за формулами (6.3), (6.4) обчислюються послідовні стани й значення вихідних сигналів нейронів до досягнення ними стійких станів.

Приклад 6.3. Розглянемо функціонування мережі Гопфілда, що складається з 7 нейронів і біполярної порогової функції активації з порогом $\theta = 0$, для навчання якої використовуються образи $x^1 = (1\ -1\ -1\ 1\ -1\ 1\ 1)^T$ і $x^2 = (-1\ -1\ 1\ -1\ 1\ -1\ 1)^T$ і яка має

розпізнати спотворений образ $x^3 = \tilde{x}^2 = (-1\ -1\ \underline{-1}\ -1\ 1\ -1\ 1)^T$ (спотворений біт підкреслений).

На рис. 6.5 наведена ця мережа й позначені деякі її зв'язки.

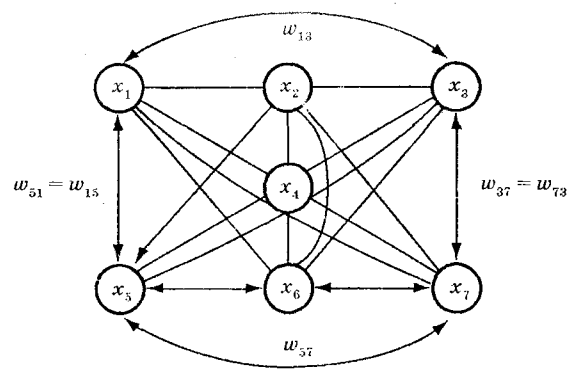


Рис. 6.5. Мережа Гопфілда, що складається з 7 нейронів

Елементи w_{ij} матриці ваг W розмірності 7×7, що має вигляд

$$W = W^1 + W^2 = \begin{pmatrix} w_{11}=0 & w_{12} & w_{13} & \dots & w_{17} \\ w_{21} & w_{22}=0 & w_{23} & \dots & w_{27} \\ \dots & \dots & \dots & \dots & \dots \\ w_{71} & w_{72} & w_{73} & \dots & w_{77}=0 \end{pmatrix},$$

де W^i — матриця ваг для i-го ($i = 1, 2$) образу, що навчає, визначаються за формулою (6.1)

$w_{12} = 1\cdot(-1) + (-1)\cdot(-1) = 0$; $\quad w_{13} = 1\cdot(-1) + (-1)\cdot 1 = -2$;

$w_{14} = 1\cdot 1 + (-1)\cdot(-1) = 2$; $\quad w_{15} = 1\cdot(-1) + (-1)\cdot 1 = -2$;

$w_{16} = 1\cdot 1 + (-1)\cdot(-1) = 2$; $\quad w_{17} = 1\cdot 1 + (-1)\cdot 1 = 0$ и т. д.

Після визначення всіх ваг матриця набуває вигляду

$$W = \begin{pmatrix} 0 & 0 & -2 & 2 & -2 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 \\ -2 & 0 & 0 & -2 & 2 & -2 & 0 \\ 2 & 0 & -2 & 0 & -2 & 2 & 0 \\ -2 & 0 & 2 & -2 & 0 & -2 & 0 \\ 2 & 0 & -2 & 2 & -2 & 0 & 0 \\ 0 & -2 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Обчислимо енергію кожного образу, беручи до уваги тільки активні нейрони. Для першого образу

$$E^1 = \sum_{i=1}^{7} E_i^1 = E_1^1 + E_4^1 + E_6^1 + E_7^1,$$

$$E_1^1 = -\frac{1}{2}\left[x_1(w_{13}x_3 + w_{14}x_4 + w_{15}x_5 + w_{16}x_6)\right] =$$

$$= -\frac{1}{2}\left[(-2)\cdot(-1) + 2\cdot 1 + (-2)\cdot(-1) + 2\cdot 1\right] = -4;$$

$$E_4^1 = -\frac{1}{2}\left[x_4(w_{41}x_1 + w_{43}x_3 + w_{45}x_5 + w_{46}x_6)\right] =$$

$$= -\frac{1}{2}\left[2\cdot 1 + (-2)\cdot(-1) + (-2)\cdot(-1) + 2\cdot 1\right] = -4;$$

$$E_6^1 = -\frac{1}{2}\left[x_6(w_{61}x_1 + w_{63}x_3 + w_{64}x_4 + w_{65}x_5)\right] =$$

$$= -\frac{1}{2}\left[2\cdot 1 + (-2)\cdot(-1) + 2\cdot 1 + (-2)\cdot(-1)\right] = -4;$$

$$E_7^1 = -\frac{1}{2}\left[x_7(w_{72}x_2)\right] = -\frac{1}{2}\left[(-2)\cdot(-1)\right] = -1.$$

Тоді $E^1 = -13$. Аналогічно отримуємо для другого образу

$$E^2 = \sum_{i=7}^{7} E_i^2 = E_3^2 + E_5^2 + E_7^2 = -9.$$

Розпізнавання образу $x^3 = \tilde{x}^2$ починаємо з визначення його енергії, що після проведення аналогічних обчислень дорівнюватиме

$$E^3 = \sum_{i=1}^{7} E_i^3 = E_5^3 + E_7^3 = -5.$$

За формулами (6.3) і (6.4) обчислюємо нові стани нейронів і відповідні значення вихідних сигналів, які з урахуванням біполярної активаційної функції дорівнюватимуть

$$z_j^3 = \sum_{i=1}^{7} w_{ij}x_i^3, \quad j = \overline{1,7}$$

$$z_1^3 = w_{31}x_3^3 + w_{41}x_4^3 + w_{51}x_5^3 + w_{61}x_6^3 + x_1^3 =$$

$$= (-2)\cdot(-1) + 2\cdot(-1) + (-2)\cdot 1 + 2\cdot(-1) - 1 = -5 < 0; \quad y_1^3 = -1;$$

$$z_2^3 = w_{72}x_7^3 + x_2^3 = (-2)\cdot 1 - 1 = -3 < 0; \quad y_2^3 = -1;$$

$$z_3^3 = w_{13}x_1^3 + w_{43}x_4^3 + w_{53}x_5^3 + w_{63}x_6^3 + x_3^3 =$$

$$= (-2)\cdot(-1) + (-2)\cdot(-1) + 2\cdot 1 + (-2)\cdot(-1) - 1 = 7 > 0; \quad y_3^3 = 1;$$

$$z_4^3 = w_{14}x_1^3 + w_{34}x_3^3 + w_{54}x_5^3 + w_{64}x_6^3 + x_4^3 =$$

$$= 2\cdot(-1) + (-2)\cdot(-1) + (-2)\cdot 1 + 2\cdot(-1) - 1 = -5 < 0; \quad y_4^3 = -1;$$

$$z_5^3 = w_{15}x_1^3 + w_{35}x_3^3 + w_{45}x_4^3 + w_{65}x_6^3 + x_5^3 =$$

$$= (-2)\cdot(-1) + 2\cdot(-1) + (-2)\cdot(-1) + (-2)\cdot(-1) + 1 = 5 > 0; \quad y_5^3 = 1;$$

$$z_6^3 = w_{16}x_1^3 + w_{36}x_3^3 + w_{46}x_4^3 + w_{56}x_5^3 + x_6^3 =$$

$$= 2\cdot(-1) + (-2)\cdot(-1) + 2\cdot(-1) + (-2)\cdot 1 - 1 = -5 < 0; \quad y_6^3 = -1;$$

$$z_7^3 = w_{27}x_2^3 + x_7^3 = (-2)\cdot(-1) + 1 = 3 > 0; \quad y_7^3 = 1.$$

Таким чином, після подання $x^3 = \tilde{x}^2$ мережа перейде в стійкий стан $(-1\ -1\ 1\ -1\ 1\ -1\ 1)^T$, що відповідає образу x^2 з енергією, як неважко переконатися, $E = -9$. Отже, мережа правильно розпізнала (відновила спотворений) образ.

6.3. Синхронна мережа Гопфілда

Як ми вже зазначали, у цій мережі нейрони змінюють свій стан одночасно, прагнучи перейти в деякі стійкі стани. Однак при цьому поняття стійкого стану має дещо інший зміст. Розглянемо це на прикладі.

Приклад 6.4. Нехай на виходах синхронної мережі Гопфілда, заданою матрицею ваг $w_{ii} = 0$, $w_{ij} = w_{ji}\ (i, j = \overline{1,3})$ і значеннями порогів $\theta_i = -1$, у деякий момент часу k при подачі на її входи вектора $x(k) = (000)^T$ з'являються одиничні сигнали $y(k) = (111)^T$. Тоді в наступний $(k+1)$-й момент часу нейрони перейдуть у нові стани, визначені відповідно до (6.3)

$$z_i(k+1) = 0 + (-1)\cdot 1 + (-1)\cdot 1 = -2,$$

і на їхніх виходах з'являться сигнали

$$x_i(k+1) = 0 \quad (i = \overline{1,3}).$$

На $(k+2)$-му такті сигнали $y_i(k+1)$ надійдуть на входи нейронів, переводячи їх у стани

$$z_i(k+2) = 0 + (-1)\cdot 0 + (-1)\cdot 0 = 0,$$

після чого на їхніх виходах з'являться сигнали $y_i(k+2) = 1$ $(i = \overline{1,3})$, тобто мережа перейшла в початковий стан. Неважко помітити, що мережа осцилює.

Що ж відбувається з енергією мережі? У момент часу k значення енергетичної функції (6.5) дорівнюватиме

$$E(k)=-\frac{6}{2}\left[(-1)\cdot 0\cdot 1\right]+3\left[0\cdot(-1)\right]=0,$$

а при переході в новий стан на $(k+1)$-му такті

$$E(k+1)=-\frac{6}{2}\left[(-1)\cdot 0\cdot 0\right]+3\left[0\cdot(-1)\right]=0.$$

Отже, хоча синхронна мережа осцилює, значення її енергетичної функції не змінюється.

Використання енергетичної інтерпретації дозволяє побачити основну відмінність між синхронним й асинхронним способами активації мережі Гопфілда. Дійсно, випадковий вибір вершини при асинхронній активації призводить до зміни шляху досягнення мережею свого стійкого стану, тобто до зміни послідовності аналізу мережею проміжних образів. Отже, асинхронна активація збільшує деяку невизначеність шляху переходу мережі з початкового в кінцевий стан. При синхронній активації всі вершини оновлюються разом, тому проміжні образи не змінюються. Крім того, мережа осцилює між двома різними станами. Обидва ці способи активації призводять до одного результату, тому звичайно їхній вибір не є принциповим.

6.4. Неперервна мережа Гопфілда

Неперервний варіант мережі Гопфілда є узагальненням дискретної мережі у випадку використання замість порогової функції активації неперервної. Звичайно в якості такої функції використовують синусоїдальну або функцію гіперболічного тангенса, а динаміку мережі описують у неперервному часі. У найбільш загальному вигляді такий опис, запропонований у роботах *М. Коена* й *С. Гроссберга* [22] і *Дж. Гопфілда* [24], може бути поданий у такий спосіб:

$$\tau_i\frac{dx_i}{dt}=-x_i+f_i\left(\sum_{j=1}^{n}w_{ij}x_j\right),\quad i=\overline{1,n},\qquad(6.17)$$

де τ_i — деяка стала часу; $f(\cdot)$ — функція активації виду

$$f(x)=\frac{1}{1+\exp(-x)}.$$

Система, динаміка якої описується рівнянням (6.17), може або прагнути до деякого стійкого стану, або перебувати в хаотичному русі, або осцилювати. Вибір симетричної матриці вагових коефіцієнтів дозволяє забезпечити рух системи до стійкого стану. При цьому збіжність гарантується *теоремою Коена — Гроссберга* [22]. Стійкі стани свідчать про те, що система перебуває в рівновазі й справедливо

$$\frac{dx_i}{dt}=0\ \text{для всіх } i.$$

У цьому випадку виходи системи визначаються в такий спосіб:

$$x_i=f_i\left(\sum_j w_{ij}x_j\right),\quad i=\overline{1,n},\qquad(6.18)$$

де x_i приймає дійсні значення з інтервалу [0, 1].

Динаміка неперервної мережі Гопфілда може бути описана за аналогією з (6.17) шляхом заміни змінних $\boldsymbol{x}$ на стани нейронів z_i $(i=\overline{1,n})$

$$z_i=\sum_j w_{ij}x_j,\quad i=\overline{1,n}.$$

У цьому випадку система диференціальних рівнянь, що описують мережу, набуває вигляду

$$\tau_i\frac{dz_i}{dt}=-z_i+\sum_{j=1}^{n}w_{ij}f(z_j),\quad i=1,2,\ldots,n,$$

а стійкі стани мережі можуть бути визначені у такий спосіб:

$$z_i=\sum_j w_{ij}f(z_j),\quad i=\overline{1,n}.$$

Як й у випадку дискретної мережі, аналіз неперервної мережі Гопфілда може бути проведений за допомогою енергетичної функції. Однак доведення збіжності для неперервної мережі загального вигляду є досить складним. Для аналізу неперервного випадку Гопфілд використав енергетичну функцію такого вигляду:

$$E=-\frac{1}{2}\sum_i\sum_j w_{ij}x_ix_j+\sum_i\int_o^{x_i}f^{-1}(x)dx.\qquad(6.19)$$

Можна показати, що при переході в стійкий стан, як і в дискретному випадку, енергетична функція мережі зменшується.

6.5. Асоціативна мережа BSB

На відміну від розглянутих вище статичних схем асоціативної пам'яті, ця мережа реалізує *динамічну АП*. Ця мережа, що є рекурсивною автоасоціативною пам'яттю, запропонована *Дж. Андерсоном* й ін. [86]. Свою назву *Brain-State-in-a-Box* (*BSB*) вона отримала через те, що її стан обмежений гіперкубом [+1, −1] із центром на початку координат.

Структуру мережі BSB наведено на рис. 6.6.

Структура й принцип роботи мережі BSB аналогічні структурі й принципу роботи мережі Гопфілда. Як і мережа Гопфілда, ця мережа є одношаровою. Основна відмінність у тому, що в мережі BSB допускається наявність власних зворотних зв'язків нейронів, тобто знімається обмеження $w_{ii} = 0$, і крім того, можлива відсутність зв'язків між окремими нейронами, тобто допускається $w_{ij} = 0$.

Як активаційні у мережі використовуються функції вигляду (2.8), тому j-й вихідний сигнал у момент часу $(k + 1)$ визначається у такий спосіб:

$$y_j(k+1) = \begin{cases} +1, & \text{якщо } x_j(k) > 1; \\ y_j(k), & \text{якщо } |x_j(k)| \le 1; \\ -1, & \text{якщо } x_j(k) < 1, \end{cases} \qquad (6.20)$$

де $x_j(k)$ — j-й вхідний сигнал у момент часу k

$$x_j(k) = y_j(k) + \beta \sum_{i=1}^{N} w_{ij} x_i(k). \qquad (6.21)$$

Тут β — мала додатна величина, що називається параметром зворотного зв'язку.

Наявність додатного зворотного зв'язку призводить до того, що подані на входи мережі сигнали підсилюються доти, поки всі нейрони не ввійдуть у насичення. Характерною рисою даної мережі є можливість роботи з аналоговими сигналами, які переводяться в бінарні форми.

У початковому стані мережа перебуває всередині одиничного гіперкуба, обмеженого вхідними біполярними сигналами. Динамічні властивості мережі забезпечує рекурсивна зміна її стану в напрямку ребер (граней) гіперкуба. Ці грані визначають атрактори нелінійної динамічної системи, що відповідають збереженим образам. Поява на вході мережі будь-якого сигналу призводить до того, що мережа з часом виявиться в одному з таких стійких станів, що визначає даний вхідний образ (сигнал) і що перебуває в деякій вершині гіперкуба.

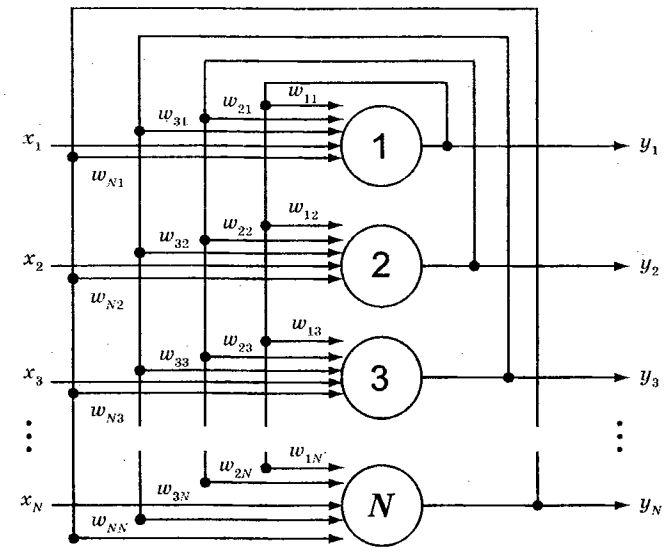


Рис. 6.6. Структура мережі BSB

Для навчання мережі використовується або правило Гебба, або дельта-правило. Як початкові значення ваг вибирають малі випадкові величини $w_{ij} \le 1$. Метою навчання є знаходження матриці синаптичних ваг W, що забезпечує існування N стійких станів, таких, що

$$\boldsymbol{y}(k) = W\boldsymbol{x}(k), \quad k = \overline{1,N}. \qquad (6.22)$$

Для аналізу стійкості мережі BSB використовується той самий підхід, що й під час аналізу дискретної мережі Гопфілда. Локальні мінімуми енергетичної функції досягаються на грані гіперкуба й визначають стійкі стани мережі, що відповідають різним накопиченим образам.

BSB-модель, будучи асоціативною пам'яттю, вирішує по суті завдання кластеризації заданого масиву даних, оскільки вершини гіперкуба в процесі обробки інформації діють як точкові атрактори з вираженими областями притягання, які ділять N-вимірний простір ознак на відповідну множину добре визначених областей. Якщо центри-прототипи кластерів відомі заздалегідь, їх можна

поєднати з вершинами гіперкуба, після чого BSB-модель може працювати взагалі без навчання тільки за рахунок автозбуждения, що викликається додатним зворотним зв'язком. При цьому пропоновані мережі образи автоматично «стягуватимуться» до найближчих вершин гіперкуба — центрів кластерів.

Контрольні запитання

1. Що являє собою модель Гопфілда? Яке значення має ця модель для подальшого розвитку ШНМ?
2. До чого призведе порушення обмеження на вибір елементів матриці ваг?
3. Поясніть суть алгоритму навчання мережі Гопфілда.
4. У чому особливості синхронної мережі Гопфілда?
5. Що являє собою неперервна мережа Гопфілда?
6. Що являє собою мережа BSB?
7. У чому відмінність мережі BSB від мережі Гопфілда?
8. Як відбувається навчання мережі BSB?

7. СИНЕРГЕТИЧНИЙ КОМП'ЮТЕР

Дана мережа запропонована засновником *синергетики Х. Хакеном* [87, 88] як ШНМ, функціонування якої може бути пояснене із синергетичної точки зору. Ця мережа має багато спільного з мережею Гопфілда й більш того, може розглядатися як один з її варіантів, що працює з аналоговими сигналами. Цей комп'ютер відноситься до асоціаторів, а збережені в ньому образи можуть бути інтерпретовані як локальні енергетичні мінімуми. Відмінність його від мережі Гопфілда полягає в тому, що його нейрони не містять нелінійностей, оскільки нелінійне перетворення здійснюється спеціальним модулем. Принцип побудови *«синергетичного комп'ютера»* наведено на рис. 7.1.

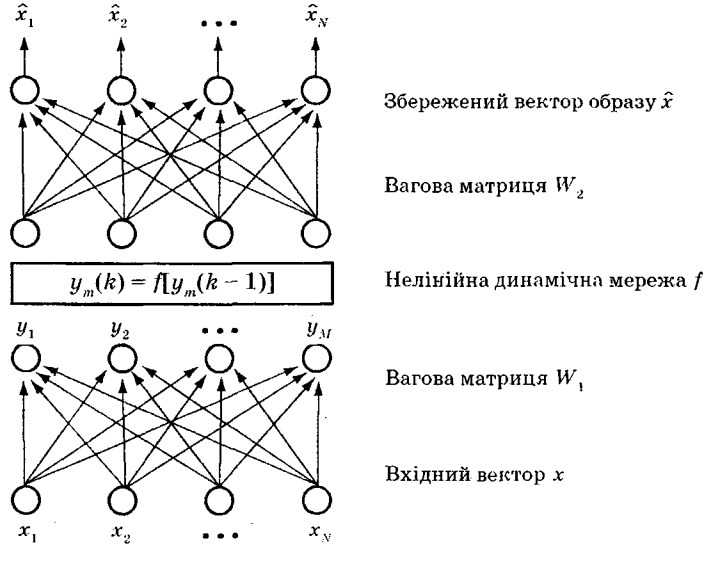


Рис. 7.1. Синергетичний комп'ютер

ШНМ складається із вхідного шару, на який надходить зашумлений або спотворений N-вимірний образ, що відповідає одному з M збережених $\hat{x}_m$, $m = \overline{1,M}$. Вхідний вектор $\boldsymbol{x}$ перетвориться нейронним шаром з ваговою матрицею W_1. Вихідним сигналом шару є M-вимірний вектор $\boldsymbol{y}$, кількість компонент якого збігається із кількістю збережених образів. Цей вектор $\boldsymbol{y}$ подається на модуль, описуваний системою нелінійних диференціальних рівнянь, що після деякої кількості ітерацій обчислює нові значення компонент y_i вектора $\boldsymbol{y}$. Вектор $\boldsymbol{y}$ зі знову обчисленими компонентами подається на наступний шар через матрицю W_2. Вихідним сигналом останнього шару $\hat{\boldsymbol{x}}$ є N-вимірний вектор, що відповідає образу $\boldsymbol{x}_M$. Отже, асоціативний виклик накопиченого образу здійснюється подачею на вхід вектора $\boldsymbol{x}$.

Вибір значень елементів матриць W_1 й W_2 дуже простий — вони визначаються безпосередньо векторами збережених образів за формулою

$$W_1^T = W_2 = \left| \hat{\boldsymbol{x}}_1 \hat{\boldsymbol{x}}_2 \ldots \hat{\boldsymbol{x}}_m \right|, \quad m = \overline{1,M}. \tag{7.1}$$

Щоб простіше було зрозуміти роботу мережі, розглянемо окремі етапи обчислення вихідного образу.

Вихідний сигнал першого шару визначається як

$$y = W_1 \boldsymbol{x}, \tag{7.2}$$

тобто є скалярним добутком векторів $\hat{\boldsymbol{x}}_m$ і $\boldsymbol{x}$, що тим більше, чим ближче розташовані ці вектори. Таким чином, компоненти вихідного сигналу $\boldsymbol{y}$ характеризують міру подібності накопиченого образу $\hat{\boldsymbol{x}}_m$ пропонованому $\boldsymbol{x}$, і якщо пропонований образ $\boldsymbol{x}$ буде найближчим до збереженого $\hat{\boldsymbol{x}}_{m^*}$, компонента m^* вихідного сигналу $\boldsymbol{y}_{m^*}$ матиме найбільше значення. Компоненти y_m $(m = \overline{1,M})$ є початковими значеннями компонентів вектора $\boldsymbol{y}$, що обчислюють за такою рекурсивною формулою:

$$y_m(k) = y_m(k-1)\left[\alpha_k - \beta \sum_{\substack{i=1 \\ i \neq m}}^{M} y_i^2(k-1) + (\beta - 1) y_m^2(k-1)\right]. \tag{7.3}$$

Це рівняння можна розглядати як рівняння руху тіла в потенціальному полі, на яке діє сила, що задається похідною потенціальної функції по координатах тіла.

Вибираючи відповідну потенційну функцію у вигляді

$$V(y_m) = -\frac{1}{2}\sum_{i=1}^{M} \alpha_i y_i^2 + \frac{\beta}{4}\sum_{\substack{i=1 \\ i \neq m}}^{M} y_m^2 y_i^2 + \frac{1}{4}\sum_{i=1}^{M} y_i^4 \tag{7.4}$$

й обчислюючи рівняння руху як

$$y_m(k) = -\frac{\partial V}{\partial y_m}, \tag{7.5}$$

можна отримати (7.3). Рівняння (7.5) відповідає градієнтному методу пошуку локального мінімуму. Можна показати, що найбільший компонент y_m збігається до одиниці, а інші — до нуля. Таким чином, завдання (7.3) полягає у тому, щоб збільшити значення найбільшого компонента й зменшити значення всіх інших. Після рекурсивних обчислень відповідно до (7.3) тільки один компонент вихідного вектора дорівнюватиме одиниці, а інші — нулю. Цей вектор передається з ваговою матрицею W_2 на інший шар (вихідний)

$$\hat{\boldsymbol{x}} = W_2 \boldsymbol{y}.$$

Оскільки компоненти матриці W_2 обрані відповідно до (7.1), то при надходженні на цей шар сигналу $\boldsymbol{y}$, всі компоненти якого, крім однієї, m^*-ї, рівної одиниці, дорівнюють нулю, відповідний вихідний вектор $\hat{\boldsymbol{x}}$ точно відповідатиме m^*-му образу $\boldsymbol{x}_{m^*}$, що навчає.

Якщо образи $\hat{\boldsymbol{x}}_m$, що навчають, ортонормовані в M-вимірному просторі, то подання вхідного вектора $\boldsymbol{x}$ й обчислення $\boldsymbol{y}$ за формулою (7.2) є не що інше, як обчислення проекції $\boldsymbol{x}$ на M базисних векторів. А рекурсивне обчислення (7.3) і збіжність до 1 означає, що застосування (7.3) повертає $\boldsymbol{x}$ до того базисного вектора, проекція на який у нього найбільша. Після рекурсивних обчислень вектор $\boldsymbol{y}$ ідентичний цьому базисному вектору. У зв'язку з тим, що сигнал $\boldsymbol{y}$ містить тільки одну одиницю, а всі інші компоненти нульові, дана система завжди зводить вхідний вектор $\boldsymbol{x}$ до однієї з базисних функцій. Оскільки система (7.3) випливає з потенційної функції (7.4), то базисні вектори $(1,0\ldots)^T$, $(0,1,\ldots 0)^T$ тощо є мінімумами потенціальній функції. А у зв'язку з тим, що базисні вектори шляхом координатної трансформації з'являються у вигляді навчальних образів $\hat{\boldsymbol{x}}_m$, це означає, що дані образи є мінімумами потенційної функції (7.4). Отже, як й у мережі Гопфілда, для образів «синергетичного комп'ютера», що навчають, можна задати енергетичну (потенційну) функцію, мінімуми якої відповідають цим образам. Пропонований мережі вхідний образ $\boldsymbol{x}$ збігається до того мінімуму, в області притягання (околі) якого він перебуває.

На завершення слід зазначити, що «синергетичним комп'ютером» ця мережа була названа тому, що основним принципом синергетики є встановлення (аналіз) стаціонарних станів динамічної системи, що залежать від початкових умов й областей притягання стаціонарних точок системи. Тому система рівнянь $y_m(k) = f[y_m(k-1)]$ на рис. 7.1 описує синергетический модуль ШНМ. Перетворений першим шаром за допомогою вагової матриці W_1 вхідний сигнал $\boldsymbol{x}$ подається в системі координат, базисні вектори якої є мінімумами синергетичної системи, що характеризується потенційною функцією (7.4) й описується співвідношеннями (7.3) і (7.4). Рекурсивне обчислення (7.3) дозволяє досягти локального мінімуму, тобто базисного вектора, в області притягання якого перебуває трансформований вхідний вектор $\boldsymbol{x}$. Другий шар мережі за допомогою матриці W_2 здійснює зворотну трансформацію базисного вектора в область образів, що навчають, і викликає асоціативно вектор образів $\boldsymbol{x}_{m^*}$, що відповідає мінімуму потенційної функції.

Слід зазначити, що кількість збережених образів M не може бути більше розмірності образу N.

На відміну від мережі Гопфілда, «синергетичний комп'ютер» не має нестійких станів (*spurious states*). Це пояснюється тим, що його динамічний модуль завжди збігається до одного з базисних векторів. А оскільки завжди в процесі навчання тільки один компонент y дорівнюватиме одиниці, а всі інші нулю, то й завжди забезпечується виклик одного з образів $\hat{\boldsymbol{x}}_m$ шляхом обчислення $\hat{\boldsymbol{x}}_m = W_2 y$.

Контрольні запитання

1. Якою є структура синергетичного комп'ютера?
2. Як відбувається навчання синергетичного комп'ютера?
3. У чому відмінність синергетичного комп'ютера від мережі Гопфілда?

8. МЕРЕЖА ХЕММІНГА

Істотного скорочення витрат на пам'ять й обсяг необхідних обчислень можна досягти, якщо замість збереженого в пам'яті образу мережа видає тільки його номер. Таке спрощення мережі досягнуто в описаній *Р. Ліппманом мережі Хеммінга* [82].

Ця мережа заснована на обчисленні відстані Хеммінга між двійковими сигналами однакової довжини. *Відстанню Хеммінга* між двома векторами, поданими у двійковій формі, називається число позицій (бітів), у яких вони відрізняються. Мережа Хеммінга реалізує паралельне обчислення відстаней Хеммінга між пропонованим вхідним вектором і збереженими в пам'яті образами.

Мережа Хеммінга складається із двох шарів (рис. 8.1).

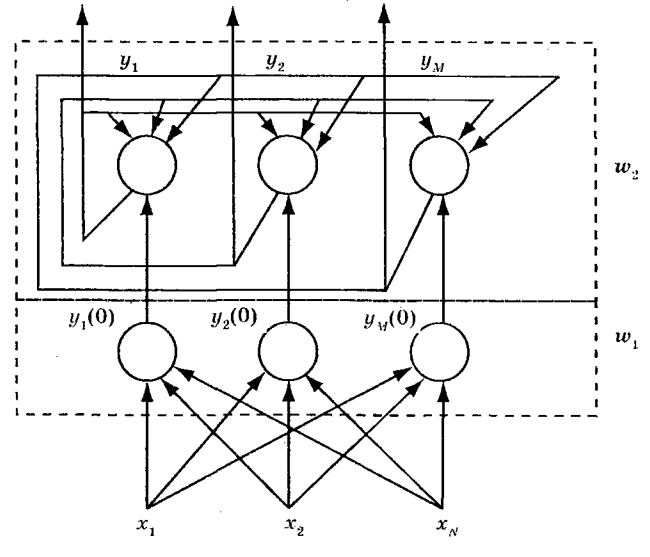


Рис. 8.1. Мережа Хеммінга

На відміну від мережі Гопфілда, ємність мережі Хеммінга визначається не розмірністю вхідного сигналу N, а кількістю нейронів у вихідному шарі M.

На вхід мережі надходять двійкові біполярні сигнали, тобто сигнали, що приймають значення «+1» або «−1». Звичайні двійкові сигнали, що приймають значення «+1» й «0», перетворяться таким чином, щоб значення «0» перетворилося в «−1». З цією метою компонента нового вектора $\tilde{x}_i$ перераховуються за формулою

$$\tilde{x}_i = 2x_i - 1.$$

Задача мережі полягає в тому, щоб при поданні їй деякого невідомого двійкового вхідного образу $\boldsymbol{x}$ вона видавала асоційований з ним збережений у мережі образ $\boldsymbol{x}_k^*$. Отже, вона працює за принципом *асоціативної пам'яті*. При цьому на виході мережі з'являється не сам асоціативний образ, а його номер k у вигляді сигналу, що виникає на виході k-го нейрона вихідного шару і має максимальне значення, у той час як на виходах інших нейронів сигнали будуть слабкими.

Мережа Хеммінга складається із двох шарів, перший з яких є мережею прямого поширення й характеризується ваговою матрицею W_1

$$W_1^T = \left|\boldsymbol{x}_1^* \boldsymbol{x}_2^* \ldots \boldsymbol{x}_M^*\right|. \tag{8.1}$$

Вихідний сигнал цього шару реалізується відповідно до формули

$$\boldsymbol{y} = 0{,}5(W_1 \boldsymbol{x} + (M\ M \ldots M)^T), \tag{8.2}$$

де M — кількість збережених у пам'яті образів.

Отже, кожен компонент вихідного сигналу обчислюється за формулою

$$y_i = 0{,}5\left[\left(\boldsymbol{x}_i^*\right)^T \boldsymbol{x} + M\right]. \tag{8.3}$$

Як уже було зазначено, компоненти векторів $\boldsymbol{x}_i^*$ й $\boldsymbol{x}$ можуть приймати значення «+1» й «−1», причому знаки компонентів можуть збігатися або не збігатися. Якщо кількість компонентів (біт), які не збіглися, дорівнює H, то H є *хеммінговою відстанню* цих векторів. Отже, при H неспівпадаючих компонентах кількість співпадаючих компонент дорівнює $M - H$, а скалярний добуток $(\boldsymbol{x}_i^*)^T \boldsymbol{x}$ дорівнюватиме

$$(\boldsymbol{x}_i^*)^T \boldsymbol{x} = (M - H) - H = M - 2H. \tag{8.4}$$

Підстановка (8.4) в (8.3) дає

$$y_i = 0{,}5(M - 2H + M) = M - H, \tag{8.5}$$

тобто вихідний сигнал першого шару характеризує кількість співпадаючих розрядів у векторів (хеммінгова відстань) $\boldsymbol{x}_i^*$ й $\boldsymbol{x}$. Якщо $H = 0$, то $y_i = M$, тобто вектори $\boldsymbol{x}_i^*$ й $\boldsymbol{x}$ повністю збігаються. При $H = M$ вектори $\boldsymbol{x}_i^*$ й $\boldsymbol{x}$ повністю не збігаються.

Вихідний сигнал першого шару надходить на входи M нейронів другого шару, рекурсивного, вагова матриця W_2 якого має вигляд

$$W_2 = \begin{vmatrix} 1 & -\varepsilon & \ldots & -\varepsilon \\ -\varepsilon & 1 & \ldots & -\varepsilon \\ \ldots & \ldots & \ldots & \ldots \\ -\varepsilon & -\varepsilon & \ldots & 1 \end{vmatrix}, \tag{8.6}$$

де $\varepsilon < \dfrac{1}{M}$.

Нейрони цього шару, що називаються мережею MAXNET, мають зворотні зв'язки, що й забезпечує рекурсивність шару. Тому вихідний сигнал всієї мережі описується співвідношенням

$$\boldsymbol{y}(k) = W_2 \boldsymbol{y}(k-1). \tag{8.7}$$

Отже, вихідний сигнал першого шару показує кількість бітів збігу пропонованого образу й збереженого, а вихідний сигнал всієї мережі, що з'являється після декількох ітерацій на виході одного нейрона другого шару, визначає номер збереженого образу, що найменше відрізняється від поданого.

Приклад 8.1. Нехай мережа призначена для зберігання двох образів ($M = 2$), тобто $\boldsymbol{y} = (y_1\, y_2)^T$. Компоненти y_1 й y_2 обчислюються відповідно до (8.7)

$$\begin{aligned} y_1(k) &= y_1(k-1) - \varepsilon y_2(k-1); \\ y_2(k) &= y_2(k-1) - \varepsilon y_1(k-1). \end{aligned} \tag{8.8}$$

Початкові умови визначаються з початкових умов вектора виходу першого шару $y^T(0) = [y_1(0)\ y_2(0)]$, скільки він містить кількість бітів подібності поданих образів.

Розв'язок (8.8) має вигляд

$$y_1(k) = \frac{y_1(0) - y_2(0)}{2}(1+\varepsilon)^k + \frac{y_1(0) + y_2(0)}{2}(1-\varepsilon)^k,$$

$$y_2(k) = \frac{y_2(0) - y_1(0)}{2}(1+\varepsilon)^k + \frac{y_2(0) + y_1(0)}{2}(1-\varepsilon)^k.$$

Оскільки $\varepsilon > 0$, другі доданки обох рівнянь із зростанням k збігаються до нуля. Якщо $y_1(0) > y_2(0)$, тобто $y_1(0)$ більше подібний на поданий образ, то $y_1(k) \to \infty$, а $y_2(k) \to -\infty$. Якщо ж $y_1(0) < y_2(0)$, то навпаки. Це справедливо для будь-якої кількості образів M. Отже, вихід нейрона, вхідний сигнал якого найбільшою мірою відповідає поданому образу, із зростанням часу прагне до нескінченності. Виходи ж інших нейронів прямують до $-\infty$.

Приклад 8.2. Розглянемо мережу Хеммінга, що має три вхідних нейрони ($N = 3$) і два вихідних ($M = 2$), що навчається на векторах $x^*(1) = (1\ \ -1\ \ -1)^T$, $x^*(2) = (1\ \ 1\ \ -1)^T$. Нехай $\varepsilon = 0{,}3$.
Тоді, відповідно до (8.1), матриця ваг матиме вигляд

$$w_1 = \begin{pmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{pmatrix},$$

а компоненти вихідного сигналу першого шару при подачі сигналу x_1 визначаються з (8.3) (з урахуванням того, що $M = 2$)

$$y(0) = 0{,}5\left[\begin{pmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix} + \begin{pmatrix} 2 \\ 2 \end{pmatrix}\right] = \begin{pmatrix} 2{,}5 \\ 1{,}5 \end{pmatrix}.$$

Після першої ітерації, відповідно до (8.7), на виході отримуємо

$$y(1) = \begin{pmatrix} 1 & -0{,}3 \\ -0{,}3 & 1 \end{pmatrix}\begin{pmatrix} 2{,}5 \\ 1{,}5 \end{pmatrix} = \begin{pmatrix} 2{,}05 \\ 0{,}75 \end{pmatrix},$$

після другої —

$$y(2) = \begin{pmatrix} 1 & -0{,}3 \\ -0{,}3 & 1 \end{pmatrix}\begin{pmatrix} 2{,}05 \\ 0{,}75 \end{pmatrix} = \begin{pmatrix} 1{,}825 \\ 0{,}135 \end{pmatrix}.$$

Далі отримуємо, що перша компонента вектора вихідного сигналу $y(i)$ на кожній наступній ітерації збільшується, а друга — зменшується (*див.* приклад 8.1).
Аналогічно при подачі на вхід мережі $x^*(2)$ зростатиме друга компонента і зменшуватиметься перша.
Нехай на вхід мережі надходить образ $x(3) = (\underline{-1}\ \ -1\ \ -1)^T$, що є спотвореним $x^*(1)$ (біт, у якому відбулося спотворення, підкреслений).

Тоді, відповідно до (8.5), на виході першого шару з'явиться сигнал

$$\tilde{y}(0) = 0{,}5\left[\begin{pmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{pmatrix}\begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix} + \begin{pmatrix} 2 \\ 2 \end{pmatrix}\right] = \begin{pmatrix} 1{,}5 \\ 0{,}5 \end{pmatrix}.$$

Після першої ітерації в рекурсивному шарі отримуємо

$$\tilde{y}(1) = \begin{pmatrix} 1 & -0{,}3 \\ -0{,}3 & 1 \end{pmatrix}\begin{pmatrix} 1{,}5 \\ 0{,}5 \end{pmatrix} = \begin{pmatrix} 1{,}35 \\ 0{,}05 \end{pmatrix},$$

після другої —

$$\tilde{y}(2) = \begin{pmatrix} 1 & -0{,}3 \\ -0{,}3 & 1 \end{pmatrix}\begin{pmatrix} 1{,}35 \\ 0{,}05 \end{pmatrix} = \begin{pmatrix} 1{,}335 \\ -0{,}355 \end{pmatrix} \text{ і т. д.},$$

тобто перша компонента вихідного сигналу мережі збільшується, а друга зменшується. Отже, поданий образ x_3 мережа розпізнає як $x^*(1)$, тобто правильно. Зазначимо також, що відстані Хеммінга між образом $x(3)$ і образами $x^*(1)$ і $x^*(2)$ дорівнюватимуть відповідно $H_1 = 1$ й $H_2 = 2$. Отже, мережа сприйняла образ $x(3)$ як накопичений нею раніше образ x_1^*, що знаходиться від $x(3)$ на мінімальній хеммінговій відстані.

На завершення зазначимо, що при одній і тій самій ємності мереж мережа Хеммінга міститиме значно менше з'єднань між нейронами порівняно з мережею Гопфілда (у мережі Хеммінга залежність між кількістю нейронів і кількістю з'єднань лінійна, а в мережі Гопфілда — квадратична).

Контрольні запитання

1. Що являє собою модель мережі Хеммінга?
2. Як задається параметр ε у мережі Хеммінга? На що впливає вибір його величини?
3. У чому основна відмінність мережі Хеммінга від мережі Гопфілда?

9. ДИНАМІЧНІ РЕКУРСИВНІ ШНМ

Відмінність розглянутих у цьому розділі мереж від мереж Гопфілда (розділ 6) полягає в такому:

1. Рекурсивна мережа може мати кілька шарів.
2. Нейрони можуть мати власні зворотні зв'язки ($w_{ij} \neq 0$).
3. Матриця ваг може бути несиметричною.
4. Може здійснюватися контрольоване навчання шляхом використання алгоритму зворотного поширення.

Наявність зворотних зв'язків між нейронами різних шарів, включаючи й нейрони вихідного шару, забезпечує *динамічним рекурсивним мережам* (ДРМ) додаткові позитивні властивості, які не можуть бути досягнуті в статичних багатошарових мережах прямого поширення. До таких властивостей відноситься, наприклад, можливість роботи з образами, параметри яких змінюються в часі.

9.1. Структура ДРМ

Структуру деякої ДРМ зображено на рис. 9.1.

У даній мережі вихідними нейронами можуть бути будь-які (наприклад, на рис. 9.1 один нейрон прихованого шару є також вихідним).

Сигнали, що надходять на входи нейронів вхідного шару в певний момент часу $t > 0$, перетворяться останніми за допомогою відповідних функцій активації $f_a(x, w)$ і в наступний момент часу передаються по наявних зв'язках нейронам, які їх також перетворять. Далі перетворений сигнал по прямих і зворотних зв'язках надходить на входи нейронів і процес повторюється. Наявність зворотних зв'язків призводить до того, що залежно від значень вхідного сигналу й вагових параметрів мережа може:

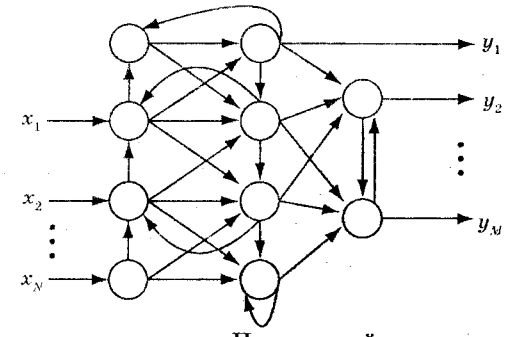


Рис. 9.1. Приклад структури ДРМ

а) досягти деякого стійкого стану;

б) осцилювати, тобто періодично повторювати значення вихідних сигналів;

в) хаотично змінювати свій стан.

Отже, динаміка ДРМ аналогічна динаміці мережі Гопфілда (*див.* розділ 6).

Наявність зворотних зв'язків у цій мережі не дозволяє використовувати для її опису настільки прості співвідношення, які застосовувалися в розглянутих раніше мережах. Тому їхня динаміка, як і динаміка мереж Гопфілда, що є окремим випадком ДРМ, описується нелінійними диференціальними (у неперервному випадку) або різницевими (у дискретному випадку) рівняннями першого порядку.

Властивості ДРМ вивчалися в роботах [89–93].

9.2. Неперервні ДРМ

Динаміка i-го нейрона неперервної ДРМ описується рівняннями, аналогічними рівнянню (6.17):

$$\tau_i \frac{dx_i(t)}{dt} = -x_i(t) + f_i\left(\sum_{j=1}^{L} w_{ij} x_j(t) + v_i\right), \quad i = \overline{1,L}, \tag{9.1}$$

де τ_i — стала часу i-го нейрона; $x_i(t)$ — стан i-го нейрона в момент часу t; $f_i(t)$ — нелінійна функція активації; v_i — зовнішній вхідний сигнал i-го нейрона; L — кількість нейронів у мережі.

Елементи w_{ij} вагової матриці W визначаються шляхом розв'язання рівнянь

$$\frac{dx(t)}{dt}=0. \tag{9.2}$$

Залежно від виду матриці W розрізнюють три типи мереж:
- симетрична матриця ваг з нульовими діагональними елементами, що описує ДРМ типу мережі Гопфілда;
- трикутна матриця W, що характеризує ДРМ прямого поширення без зворотних зв'язків;
- вагова матриця довільного виду, що характеризує ДРМ загального виду.

В останньому випадку мережа залежно від значень її параметрів може або досягати деякого стійкого стану, або осцилювати, або хаотично змінювати свій стан. Зокрема, для досягнення стійкого стану вагові коефіцієнти w_{ij} мають задовольняти такій умові:

$$\sum_i \sum_j w_{ij}^2 < (\max_i |f_i'|)^{-2}, \tag{9.3}$$

де $f_i' = \dfrac{df}{dx}$ — похідна функції активації.

9.3. Дискретна ДРМ

Рівняння, що описує динаміку дискретної ДРМ, здобувається із (9.1) шляхом переходу до кінцевих різниць, а елементи вагової матриці є розв'язком системи рівнянь

$$x_i(k+1)=f\left(\sum_{j=1}^{L} w_{ij}x_j(k)+v_i\right). \tag{9.4}$$

При $\Delta t \to 0$ поведінка неперервної й дискретної ДРМ ідентична.

У загальному ж випадку поведінка обох мереж навіть при однакових значеннях елементів вагової матриці різна (наприклад, якщо одна досягає стійкого стану, то інша може осцилювати).

Від опису (9.4) можна перейти до опису в просторі станів виду

$$\begin{aligned} \boldsymbol{z}(k+1) &= f(\boldsymbol{z}(k),\boldsymbol{x}(k)),\\ \boldsymbol{y}(k) &= g(\boldsymbol{z}(k),\boldsymbol{x}(k)), \end{aligned} \tag{9.5}$$

де $z(i)$ — стан мережі в момент часу i.

Розглянемо деякі різновиди дискретних ДРМ.

9.3.1. Повнозв'язні ДРМ

Структуру повнозв'язної ДРМ наведено на рис. 9.2. ДРМ даного типу вивчалися в роботах [19, 89–91].

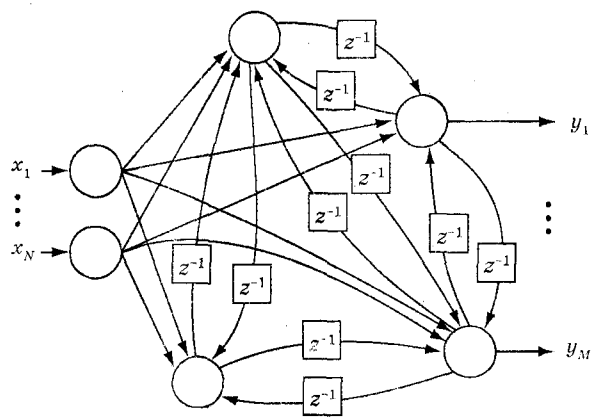


Рис. 9.2. Повнозв'язна ДРМ

Подана архітектура була спочатку запропонована для розв'язання задач, пов'язаних з аналізом й обробкою послідовностей, але згодом була використана також для ідентифікації нелінійних динамічних об'єктів. Однак цій мережі властивий серйозний недолік — повільна збіжність (істотна тривалість процесу навчання) і проблеми стійкості, які при цьому виникають [38].

9.3.2. Частково-рекурсивні мережі

Рекурсивні мережі дуже зручні для розв'язання задач розпізнавання, класифікації образів і прогнозування часових рядів. Іноді замість одиничних образів на вхід мережі подається одночасно послідовність $\boldsymbol{w}$ часткових образів у вигляді деякого вікна, що зміщується при надходженні кожного нового образу назад. Хоча таке ковзне вікно може бути реалізоване за допомогою ШНМ прямого поширення, більш ефективним є розв'язання таких задач за допомогою частково-рекурсивних мереж. Дані мережі займають проміжну позицію між «чистими» мережами прямого поширення й «чистими» рекурсивними мережами.

На відміну від повнозв'язних ДРМ, частково-рекурсивні мережі представляють багатошаровий персептрон, доповнений так

званим *контекстним* шаром, нейрони якого реалізують пам'ять мережі.

Запропонована *М. Джорданом* мережа є ШНМ прямого поширення, доповненою шаром контекстних нейронів (рис. 9.3), кількість яких збігається з кількістю виходів ШНМ [92].

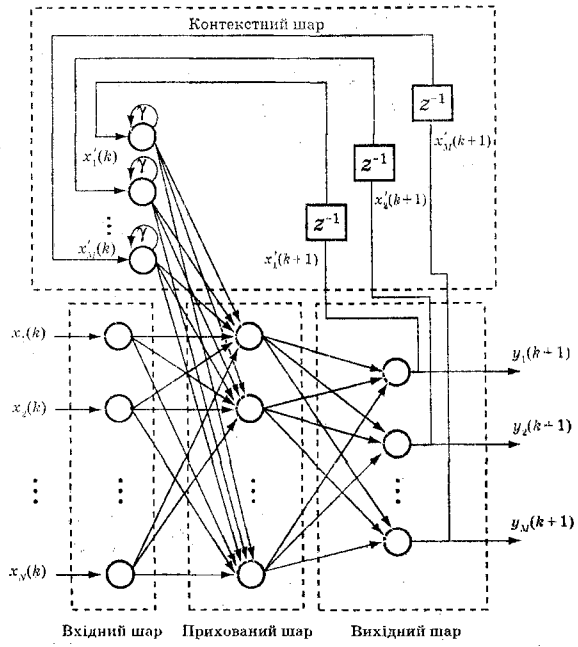


Рис. 9.3. Мережа Джордана

Вхідні сигнали мережі разом із сигналами контекстного шару надходять на входи нейронів прихованого шару, вихідні сигнали яких передаються на вихідний шар. Виходи нейронів цього шару є виходами ШНМ. Крім того, вихідні сигнали у вигляді сигналів зворотного зв'язку передаються з вагами λ на контекстний шар. Нейрони контекстного шару мають власні зворотні зв'язки з вагою γ, що звичайно не змінюється.

Вихідний сигнал мережі в кожен момент часу залежить від значення вхідного сигналу й стану, накопиченого в контекстному шарі. Тому мережа описується рівнянням і (9.5)

$$z(k+1) = f(z(k), x(k));$$
$$y(k) = g(z(k), x(k)),$$

де $z(k)$ — стан у момент часу k; g, f — функції, що визначають відповідно вихід і стан мережі на наступному такті (у теорії цифрових автоматів їх називають функціями виходів і переходів відповідно).

Те, що стан $z(k+1)$ залежить від $z(k)$ і $x(k)$, видно з топології мережі

$$z(k+1) = \hat{f}(z(k), y(k)) = \hat{f}(z(k), g(z(k), x(k)) = f(z(k), x(k)).$$

Послідовна зміна станів нейронів при деякому початковому стані z_0 описується формулою

$$z(k) = \begin{cases} z_0, & \text{якщо } k = 1; \\ \gamma z(k-1) + \lambda y(k-1), & \text{якщо } k > 1, \end{cases} \tag{9.6}$$

тобто

$$z(k) = \gamma^{k-1} z_0 + \lambda \sum_{j=1}^{k-1} \gamma^{j-1} y(k-j)\ , \tag{9.7}$$

де $\lambda \in (0,1]$ — ваговий параметр зворотного зв'язку (звичайно $\lambda = 1$).

Якщо $z_0 = 0$ и $\lambda = 1$, то

$$z(k) = \sum_{j=1}^{k-1} \gamma^{j-1} y(t-j), \tag{9.8}$$

тобто стан є експоненційно зваженою сумою всіх вихідних сигналів, що були до цього часу.

При малих значеннях γ вплив попередніх станів малий, а при $\gamma \to 1$ великий. При $\gamma = 1$ всі стани мають однакову вагу (використання параметра $\gamma < 1$, з одного боку, зменшує вплив образів, що раніше надійшли на поточний, а з іншого — дозволяє врахувати цей вплив). Зазвичай приймають $\gamma \approx 0{,}5$.

У процесі навчання мережі ваги γ_i не змінюються. Хоча принципово зміна γ можлива, проте, як свідчать дослідження, це лише призводить до затягування процесу навчання без істотного поліпшення роботи ШНМ. Однак питання вибору γ, малі значення якого дозволяють краще реагувати на образ, що знову надійшов, а більші — забезпечувати накопичення всіх образів, що надійшли раніше, залишається відкритим.

Мережа Джордана здатна асоціювати різні вхідні образи з різними вихідними послідовностями.

Мережа *Дж. Елмана* [93] є модифікацією мережі Джордана, у якій сигнали зворотних зв'язків надходять не з вихідного шару, а з виходів нейронів прихованого шару, тому кількість нейронів контекстного й прихованого шарів збігається. Крім того, тут немає власних зворотних зв'язків нейронів контекстного шару. Дані нейрони виконують роль функції активації. Надходження на вхід мережі першого образу активує нейрони всіх шарів: прихованого, вихідного й контекстного. Оскільки нейрони контекстного шару відіграють роль, аналогічну активаційній функції, вони перейдуть у новий стан, що відповідає стану нейронів прихованого шару, тобто копіюють (запам'ятовують) інформацію. З виходів нейронів прихованого шару сигнал передається на входи нейронів вихідного шару, які й формують вихідний сигнал мережі. При надходженні наступного образу стан нейронів контекстного шару відповідає попередньому образу.

Основним завданням нейронів прихованого шару є генерація бажаного вихідного сигналу на підставі порівняння образу, що знову надійшов, і образу, що запам'ятався в контекстному шарі.

Найпростіша мережа Елмана складається з одного прихованого й одного контекстного шару нейронів (рис. 9.4).

Для розв'язання складніших задач використовують ієрархічну частково-рекурсивну мережу, у якій кожен прихований шар має свій контекстний (рис. 9.5).

Можливості мережі різко зростають внаслідок можливості вибору різних значень γ_i.

Якщо відкинути сигнали зворотних зв'язків, що надходять на контекстний шар, то вийде чиста мережа прямого поширення, у якої контекстний шар є додатковим вхідним. Тому розширеий вхідний вектор складається із безпосередньо вхідного вектора x і вектора станів нейронів контекстного шару z, що визначається на кожному такті функцією переходів.

9.3.3. Локально-рекурсивні мережі прямого поширення

Мережі даного типу не використовують ні зворотний зв'язок між нейронами сусідніх шарів, ні латеральні зв'язки між нейронами одного шару. Рекурсивність у них завжди обмежується одним нейроном. Отримувані при цьому структури є лінійними, а зворотні зв'язки, що вводяться, інтерпретуються як фільтри з кінцевою (КІХ) або нескінченною (НІХ) імпульсною характеристикою. Як показано на рис. 9.6, існує три різних способи отримування локальної рекурсивності або, інакше кажучи, локального введення в мережу динаміки:

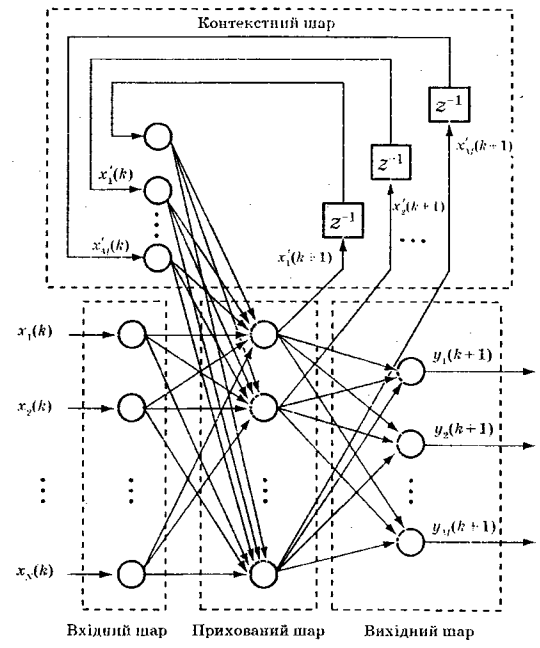


Рис. 9.4. Мережа Елмана

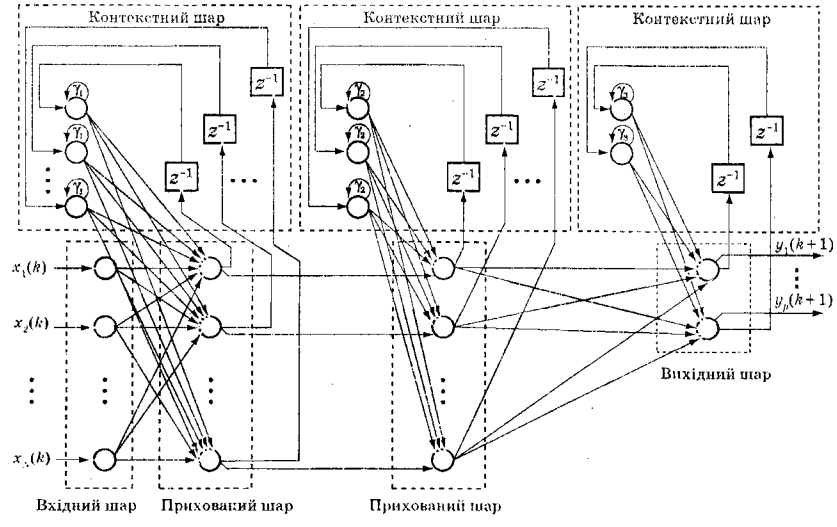
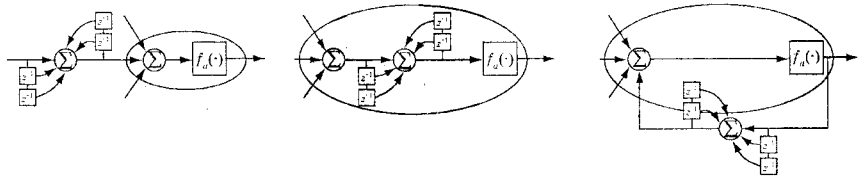


Рис. 9.5. Ієрархічна частково-рекурсивна мережа

- *динаміка синапсів*, що використовує локальні зворотні зв'язки нейронів (рис. 9.6, *а*);
- *«активаційна» динаміка*, що є окремим випадком динаміки синапсів і дозволяє у випадку однакових передавальних функцій усіх синапсів одного нейрона істотно спростити структуру мережі; у цьому випадку фільтри, використовувані в синапсах, можуть бути замінені одним фільтром, який має ті ж нулі й полюси та стоїть після операції підсумовування [94] (рис. 9.6, *б*);
- *динаміка зворотного зв'язку*, реалізована шляхом введення лінійного зворотного зв'язку з виходу нейрона на його вхід (рис. 9.6, *в*).



а *б* *в*

Рис. 9.6. Типи локальної рекурсивності

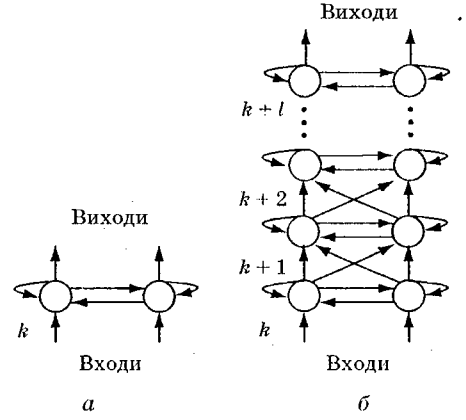
Хоча основою локально-рекурсивних мереж є звичайно багатошаровий персептрон, ці мережі можуть бути також реалізовані й на основі радіально-базисних мереж (*див.* розділ 15).

9.4. Навчання ДРМ

Оскільки ДРМ описуються нелінійними рівняннями, для настроювання параметрів цих мереж використовують методи нелінійної оптимізації, серед яких найчастіше застосовують градієнтні методи. Наявність у ДРМ зворотних зв'язків призводить до того, що значення градієнта залежить від попередніх станів мережі. У зв'язку з цим розрізнюють два підходи до розв'язання задачі навчання мережі: застосування *алгоритму зворотного поширення помилки* й застосування *адаптивних алгоритмів*, в основі яких лежать рекурентні процедури. Обидва ці підходи використовують градієнтні схеми мінімізації, тобто є звичайними градієнтними методами першого порядку (хоча можуть бути використані й методи більш високих порядків), що відрізняються кількістю використовуваної та збереженої в пам'яті інформації.

9.4.1. Алгоритм зворотного поширення помилки

Цей алгоритм запропоновано *Д. Румельхартом*, *Г. Гінтоном* і *Р. Уільямсом* у роботі [19] і використовує подання ДРМ у вигляді багатошарової мережі прямого поширення (БМПП), у якої на кожному такті роботи відбувається збільшення шарів на одиницю. Кожен шар БМПП складається з тієї ж кількості нейронів, що мають ті ж зв'язки, що й вихідна ДРМ. На рис. 9.7, *б* наведено багатошарову мережу прямого поширення, що розвивається, еквівалентну повнозв'язній ДРМ, яка містить два нейрони (рис. 9.7, *а*).



а *б*

Рис. 9.7. Проста повнозв'язна ДРМ (*а*), еквівалентна БМПП (*б*), що розвивається в часі

Перший шар такої мережі відповідає стану ДРМ у момент часу $k = 1$.

Очевидно, для навчання мережі може бути застосований алгоритм зворотного поширення помилки, що використовує в будь-який момент часу k ($k = 1, 2, \ldots$) обчислення градієнта

$$\frac{\partial I(k)}{\partial \boldsymbol{w}} = g(\boldsymbol{w}, Z(k), X(k)), \qquad (9.9)$$

де $\boldsymbol{w}$ — шукані параметри мережі; $Z(k)$ — матриця станів мережі розмірності $n \times k$ (n — кількість станів мережі, k — кількість попередніх тактів); $X(k)$ — матриця вхідних сигналів $N \times k$ (N — кількість входів).

З (9.9) видно, що для реалізації алгоритму настроювання ваг w необхідне запам'ятовування великого обсягу інформації, що зростає зі збільшенням кількості тактів навчання k. У зв'язку з цим частіше застосовують модифікації алгоритму зворотного поширення, що полягають у розповсюдженні вихідної помилки не на всі шари, починаючи з вихідного, а на обмежену їх кількість.

9.4.2. Адаптивний алгоритм навчання

Даний алгоритм запропоновано і досліджено *Р. Уільямсом* і *Д. Ціпсером* [95] та використовує обчислення градієнта в момент часу k на основі часток похідних станів нейронів, обчислених у момент часу $(k-1)$, тобто

$$\frac{\partial I(k)}{\partial w} = g(w, \frac{\partial z(k-1)}{\partial w}, z(k-1), x(k)). \qquad (9.10)$$

У цьому випадку обсяг необхідної пам'яті істотно зменшується й залежить тільки від кількості нейронів у мережі й кількості настроюваних параметрів. Хоча й тут необхідно брати до уваги всі попередні стани мережі, у цьому алгоритмі відбувається облік інформації лише про один останній такт, тобто складові в (9.10) $z(k-1)$ і $x(k)$ мають розмірності $n \times 1$ й $N \times 1$ відповідно.

Під час використання даного алгоритму в реальному часі припускають, що

$$\frac{\partial z(k-1)}{\partial w} \approx \frac{\partial z(k-1)}{\partial w(k-1)}, \qquad (9.11)$$

що й враховується в потактовій корекції ваг.

Контрольні запитання

1. У чому відмінність ДРМ від мережі Гопфілда?
2. Які існують різновиди ДРМ? У чому відмінність мережі Елмана від мережі Джордана?
3. Яким чином здійснюється навчання ДРМ?

10. МЕРЕЖА ВЕКТОРНОГО КВАНТУВАННЯ

Багато завдань розпізнавання мовних сигналів, зображень тощо пов'язані з необхідністю зберігання, обробки й передачі великих масивів даних, що вимагає значних обчислювальних ресурсів і витрат часу. Якщо масиви даних різняться незначною мірою, то істотної економії як обчислювальних засобів, так і необхідного для розв'язання задачі часу можна досягти, використовуючи стиснення даних або спеціальні методи їхнього кодування. При цьому дані можуть бути, наприклад, згруповані в деякі класи, яким привласнюється свій код або індекс. Образи можна також розбити на деякі кластери, визначивши для кожного з них свого типового представника (опорного представника, центра кластера). Досить добре розроблені методи стиснення даних вимагають для своєї реалізації наявності статистичної інформації (щільності розподілу) про досліджувані процеси; застосування ефективних методів кодування можливе, якщо відома частота появи образів. В умовах, коли така інформація відсутня, найбільш ефективними є ШНМ *векторного квантування* (*vector quantization, VQ*), розроблені й досліджені *Т. Когоненом* [8–14].

10.1. Структура мережі векторного квантування

Під векторним квантуванням (ВК) розуміють процес перетворення деякого вектора x з множини $A \in R^N$ у вектор w з множини $B \in R^M$, де $M < N$. Інакше кажучи, множина векторів x розмірності $N \times 1$ розбивається на кінцеве число класів M, $M < N$, кожному з яких привласнюється свій код (призначається опорний представник) w_i $(i = \overline{1,M})$. Множина всіх w_i утворить *кодову множину* (*кодову книгу*) класифікатора.

Векторне квантування здійснюється за допомогою методу «*найближчого сусіда*», причому під «найближчим» розуміється вектор, що задовольняє різним вимогам. Якщо як такий вибирається

вектор, що знаходиться від даного на мінімальній евклідовій відстані, тобто на відстані, що мінімізує вартісний функціонал

$$I(\boldsymbol{x},\boldsymbol{w}_j) = \|\boldsymbol{x}-\boldsymbol{w}_j\|^2 = \sum_{i=1}^{N}(x_i - w_{ij})^2, \qquad (10.1)$$

маємо класифікатор, що називається у літературі *Voronoi-класифікатор*. Приклад такої класифікації, розбивання всієї множини вхідних векторів на 7 класів ($M = 7$), наведено на рис. 10.1. Тут зірочками позначені вхідні вектори, а кільцями — відповідні опорні представники $w_i (i = \overline{1,7})$.

При поданні вхідного образу, що підлягає розпізнаванню, x_j видається представник w_i того класу, до якого відноситься x_j. Межі між класами утворять медіатриси між опорними представниками.

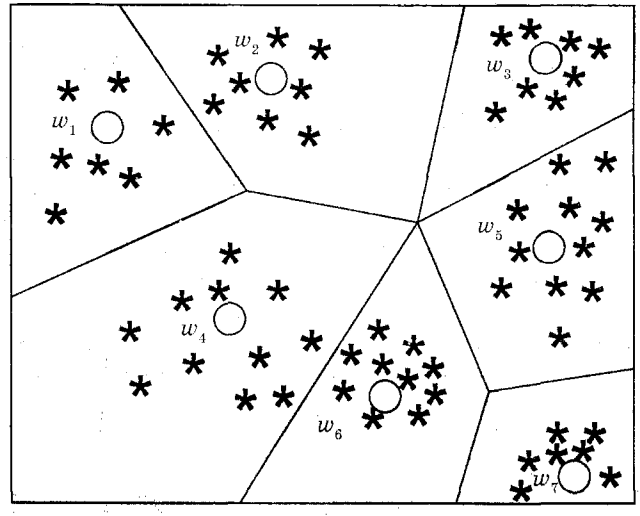


Рис. 10.1. Розбивання множини вхідних образів на класи

У мережі ВК реалізовано принцип *«переможець отримує все»* (*winner takes all*), коли за право подання будь-якого вхідного образу (вектора) змагаються всі нейрони вихідного шару. Нейрон, ваговий вектор якого w_c найближчий до вхідного образу, є переможцем.

Конкуруючі нейрони Когонена мають як власні зворотні зв'язки, так і латеральні, причому перші служать для посилення власного сигналу нейрона, а другі — для придушення впливу сигналів, що надходять від інших нейронів.

Кожен нейрон вхідного шару з'єднаний з усіма нейронами шару векторного квантування (VQ-шару). Сила зв'язку визначається відповідною вагою. На рис. 10.2 $w_j = (w_{1j}, w_{2j}, \ldots, w_{Nj})^T$ — вектор ваг j-го нейрона Когонена.

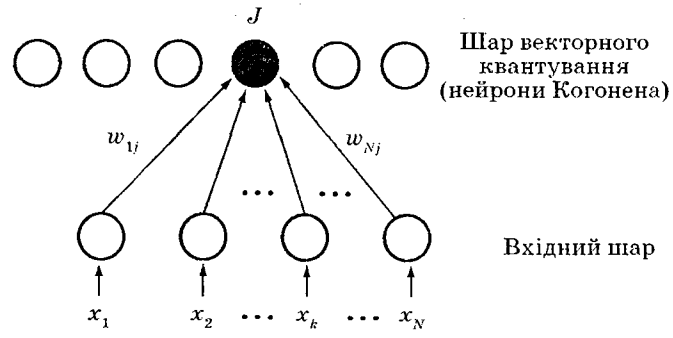


Рис. 10.2. Мережа векторного квантування

Динаміка мережі описується такими диференціальними рівняннями:

$$\frac{dy_j}{dt} = \sum_i w_{ij}x_i + \sum_{K\in S_j} v_{k_j} y_k - h(y_j), (j = \overline{1,M}), \qquad (10.2)$$

де x_i — вхідні сигнали ($i = \overline{1,N}$); w_{ij} — вага зв'язку між i-м та j-м нейронами; v_{k_j} — вага зв'язку між i-м й j-м нейронами; s_j — множина нейронів, пов'язаних з j-м нейроном; $h(y_j)$ — нелінійний член, що враховує ефекти насичення й зміщення.

У рівнянні (10.2) вагові коефіцієнти w_{ij} є тими, що настроюються, ваги ж зворотних зв'язків v_{k_j} задаються й у процесі роботи мережі не змінюються.

10.2. Неконтрольоване навчання мережі ВК

Існує кілька підходів до навчання мережі ВК. У найпростішому, початковому варіанті під час навчання відбувалася зміна ваг тільки нейрона-переможця відповідно до правила

$$\frac{dw_{ij}}{dt} = \begin{cases} \alpha(x_i - w_{ij}), & \text{якщо } y_j = 1; \\ 0 & \text{в іншому випадку.} \end{cases} \qquad (10.3)$$

Тут α — коефіцієнт навчання.

Якщо сума ваг для кожного нейрона є величиною постійною й $\|w_i\|^2 \approx 1$, а вхідні вектори нормалізовані $\|x\|^2 = 1$, то, як показав Когонен, час визначення нейрона-переможця може бути скорочено. При цьому нейрон-переможець c визначається шляхом мінімізації евклідової норми

$$\|x - w_c\|^2 = \min_j(\|x - w_j\|^2) \tag{10.4}$$

або

$$c = \arg\min_j(\|x - w_j\|^2). \tag{10.5}$$

При виконанні зазначених вище умов щодо норм вхідних і вагових векторів

$$\|x - w_j\|^2 = 2 - x^T w_j, \tag{10.6}$$

величина

$$y_j = x^T w_j = \sum_{i=1}^{N} x_i w_{ij} \tag{10.7}$$

може використовувався як критерій подібності образів x й w_j.

У дискретному випадку при поданні на кожному такті навчання нового вхідного образу відбувається корекція ваг нейрона-переможця за формулою

$$w_c(k+1) = w_c(k) + \alpha(x(k) - w_c(k)), \tag{10.8}$$

$$w_i(k+1) = w_i(k) \text{ при } i \neq c.$$

Як і в (10.3), коефіцієнт навчання α може бути обраний постійним або зменшуваним у процесі навчання. Як випливає з (10.8), вектор ваг нейрона-переможця w_c зміщується в напрямку вектора вхідного образу x. Процес корекції ваг цього нейрона зображено на рис. 10.3.

При $\alpha(k) \neq 1$ вектор $w(k+1)$ корегується на величину $\alpha(k)x(k) - w(k)$ і має нове розміщення $w(k+1)$. Видно, що навчання полягає в обертанні вагового вектора в напрямку вектора входів $x(k)$ без істотної зміни його довжини.

Процес, відображений виразами (10.4), (10.5) і (10.8), описує послідовну корекцію векторів вагових коефіцієнтів, які в асимптотиці забезпечують практично оптимальне розбивання простору вхідних образів на кластери. Оптимальність тут розуміється в тому значенні, що внаслідок віднесення поданого образу до найближчого опорного представника при даному розбиванні кількість неправильно класифікованих образів буде мінімальною.

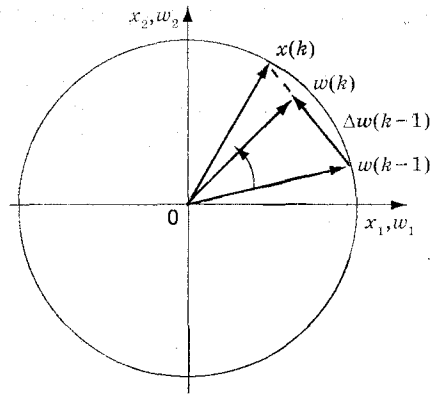


Рис. 10.3. Корекція ваг нейрона-переможця

Основним недоліком цього виду навчання є те, що якщо початкові розподіли векторів ваг і вхідних образів не є приблизно однаковими, то може виникнути ситуація, коли деякі з нейронів ніколи не стануть переможцями, тобто їхні вектори ваг не змінюватимуться. Для виключення подібної ситуації в алгоритм навчання вводять деякий механізм «пам'яті», що штрафує часто кореговані ваги нейронів-переможців. Як алгоритм навчання з «пам'яттю» може бути використаний, наприклад, такий:

$$w_i(k+1) = w_i(k) + \alpha(x(k) - w_i(k))z_i, \tag{10.9}$$

де

$$z_i = \begin{cases} 1, \text{ якщо } \|x - w_i\|^2 - B_i < \|x - w_j\|^2 \text{ для всіх } j \neq i, \\ 0 \text{ в іншому випадку;} \end{cases}$$

$B_i = c\,(\frac{1}{n} - p_i)$ — коефіцієнт штрафу;

p_i — часовий інтервал, на якому i-й нейрон є переможцем

$$p_i(k+1) = p_i(k) + b(y_i(k) - p_i(k));$$

$b \in (0,1]$ — постійний параметр;

c — постійний параметр, що впливає на величину штрафу.

Цей алгоритм навчання є більш ефективним, ніж (10.8). Однак сьогодні значного поширення отримали алгоритми корекції параметрів, в основі яких лежить *контрольоване навчання*.

10.3. Контрольоване навчання мережі ВК

Основною відмінністю *контрольованого навчання* мережі ВК (*Learning Vector Quantization, LVQ*) від розглянутого вище є використання для кожного вхідного образу $\boldsymbol{x}$ бажаного відповідного вихідного сигналу. Цей вид навчання реалізується різними способами.

10.3.1. LVQ1

Як і в описаному вище методі, переможцем у мережі є той нейрон, вектор ваг якого $\boldsymbol{w}_c$ найближчий до вхідного образу $\boldsymbol{x}$, тобто нейрон c визначається як

$$c = \arg\min_j(\| \boldsymbol{x} - \boldsymbol{w}_j \|^2).$$

Значення всіх ваг w_j, що мінімізують помилку класифікації, обчислюються LVQ1-методом асимптотично. При цьому корекція ваг відбувається за правилом

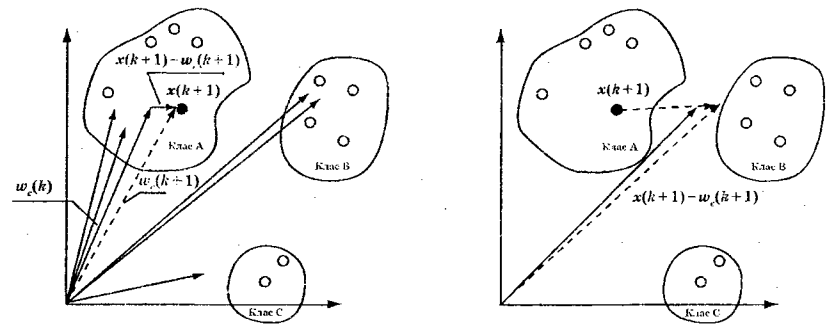
$$\boldsymbol{w}_c(k+1) = \begin{cases} \boldsymbol{w}_c(k) + \alpha(k)[\boldsymbol{x}(k) - \boldsymbol{w}_c(k)], & \text{якщо } \boldsymbol{w}_c \text{ й } \boldsymbol{x} \text{ належать одному класу;} \\ \boldsymbol{w}_c(t) - \alpha(k)[\boldsymbol{x}(t) - \boldsymbol{w}_c(k)] & \text{в іншому випадку;} \end{cases} \quad (10.10)$$

$$\boldsymbol{w}_j(k+1) = \boldsymbol{w}_j(k) \text{ для всіх } j \neq c,$$

де $\alpha(k) \in (0, 1]$. Параметр $\alpha(k)$ може залишатися постійним або монотонно зменшуватися.

Таким чином, вектор ваг нейрона-переможця $\boldsymbol{w}_c$, що найближче розташований до поданого вхідного вектора, зміщується в напрямку останнього, якщо вхідний вектор відноситься до одного з ним класу, і віддаляється від нього в іншому випадку. Ваги інших нейронів не змінюються. Зміну ваг нейрона Когонена зображено на рис. 10.4. На рис. 10.4, *а* наведено випадок, коли вектори $\boldsymbol{x}(k)$ і $\boldsymbol{w}_c(k)$ відносяться до одного класу, на рис. 10.4, *б* — до різного.

Приклад 10.1. Розглянемо роботу методу LVQ1 на прикладі навчання мережі, що складається з двох нейронів і класифікує вісім векторів, які відносяться до двох різних класів:



а *б*

Рис. 10.4. Зміна ваг нейронів Когонена

Вектор	Клас
$\boldsymbol{x}_1 = [0\ 0\ 1\ 1]^T$	1
$\boldsymbol{x}_2 = [1\ 1\ 0\ 1]^T$	2
$\boldsymbol{x}_3 = [0\ 1\ 1\ 1]^T$	1
$\boldsymbol{x}_4 = [1\ 0\ 0\ 1]^T$	1
$\boldsymbol{x}_5 = [1\ 1\ 0\ 0]^T$	2
$\boldsymbol{x}_6 = [0\ 1\ 1\ 0]^T$	2
$\boldsymbol{x}_7 = [1\ 0\ 1\ 1]^T$	1
$\boldsymbol{x}_8 = [1\ 1\ 1\ 1]^T$	2

Як початкові значення векторів ваг $\boldsymbol{w}_1$ й $\boldsymbol{w}_2$, асоційованих із класами 1 й 2, приймемо відповідно вектори $\boldsymbol{x}_1$ і $\boldsymbol{x}_2$. Навчання мережі здійснюватимемо за алгоритмом (10.10) з $\alpha = 0{,}1$.

Розглянемо перший цикл навчання.
При надходженні вектора $\boldsymbol{x}_3$ (з урахуванням того, що $\boldsymbol{w}_1(0) = \boldsymbol{x}_1$ й $\boldsymbol{w}_2(0) = \boldsymbol{x}_2$) маємо

$$\|\boldsymbol{x}_3 - \boldsymbol{w}_1\|^2 = 1 = \min,$$
$$\|\boldsymbol{x}_3 - \boldsymbol{w}_2\|^2 = 2.$$

Оскільки переможцем виявився перший нейрон, то настроюємо його ваги, беручи до уваги, що $\boldsymbol{x}_3$ відноситься до класу 1, тобто

$$\boldsymbol{w}_1(1) = \boldsymbol{w}_1(0) + 0{,}1[\boldsymbol{x}_3 - \boldsymbol{w}_1(0)] = (0\ 0\ 1\ 1)^T + 0{,}1[(0\ 1\ 1\ 1)^T - (0\ 0\ 1\ 1)^T] =$$
$$= (0\ 0{,}1\ 1\ 1)^T.$$

При надходженні вектора x_4 маємо

$$\|x_4 - w_1(1)\|^2 = 2{,}01,$$
$$\|x_4 - w_2(0)\|^2 = 1 = \min.$$

Тому корегуємо ваги $w_2(0)$ з урахуванням того, що $w_2(0)$ і x_4 відносяться до різних класів

$$w_2(1) = w_2(0) - 0{,}1[x_4 - w_2(0)] = (1\ 1\ 0\ 1)^T - 0{,}1[(1\ 0\ 0\ 1)^T - (1\ 1\ 0\ 1)^T] =$$
$$= (1\ \ 1{,}1\ \ 0\ \ 1)^T.$$

Аналогічно отримуємо
для x_5:

$$\|x_5 - w_1(1)\|^2 = 3{,}01,$$
$$\|x_5 - w_2(1)\|^2 = 1{,}01 = \min,$$

$$w_2(2) = w_2(1) + 0{,}1[x_5 - w_2(1)] = (1\ \ 1{,}1\ \ 0\ \ 1)^T +$$
$$+ 0{,}1[(1\ \ 1\ \ 0\ \ 0)^T - (1\ \ 1{,}1\ \ 0\ \ 1)^T] = (1\ \ 1{,}09\ \ 0\ \ 0{,}9)^T;$$

для x_6:

$$\|x_6 - w_1(1)\|^2 = 1{,}01 = \min,$$
$$\|x_6 - w_2(2)\|^2 = 2{,}8181,$$

$$w_1(2) = w_1(1) - 0{,}1[x_6 - w_1(1)] = (0\ 0{,}1\ 1\ 1)^T - 0{,}1[(0\ 1\ 1\ 0)^T - (0\ 0{,}1\ 1\ 1)^T] =$$
$$= (0\ \ 0{,}01\ \ 1\ \ 1{,}1)^T;$$

для x_7:

$$\|x_7 - w_1(2)\|^2 = 1{,}0101 = \min,$$
$$\|x_7 - w_2(2)\|^2 = 2{,}1981,$$

$$w_1(3) = w_1(2) + 0{,}1[x_7 - w_1(2)] = (0\ 0{,}01\ 1\ 1{,}1)^T +$$
$$+ 0{,}1[(1\ 0\ 1\ 1)^T - (0\ 0{,}01\ 1\ 1{,}1)^T] = (0{,}1\ \ 0{,}009\ \ 1\ \ 1{,}09)^T;$$

для x_8:

$$\|x_8 - w_1(3)\|^2 = 2{,}99,$$
$$\|x_8 - w_2(2)\|^2 = 1{,}01 = \min,$$

$$w_2(3) = w_2(2) + 0{,}1[x_8 - w_2(2)] = (1\ \ 1{,}09\ \ 0\ \ 0{,}9)^T +$$
$$+ 0{,}1[(1\ \ 1\ \ 1\ \ 1)^T - (1\ \ 1{,}09\ \ 0\ \ 0{,}9)^T] = (1\ \ 1{,}081\ \ 0{,}1\ \ 0{,}91)^T.$$

Перший цикл навчання завершено. Отримані значення векторів ваг

$$w_1 = (0{,}1\ \ 0{,}009\ \ 1\ \ 1{,}09)^T;$$
$$w_2 = (1\ \ 1{,}081\ \ 0{,}1\ \ 0{,}91)^T$$

призводять до того, що мережа відносить x_4 до другого класу, а x_6 — до першого. Продовжуючи навчання мережі, робимо висновок, що правильна класифікація буде досягнута після 12 циклу. При цьому

$$w_1 = (0{,}2251\ \ 0{,}0816\ \ 1\ \ 1{,}1419)^T;$$
$$w_2 = (0{,}9269\ \ 1{,}2046\ \ 0{,}2980\ \ 0{,}7130)^T.$$

Після 500 циклів навчання з $\alpha = 0{,}1$ отримаємо такі значення шуканих векторів ваг:

$$w_1 = (0{,}4364\ \ 0{,}2116\ \ 0{,}8087\ \ 1{,}0326)^T;$$
$$w_2 = (0{,}7901\ \ 1{,}0488\ \ 0{,}455\ \ 0{,}5481)^T.$$

Неважко перевірити, що при даних вагах всі подані мережі вектори будуть класифіковані правильно.

10.3.2. LVQ2.1

У LVQ2.1 класифікація відбувається аналогічно **LVQ1**, тобто шляхом мінімізації обраної метрики. Відмінність полягає в способі корекції ваг.

У методі LVQ2.1 визначається вектор w_i і наступний w_j та здійснюється адаптивна корекція ваг за таких умов:

— вектори w_i й w_j відносяться до різних класів;

— вектор x належить або тому класу, до якого відноситься w_i, або тому, якому належить w_j;

— вектор x перебуває в деякому «вікні» щодо перпендикуляра між цими двома класами.

Якщо позначити евклідову відстань між векторами x й w_i як d_i, а між x й w_j — як d_j, то вектор x перебуває у «вікні» відносної ширини V, якщо

$$\min\left(\frac{d_i}{d_j}, \frac{d_j}{d_i}\right) > s,$$

де $s = \dfrac{1-V}{1+V}$.

Когонен запропонував вибирати $V = 0{,}2 \div 0{,}3$. Слід зазначити, що це «вікно» — не прямокутне.

Графічно принцип роботи LVQ 2.1 наведено на рис. 10.5.
Алгоритм корекції ваг має такий вигляд:

$$\begin{aligned} \boldsymbol{w}_i(k+1) &= \boldsymbol{w}_i(k) + \alpha(k)[\boldsymbol{x}(k) - \boldsymbol{w}_i(k)]; \\ \boldsymbol{w}_j(k+1) &= \boldsymbol{w}_j(k) - \alpha(k)[\boldsymbol{x}(k) - \boldsymbol{w}_j(k)]. \end{aligned} \qquad (10.11)$$

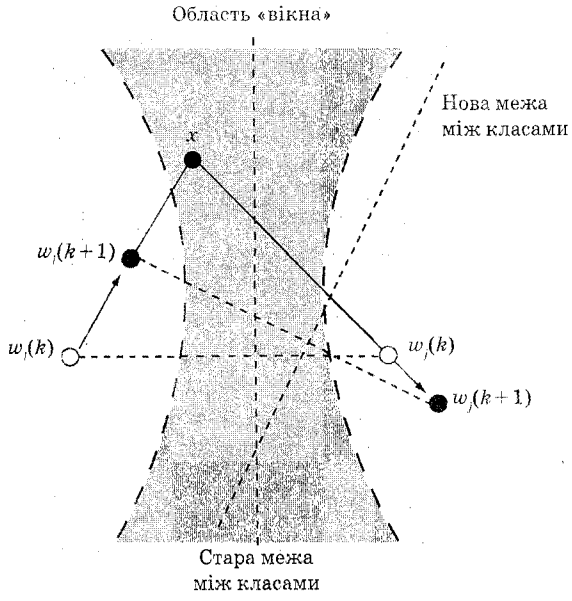


Рис. 10.5. Принцип роботи LVQ 2.1

Як видно з рисунка, у результаті роботи алгоритму (10.11) змінюється розташування межі між класами.

10.3.3. LVQ3

Цей метод є модифікацією LVQ2.1, що дозволяє змінити не тільки ваги векторів $\boldsymbol{w}_i$ й $\boldsymbol{w}_j$, але й межу між класами, якщо обидва вектори належать тому ж класу, що й $\boldsymbol{x}$.

Якщо $\boldsymbol{w}_i$ й $\boldsymbol{w}_j$ відносяться до різних класів, а $\boldsymbol{w}_i$ — до того ж класу, що й $\boldsymbol{x}$, то корекція вагових векторів відбувається, як у LVQ2.1, тобто за правилом (10.11).

Якщо ж $\boldsymbol{w}_i$ й $\boldsymbol{w}_j$ відносяться до одного класу, то адаптація ваг здійснюється за алгоритмом

$$\begin{aligned} \boldsymbol{w}_i(k+1) &= \boldsymbol{w}_i(k) + e\alpha(k)[\boldsymbol{x}(k) - \boldsymbol{w}_i(k)]; \\ \boldsymbol{w}_j(k+1) &= \boldsymbol{w}_j(k) - e\alpha(k)[\boldsymbol{x}(k) - \boldsymbol{w}_j(k)], \end{aligned} \qquad (10.12)$$

де $e \in [0{,}1 \div 0{,}5]$ деякий заданий параметр (оптимальне значення e залежить від ширини «вікна», і для вузького «вікна» краще вибирати мале e).

Цей алгоритм є більш стійким, ніж LVQ2.1, оскільки при неперервному навчанні ваги векторів змінюються незначною мірою. Якщо LVQ1 й LVQ3 адаптують розміщення вагових векторів мережі до розподілу вхідних образів, то LVQ2.1 оптимізує відносне вилучення вагового вектора від межі між класами. Когонен рекомендує застосовувати для швидкого початкового навчання мережі в першу чергу LVQ2.1 з невеликим коефіцієнтом навчання α і відносно малим числом тактів навчання.

10.3.4. OLVQ1

Недолік LVQ1 — повільна швидкість збіжності — усунутий в оптимізованому LVQ1, OLVQ1 (*Optimized Learning Vector Quantization*), відповідно до якого адаптація вектора ваг мережі відбувається за правилом

$$\mathbf{w}_c(k+1) = \begin{cases} \boldsymbol{w}_c(k) + \alpha_c(k)[\boldsymbol{x}(k) - \boldsymbol{w}_c(k)], & \text{якщо } \boldsymbol{w}_c \text{ й } \boldsymbol{x} \text{ відносяться} \\ & \text{до одного класу;} \\ \boldsymbol{w}_c(k) - \alpha_c(k)[\boldsymbol{x}(k) - \boldsymbol{w}_c(k)] & \text{в іншому випадку;} \end{cases} \qquad (10.13)$$

$$\boldsymbol{w}_j(k+1) = \boldsymbol{w}_j(k) \text{ для всіх } j \neq c.$$

Оптимізація LVQ1 здійснюється шляхом вибору оптимальних значень коефіцієнта навчання $\alpha_c^{\text{опт}}(k)$. Визначення оптимального $\alpha_c^{\text{опт}}(k)$ відбувається у такий спосіб.

Запишемо (10.13) у вигляді

$$\boldsymbol{w}_c(k+1) = \boldsymbol{w}_c(k) + S(k)\alpha_c(k)\big[\boldsymbol{x}(k) - \boldsymbol{w}_c(k)\big], \qquad (10.14)$$

де

$$S(k) = \begin{cases} +1, & \text{якщо } w \text{ й } x \text{ відносяться до одного класу;} \\ -1, & \text{якщо } w \text{ й } x \text{ відносяться до різних класів.} \end{cases}$$

Слід зазначити, що $x(k)$ несе інформацію про образ, що знову надійшов, а інформація про всі попередні вхідні сигнали міститься в $w_c(k)$, оскільки

$$\begin{aligned} w_c(k+1) &= [1 - S(k)\alpha_c(k)]w_c(k) + S(k)\alpha_c(k)x(k) = \\ &= [1 - S(k)\alpha_c(k-1)][1 - S(k)\alpha_c(k)]w_c(k-1) + \\ &+ S(k)\alpha_c(k)[1 - S(k)\alpha_c(k-1)]x(k-1) + S(k)\alpha_c(k)x(k). \end{aligned} \tag{10.15}$$

Величини коефіцієнтів навчання $\alpha(k)$, $\alpha(k-1)$, ... відбивають вплив відповідних вхідних образів на корегувальний вектор w_c. З (10.15) видно, що вплив на $w_c(k+1)$ образу $x(k)$ визначається вагою $\alpha_c(k)$, образу $x(k-1)$ — вагою $\alpha_c(k-1)[1 - S(k)\alpha_c(k)]$ і т. д. Розподіл вагових векторів w є статистично оптимальним, якщо при $k \to \infty$ вплив усіх образів, що навчають, на зміну вектора w_c однаковий. Формально це можна записати так:

$$\alpha_c(k) = [1 - S(k)\alpha_c(k)]\alpha_c(k-1). \tag{10.16}$$

Оскільки на двох сусідніх тактах підсумовуючий вплив відповідних вхідних векторів на ваговий вектор однаковий й це справедливо для всіх моментів часу k, то за індукцією можна показати, що при $k \to \infty$ вплив усіх вхідних векторів на w однаковий. З рівняння (10.16) маємо:

$$\alpha_c(k) = \frac{\alpha_c(k-1)}{1 + \alpha_c(k-1)S(k)}. \tag{10.17}$$

Хоча $\alpha_c(k)$ принципово може зростати внаслідок того, що на деяких тактах $S(k) = -1$, значення цього коефіцієнта не має бути більше одиниці.

Зазвичай як початкове значення вибирають $\alpha_c(0) = 0{,}3$.

Оскільки цей метод є найбільш швидким, Когонен рекомендує починати навчання мережі саме з його допомогою, а потім переходити на будь-який із LVQ.

Приклад 10.2. Розглянемо роботу алгоритму OLVQ на тестовому завданні з прикладу 10.1. Як початкове значення параметра $\alpha_c(0)$ приймалося $\alpha_c(0) = 0{,}3$. Після першого циклу навчання із кроком 0,1 було отримано

$w_1 = (0{,}3000 \;\; 0{,}0969 \;\; 1{,}0000 \;\; 1{,}1615)^T$;

$w_2 = (1{,}0000 \;\; 1{,}1200 \;\; 0{,}3000 \;\; 0{,}7000)^T$;

$\alpha = 0{,}2308$.

Це призвело до того, що невірно розпізнався вектор x_4 (під час використання методу LVQ1 після першого циклу навчання неправильно розпізналися вектори x_4 й x_6).

Після другого циклу навчання всі вектори були правильно класифіковані. При цьому

$w_1 = (0{,}3639 \;\; 0{,}1051 \;\; 1{,}0000 \;\; 1{,}2233)^T$;

$w_2 = (1{,}0000 \;\; 1{,}1857 \;\; 0{,}3588 \;\; 0{,}6412)^T$;

$\alpha = 0{,}1579$.

Отже, застосування методу OLVQ привело до отримання розв'язку за два цикли навчання (метод LVQ1 забезпечив отримання розв'язку після 12 циклів).

Контрольні запитання

1. Розгляньте структуру мережі ВК. Який принцип вона реалізує?
2. Як здійснюється неконтрольоване навчання в мережі ВК?
3. Назвіть основні алгоритми контрольованого навчання мережі ВК. Поясніть відмінності між ними.

11. МЕРЕЖА КОГОНЕНА

У багатьох моделях ШНМ вирішальну роль відіграють зв'язки між нейронами, які визначаються ваговими коефіцієнтами й зазначають місце нейрона в мережі. Однак у біологічних системах, наприклад, у мозку, сусідні нейрони, отримуючи аналогічні вхідні сигнали, реагують на них подібним чином, тобто групуються, утворюючи деякі області. Оскільки під час обробки багатовимірного вхідного образу здійснюється його проектування на область меншої розмірності зі збереженням його топології, нерідко подібні мережі називають *мапами* (*self-organizing feature map*). У таких мережах істотним є врахування взаємного розташування нейронів одного шару.

Мережа Когонена (*самоорганізувальна мапа*) відноситься до мереж, що самоорганізуються, які під час надходження вхідних сигналів, на відміну від мереж, що використовують навчання із учителем, не отримують інформацію про бажаний вихідний сигнал. У зв'язку з цим неможливо сформувати критерій настроювання, заснований на неузгодженості реальних і необхідних вихідних сигналів ШНМ, тому вагові параметри мережі корегують, виходячи з інших міркувань. Усі подані вхідні сигнали із заданої навчальної множини самоорганізувальна мережа у процесі навчання розділяє на класи, будуючи так звані топологічні мапи.

Основною в цьому напрямку стала робота *Т. Когонена* [11].

11.1. Структура мережі Когонена

Мережа Когонена використовує таку модель (рис. 11.1): мережа складається з M нейронів, що утворюють прямокутні решітки на площині — шар.

До нейронів, розташованих в одному шарі, що є двовимірною площиною, підходять нервові волокна, по яких надходить N-вимірний вхідний сигнал. Кожен нейрон характеризується своїм

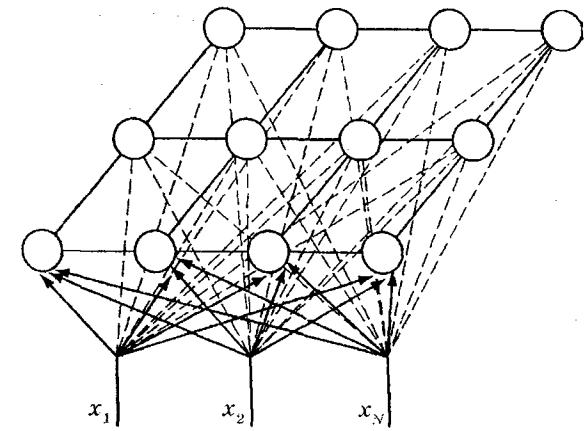


Рис. 11.1. Модель мережі Когонена

розміщенням у шарі й ваговим коефіцієнтом. Розміщення нейронів, у свою чергу, характеризується деякою метрикою й визначається топологією шару, при якій сусідні нейрони під час навчання впливають один на одного сильніше, ніж розташовані далі. Кожен нейрон утворює зважену суму вхідних сигналів (2.2) з $w_{ij} > 0$, якщо синапси прискорювальні, і $w_{ij} < 0$ — якщо гальмуючі. Наявність зв'язків між нейронами призводить до того, що при збудженні одного з них можна обчислити збудження інших нейронів у шарі, причому це збудження зі збільшенням відстані від збудженого нейрона зменшується. Тому центр реакції шару, що виникає у відповідь на отримане роздратування, відповідає місцезнаходженню збудженого нейрона. Зміна вхідного сигналу, що навчає, призводить до максимального збудження іншого нейрона й відповідно — до іншої реакції шару.

Мережа Когонена може розглядатися як подальший розвиток LVQ (*див.* розділ 10). Відмінність їх полягає у способах навчання.

11.2. Навчання мережі Когонена

Замість того, щоб шукати місцезнаходження нейрона шляхом розв'язання загальних рівнянь збудження, Когонен істотно спростив розв'язання задачі, виділяючи зі всіх нейронів шару лише один c-й нейрон, для якого зважена сума вхідних сигналів максимальна

$$c = \arg\max_j (\boldsymbol{x}^T \boldsymbol{w}_j). \qquad (11.1)$$

Зазначимо, що досить корисною операцією попередньої обробки вхідних векторів є їхня нормалізація

$$\overline{x_i} = \frac{x_i}{\|\boldsymbol{x}\|},\ i = \overline{1,N}, \tag{11.2}$$

що перетворює вектори вхідних сигналів в одиничні з тим же напрямком.

У виразі (11.2) $\|\boldsymbol{x}\| = \left(\sum_{i=1}^{N} x_i^2\right)^{1/2}$.

У цьому випадку внаслідок того, що сума ваг кожного нейрона одного шару $\sum_i w_{ij}$ для всіх нейронів цього шару однакова й $\|\boldsymbol{x}\| = 1$, умова (11.1) еквівалентна умові

$$c = \arg\min_j \|\boldsymbol{x} - \boldsymbol{w}_j\|. \tag{11.3}$$

Отже, буде активований тільки той нейрон, вектор ваг якого $\boldsymbol{w}$ найближчий до вхідного вектора $\boldsymbol{x}$. А оскільки перед початком навчання невідомо, який саме нейрон активуватиметься при поданні мережі конкретного вхідного вектора, мережа навчається без учителя, тобто *самонавчається*.

Вводячи потенційну функцію — функцію відстані f_{ij} («сусідства») між i-м й j-м нейронами з місцезнаходження r_i й r_j відповідно, що монотонно спадає зі збільшенням відстані між цими нейронами, Когонен запропонував такий алгоритм корекції ваг:

$$\boldsymbol{w}_{ij}(k+1) = \boldsymbol{w}_{ij}(k) + \alpha(k) f_{ij}(k)\left(\boldsymbol{x}(k) - \boldsymbol{w}_{ij}(k)\right), \tag{11.4}$$

де $\alpha(k) \in (0,1]$ — коефіцієнт посилення, що змінюється в часі (звичайно вибирають $\alpha = 1$ на першій ітерації, поступово зменшуючи в процесі навчання до нуля); $f_{ij}(k)$ — монотонно спадна функція

$$f_{ij}(k) = f(\|r_i - r_j\|, k) = f(d,k) = f(d,\sigma);$$

r_i, r_j — вектори, що визначають положення нейронів i й j у решітках. При прийнятій метриці $d = \|r_i - r_j\|$ функція $f_{ij}(k)$ із зростанням часу k прямує до нуля. На практиці замість параметра часу k використовують параметр відстані σ, що задає величину області «сусідства» і зменшується із часом до нуля.

Вибір функції $f_{ij}(k)$ також впливає на величини ваг усіх нейронів у шарі. Очевидно, що для нейрона-переможця c

$$f_c(\|r_i - r_j\|) = f_c(0) = 1. \tag{11.5}$$

Зважаючи на те, що визначення нейронів-переможців у мережі Когонена й у LVQ відбувається ідентично, то й корекція ваг цих нейронів здійснюється однаково (*див.* рис. 10.3).

На рис. 11.2 наведено приклад зміни двовимірних ваг мапи $\boldsymbol{w}_j = (w_{j1}, w_{j2})^T$, що утворює ланцюг. З появою вхідного образу $\boldsymbol{x}$ найбільш сильно змінюється ваговий вектор нейрона-переможця 5, менш сильно — ваги розташованих поруч із ним нейронів 3, 4, 6, 7. А оскільки нейрони 1, 2, 8, 9 перебувають поза областю «сусідства», їхні вагові коефіцієнти не змінюються.

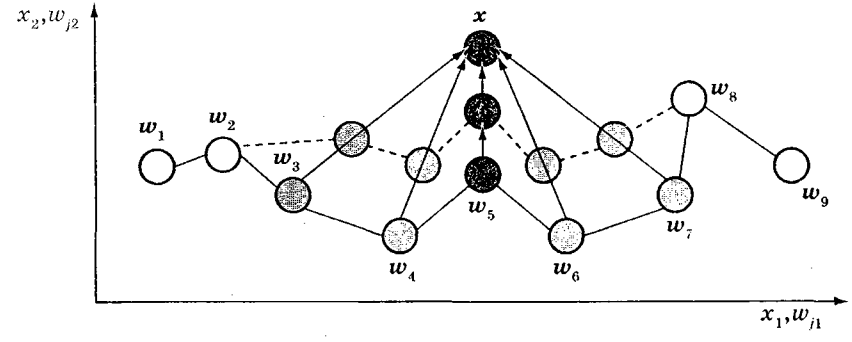


Рис. 11.2. Зміна ваг мапи Когонена

Отже, алгоритм навчання мережі Когонена може бути описаний так:

1. *Ініціалізація.*

Ваговим коефіцієнтам усіх нейронів привласнюються малі випадкові значення й здійснюється їхня нормалізація. Вибирається відповідна потенційна функція $f_{ij}(d)$ і призначається початкове значення коефіцієнта підсилення α_0.

2. *Вибір сигналу, що навчає.*

Із усієї множини векторів навчальних вхідних сигналів відповідно до функції розподілу $P(\boldsymbol{x})$ вибирається один вектор $\boldsymbol{x}$, що представляє «сенсорний сигнал», поданий мережі.

3. *Аналіз відгуку (вибір нейрона).*

За формулою (11.1) визначається активований нейрон.

4. *Процес навчання.*

Відповідно до алгоритму (11.4) змінюються вагові коефіцієнти активованого й сусідніх з ним нейронів доти, поки не буде отримано необхідного значення критерію якості навчання або не буде подане задане число вхідних векторів, що навчають. Остаточне значення вагових коефіцієнтів збігається з нормалізованими векторами входів.

Оскільки мережа Когонена здійснює проектування N-вимірного простору образів на M-вимірну мережу, аналіз збіжності алгоритму навчання є досить складною задачею. Однак властивості алгоритму можуть бути продемонстровані на простих прикладах.

Приклад 11.1. Розглянемо одновимірний випадок. Припустимо, що мережа складається з одного нейрона, а вхідний сигнал $x \in [a,b]$. Нехай початкове значення ваги дорівнює x_1 (рис. 11.3)

Рис. 11.3. Навчання мережі, що складається з одного нейрона

Кожна зміна x залежить тільки віддаленості від a й b, тобто

$$\frac{dx}{dt} = \frac{\alpha}{2}\left[(b-x)+(a-x)\right] = \alpha\left(\frac{a+b}{2} - x\right),$$

де $\alpha \in (0,1]$.

Тоді переміщення x_1 у x_2 відбуватиметься за правилом

$$x_2 = x_1 + \alpha\left(\frac{a+b}{2} - x_1\right) = \frac{a+b}{2} + (1-\alpha)\cdot\left(x_1 - \frac{a+b}{2}\right).$$

Аналогічно

$$x_3 = x_2 + \alpha\left(\frac{a+b}{2} - x_2\right) = \frac{a+b}{2} + (1-\alpha)^2\cdot\left(x_1 - \frac{a+b}{2}\right).$$

Після n-ї ітерації отримуємо

$$x_n = \frac{a+b}{2} + (1-\alpha)^{n-1}\cdot\left(x_1 - \frac{a+b}{2}\right),$$

звідки випливає, що

$$\lim_{n\to\infty} x_n = \frac{a+b}{2},$$

тобто навчання завершується, коли значення ваги ділитиме інтервал $[a, b]$ навпіл. Отже, x_n займе стійке розташування в середині інтервалу $[a, b]$.

Приклад 11.2. Розглянемо одновимірний шар, що складається з нейронів, які ділять заданий інтервал $[a, b]$ на N рівних частин. Нехай ваги нейронів упорядковані, тобто $a < x_1 < x_2 < \ldots < x < b$ (рис. 11.4)

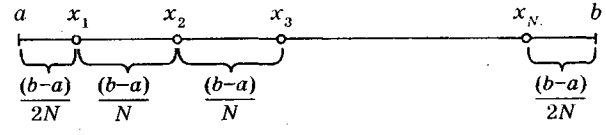


Рис. 11.4. Навчання мережі

Тоді кожна вага мережі може бути обчислена за формулою

$$x_i = a + (2i-1)\frac{b-a}{2N}, \quad i = \overline{1,N}.$$

Оскільки ці ваги ділять інтервал $[a, b]$ на N рівних частин, то вони (*див.* приклад 11.1) займають на цьому інтервалі стійкі розміщення.

Приклад 11.3. Розглянемо двовимірний шар, що складається з $N \times N$ нейронів. Вхідний вектор складатиметься із двох компонентів $\boldsymbol{x} = (x_1 x_2)^T$, $x_1 \in [a, b]$, $x_2 \in [c, d]$. Відповідно вектор вагових коефіцієнтів ij-нейрона має вигляд $\boldsymbol{w}^{ij} = (w_1^{ij}, w_2^{ij})^T$. Припустимо, що всі компоненти вагового вектора впорядковані $w_1^{ij} < w_1^{ik}$, $j < k$, $w_2^{ij} < w_2^{mj}$, $i < m$. Визначаючи для кожного стовпця нашої матриці ваг середнє значення

$$w_1^j = \frac{1}{N}\sum_{i=1}^{N} w_1^{ij}, \quad j = \overline{1,N},$$

з урахуванням упорядкованості компонентів отримуємо

$$a < w_1^1 < w_1^2 < \ldots < w_1^N < b.$$

Аналогічно отримуємо для кожного рядка матриці ваг

$$c < w_2^1 < w_2^2 < \ldots < w_2^N < d.$$

За аналогією із прикладом 11.2 слід зазначити, що в процесі навчання вагові коефіцієнти розіб'ють інтервали $[a, b]$, $[c, d]$ на рівні частини, тобто нейрони розташуються у вузлах решіток.

Якби з кожним нейроном шару асоціювався один вхідний вектор, то вектор ваг будь-якого нейрона шару Когонена міг би бути навчений за допомогою одного обчислення, оскільки вага нейрона-переможця корегувалися б з $\alpha = 1$ (відповідно до (11.4) для одновимірного випадку вага відразу б потрапляла в центр відрізка $[a, b]$). Однак звичайно навчальна множина містить безліч подібних між собою вхідних векторів, і мережа Когонена має бути навчена активувати той самий нейрон для кожного з них. Це досягається усередненням вхідних векторів шляхом зменшення величини α при поданні кожного наступного вхідного сигналу. Отже, ваги, асоційовані з нейроном, усередняться й приймуть значення поблизу «центра» вхідних сигналів, для яких даний нейрон є «переможцем».

Приклад 11.4. Нехай мережа, що складається із трьох нейронів, має класифікувати такі пропоновані їй нормалізовані вектори:

$x(1) = (0{,}8;\ 0{,}6)^T$; $x(2) = (0{,}7071;\ 0{,}7071)^T$;

$x(3) = (0{,}6;\ -0{,}8)^T$; $x(4) = (0{,}1961;\ -0{,}9806)^T$;

$x(5) = (-0{,}9806;\ -0{,}1961)^T$; $x(6) = (-0{,}9806;\ 0{,}1961)^T$.

Випадковим способом обрані нормалізовані початкові значення ваг дорівнюють

$w_1(0) = (1{,}000;\ 0{,}000)^T$;

$w_2(0) = (0;\ -1)^T$;

$w_3(0) = (-0{,}707;\ 0{,}707)^T$.

Розміщення векторів x й $w_i(0)$ зображено на рис. 11.5.

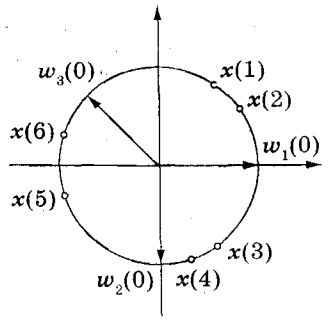


Рис. 11.5. Розміщення векторів x і $w(0)$

Подання мережі вектора $x(0)$ дає

$$w_1^T(0)x(1) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \\ -0{,}7 & 0{,}7 \end{bmatrix}^T \cdot \begin{bmatrix} 0{,}8 \\ 0{,}6 \end{bmatrix} = \begin{bmatrix} 0{,}8 \\ -0{,}6 \\ -0{,}14 \end{bmatrix}.$$

Оскільки $\max\left[w_1^T(0)x(1)\right] = 0{,}8$, переможцем буде перший нейрон. Скорегуємо його ваги відповідно до алгоритму (11.4), прийнявши $\alpha = 0{,}5$,

$$w_1(1) = w_1(0) + 0{,}5\left(x(1) - w_1(0)\right) = \begin{bmatrix} 1{,}0 \\ 0 \end{bmatrix} + 0{,}5\left(\begin{bmatrix} 0{,}8 \\ 0{,}6 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 0{,}9 \\ 0{,}3 \end{bmatrix}.$$

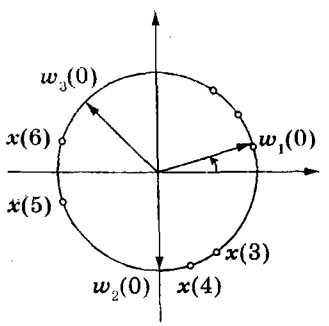


Рис. 11.6. Корекція вагового вектора w_1

Ваги змінилися таким чином, що вектор перемістився до вхідного вектора $x(1)$ (*див.* рис. 11.5). Оскільки порядок подання вхідних векторів довільний, то при поданні $x(4)$ отримуємо

$$w_1^T(1)x(4) = -0{,}11769;\ w_2^T(0)x(4) = 0{,}9806;\ w_3^T(0)x(4) = -0{,}8319.$$

Оскільки переможцем виявився другий нейрон, скорегуємо його вагові коефіцієнти, міняючи його розміщення (рис. 11.7) переміщенням до $x(4)$

$$w_2(1) = w_2(0) + 0{,}5\left(x(4) - w_2(0)\right) =$$

$$= \begin{bmatrix} 0 \\ -1 \end{bmatrix} + 0{,}5\left(\begin{bmatrix} 0{,}1961 \\ -0{,}9806 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \end{bmatrix} \right) = \begin{bmatrix} 0{,}1 \\ -0{,}99 \end{bmatrix}.$$

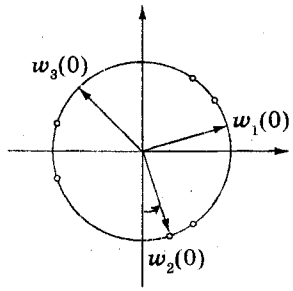


Рис. 11.7. Корекція вагового вектора w_2

Процес навчання завершиться, коли кожен ваговий вектор стане прототипом різних поданих мережі вхідних сигналів, тобто відповідатиме своїй групі вхідних сигналів. В остаточному підсумку вектори ваг займуть положення приблизно такі ж, які наведено на рис. 11.8.

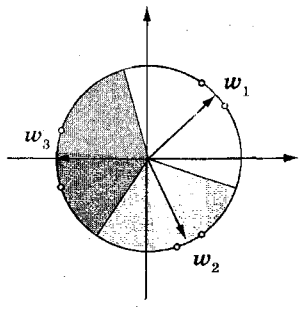


Рис. 11.8. Остаточне розміщення вагових векторів

Навчена в такий спосіб мережа може класифікувати будь-які подані на її вхід сигнали.

11.3. Вибір функції «сусідства»

На рис. 11.9 зображено шар нейронів з нейроном-переможцем, позначеним чорним кільцем. Оскільки ваги всіх затемнених нейронів змінюються по-різному, залежно від їхньої далекості від нейрона-переможця, найбільш простим є вибір як f_{ij} деякої величини, що дорівнює одиниці при $j = i$, меншої одиниці для затемнених нейронів, тобто нейронів, що лежать у безпосередній близькості від активованого нейрона, і нулю для інших, позначених світлими кружками.

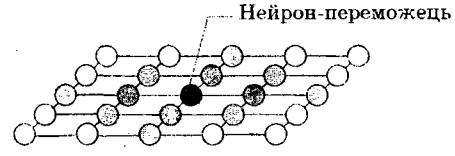


Рис. 11.9. Шар нейронів Когонена

На практиці ж як f_{ij} вибирають функції, що використовують евклідову метрику

$$d = \sum_k (r_{ik} - r_{jk})^2, \tag{11.6}$$

де r_{ik}, r_{jk} — координати i-го й j-го нейронів.

До найбільш часто використовуваних потенційних функцій відносяться:

а) дзвонувата функція Гаусса

$$f_{gauss1}(d,\sigma) = e^{-\frac{d^2}{2\sigma^2}}, \tag{11.7}$$

де σ^2 — дисперсія відхилення;

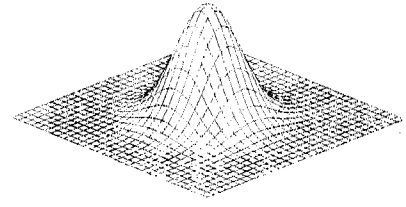


Рис. 11.10. Дзвонувата функція Гаусса

б) функція «мексиканський капелюх»

$$f_{gauss2}(d,\sigma) = (1-\left(\frac{d}{\sigma}\right)^2)\cdot e^{-\left(\frac{d}{\sigma}\right)^2}, \qquad (11.8)$$

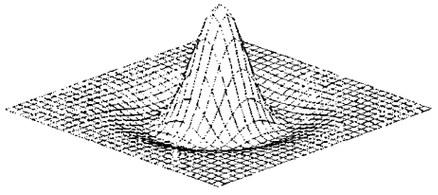


Рис. 11.11. Функція «мексиканський капелюх»

в) косинусоїдна функція

$$f_{\cos}(d,\sigma) = \begin{cases} \cos\left(\frac{d\pi}{2\sigma}\right), & d < \sigma; \\ 0, & d \geq \sigma; \end{cases} \qquad (11.9)$$

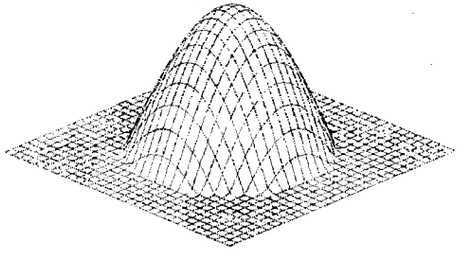


Рис. 11.12. Косинусоїдна функція

г) конусоподібна функція

$$f_{cone}(d,\sigma) = \begin{cases} 1-\frac{d}{\sigma}, & d < \sigma; \\ 0 & d \geq \sigma; \end{cases} \qquad (11.10)$$

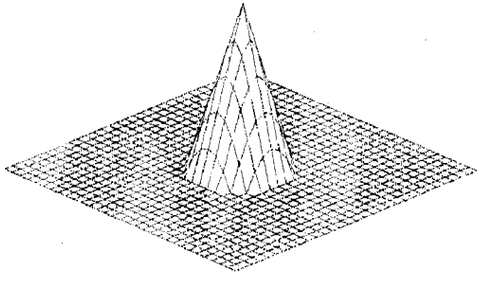


Рис. 11.13. Конусоподібна функція

д) циліндрична функція

$$f_{cylinder}(d,\sigma) = \begin{cases} 1, & \text{при } d < \sigma; \\ 0, & \text{при } d \geq \sigma; \end{cases} \qquad (11.11)$$

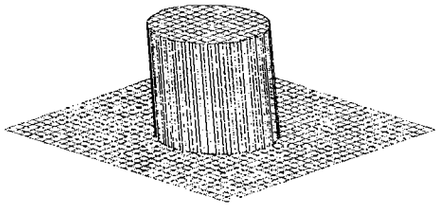


Рис. 11.14. Циліндрична функція

11.4. Побудова мапи Когонена

Як правило, спочатку будують досить грубу мапу (модель розбивання), поступово уточнюючи її в процесі навчання. Для цього необхідно повільно змінювати не тільки параметр α, але й, наприклад, параметр σ у формулі (11.7). Одним з ефективних способів зміни цих параметрів є такий:

$$\alpha(k) = \alpha(0)\left[\frac{\alpha_{min}}{\alpha(0)}\right]^{\frac{k}{k_{max}}} \qquad (11.12)$$

$$\sigma(k) = \sigma(0)\left[\frac{\sigma_{min}}{\sigma(0)}\right]^{\frac{k}{k_{max}}}, \qquad (11.13)$$

де $\alpha(0) \approx 0{,}8$; $\alpha_{max} << 1$; $\sigma(0) = 0{,}2$; $\sigma_{min} = 0{,}5$ — параметри, що визначають крутість функції f_{ij}; k_{max} — кількість ітерацій, що задаються.

На рис. 11.15 зображено процес побудови мапи Когонена, що являє собою двовимірну решітку, утворену шляхом з'єднання сусідніх нейронів, які перебувають у вузлах решіток. Виходячи з початкових умов і використовуючи алгоритм навчання (11.4), мережа в міру збільшення кількості навчальних вхідних образів розвивається й набуває вигляду решіток. Внизу кожного рисунка зображено кількість образів, на основі яких отримана відповідна мережа [55].

При реалізації мережі Когонена виникають такі проблеми:

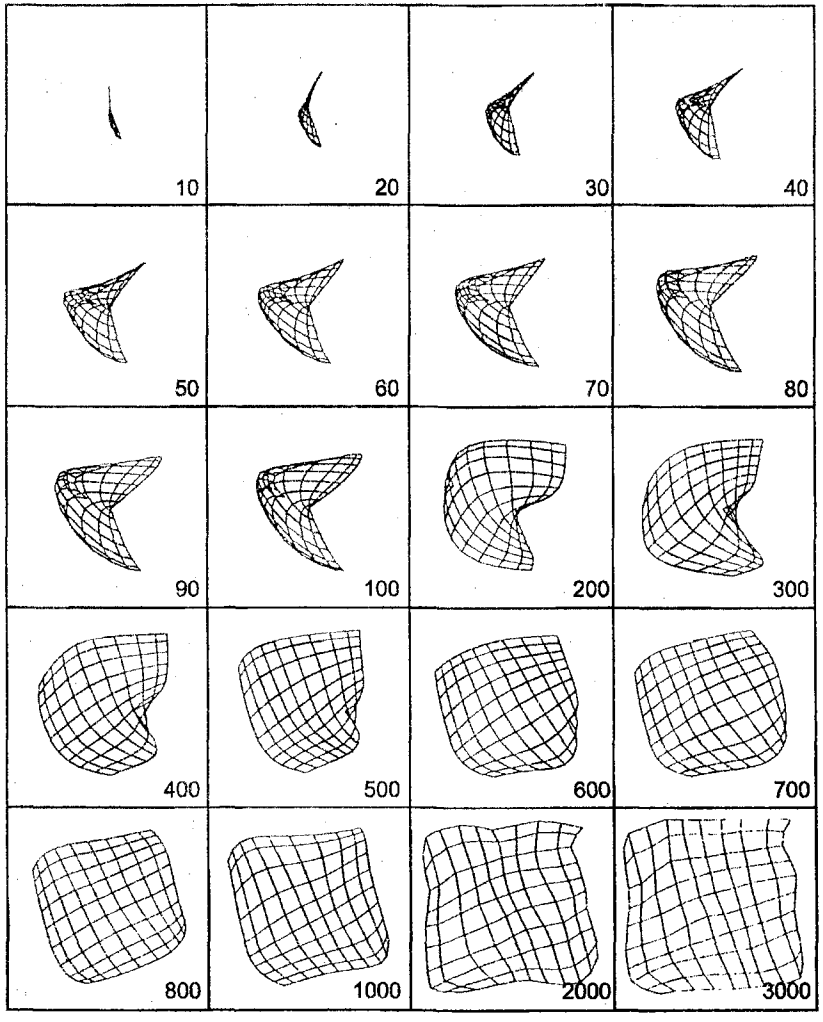


Рис. 11.15. Побудова мапи Когонена

1. *Вибір коефіцієнта навчання* α.

Цей вибір впливає як на швидкість навчання, так і на стійкість отримуваного розв'язку. Очевидно, що процес навчання прискорюється (швидкість збіжності алгоритму навчання збільшується) при виборі α близьким до одиниці. Однак у цьому випадку подання мережі різних вхідних векторів, що відносяться до одного класу, призведе до змін відповідного вектора вагових коефіцієнтів. Навпаки, при $\alpha \to 0$ швидкість навчання буде повільною, однак вектор вагових коефіцієнтів за умови досягнення центра класу при подачі на вхід мережі різних сигналів, що відносяться до одного класу, залишатиметься поблизу цього центра.

Тому одним зі шляхів прискорення процесу навчання при одночасному забезпеченні отримання стійкого розв'язку є вибір змінного α з $\alpha \to 1$ на початкових етапах навчання й $\alpha \to 0$ — на завершальних. На жаль, такий підхід не може бути застосований у тих випадках, коли мережа має безупинно підлаштовуватися до поданих їй нових вхідних сигналів.

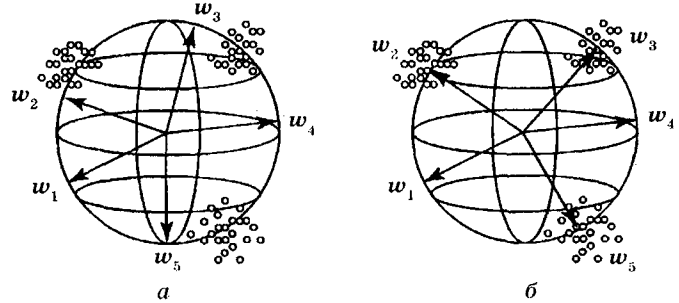
2. *Рандомізація ваг.*

Рандомізація ваг шару Когонена може породити серйозні проблеми під час навчання, оскільки в результаті цієї операції вагові вектори розподіляються рівномірно по поверхні гіперсфери. Як правило, вхідні вектори розподілені нерівномірно й групуються на відносно малій частині поверхні гіперсфери. Тому більшість вагових векторів виявляться настільки віддаленими від будь-якого вхідного вектора, що не будуть активовані й стануть марними. Більш того, активованих нейронів, які залишилися, може виявитися занадто мало, щоб розбити близько розташовані вхідні вектори на кластери.

3. *Вибір початкових значень векторів вагових коефіцієнтів і нейронів.*

Якщо початкові значення обрані невдало, тобто розташованими далеко від поданих вхідних векторів, то нейрон не виявиться переможцем ні при яких вхідних сигналах, а, отже, не навчиться.

На рис. 11.16 наведено випадок, коли початкове розташування ваг $\boldsymbol{w}_1$ й $\boldsymbol{w}_4$ призвело до того, що після навчання мережі всі пропоновані образи класифікують нейрони з вагами $\boldsymbol{w}_2$, $\boldsymbol{w}_4$ і $\boldsymbol{w}_5$. Ваги ж $\boldsymbol{w}_1$ й $\boldsymbol{w}_2$ не змінилися й можуть бути вилучені з мережі (більш докладно про це *див.* розділ 19).



а *б*

Рис. 11.16. Початкове (*а*) і кінцеве (*б*) розміщення векторів ваг

4. *Вибір параметра відстані* σ.

Якщо спочатку параметр σ обраний малим або дуже швидко зменшується, то далеко розміщені один від одного нейрони не можуть впливати один на одного. Хоча дві частини в такій мапі настроюються правильно, загальна мапа матиме топологічний дефект (рис. 11.17) [55].

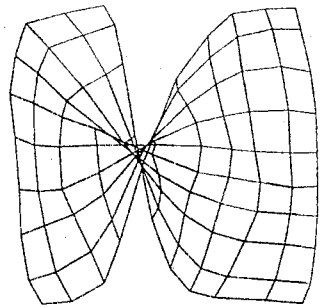


Рис. 11.17. Топологічний дефект мапи Когонена

5. *Кількість нейронів у шарі.*

Кількість нейронів у шарі Когонена має відповідати кількості класів вхідних сигналів. Це може бути неприпустимо в тих задачах, коли кількість класів заздалегідь невідома.

6. *Класи вхідних сигналів.*

Шар Когонена може формувати тільки класи, що являють собою опуклі області вхідного простору.

Контрольні запитання

1. Яка структура мережі Когонена? У чому відмінність мережі Когонена від мережі ВК?
2. Поясніть механізм навчання мережі Когонена.
3. Що таке функція «сусідства»? Наведіть приклади вибору цієї функції.
4. Які проблеми виникають при побудові мапи Когонена?

12. МЕРЕЖІ ЗУСТРІЧНОГО ПОШИРЕННЯ

Метод навчання ШНМ, що отримав назву *«зустрічне поширення»* (*Counterpropagation*), запропонований і розвинений у працях *Р. Гехт-Нільсена* [27, 28]. Він є гібридом методів навчання двох різних ШНМ: *Когонена* й *Ґроссберґа*. Даний метод має такі властивості, що роблять його досить привабливим:

- малий час навчання (особливо порівняно з методом зворотного поширення помилки);
- можливість працювати як з неперервними, так і з двійковими сигналами;
- комбінування двох видів мереж (Когонена й Ґроссберґа) забезпечує отримання таких нових властивостей, яких немає ні в одній мережі, ні в іншій;
- блокова побудова мережі шляхом каскадування спеціалізованих модулів, що підтверджує можливість успішного створення складної ШНМ із простих компонентів.

У літературі термін *«counterpropagation»* не є однозначним. Існує два різних варіанти мереж, що називаються мережами зустрічного поширення:

1. Двошарова мережа прямого поширення, що складається із шару Когонена й шару Ґроссберґа.

2. Двошарова мережа прямого поширення, що складається з самоорганізувальної мапи Когонена й шару Ґроссберґа.

Відмінність цих варіантів полягає в організації прихованого шару, і, відповідно, у способі навчання його нейронів. Якщо в першому варіанті зв'язку між сусідніми нейронами шару Когонена відсутні, то в другому такому зв'язку наявні, і тому нейрони прихованого шару навчаються як самоорганізувальна мапа Когонена.

Найбільшого поширення мережі даного типу отримали в задачах апроксимації функцій, розпізнавання, обробки й відновлення зображень і класифікації.

Розглянемо перший варіант мережі.

12.1. Архітектура мережі зустрічного поширення

Мережа зустрічного поширення, спрощену структуру якої зображено на рис. 12.1, містить вхідний шар, прихований, що є шаром Когонена (K), і вихідний, що є шаром Ґроссберґа (G).

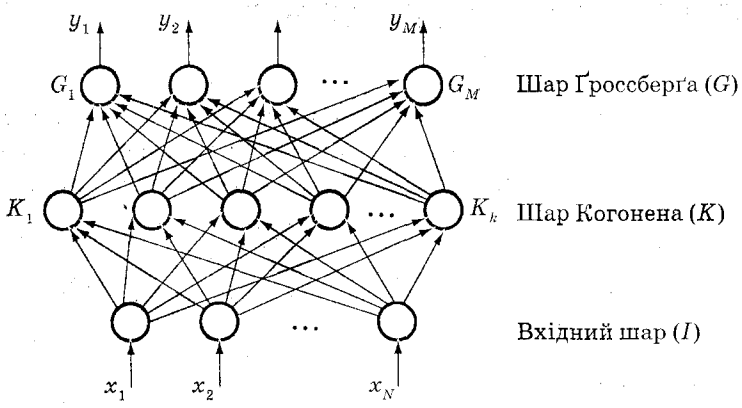


Рис. 12.1. Спрощена структура мережі зустрічного поширення

Усі нейрони вхідного шару пов'язані вагами w_{ij} з нейронами шару K. Кожен нейрон Когонена пов'язаний з кожним нейроном шару G вагами v_{ij}.

Працює мережа у такий спосіб.

Вхідні сигнали, що відповідають певному образу, подаються на всі нейрони шару K, для кожного з яких обчислюється зважена сума

$$z_j = \sum_{i=1}^{N} w_{ij} x_i, \quad j = \overline{1,k}. \tag{12.1}$$

У шарі K реалізується принцип, *переможець отримує все* (*winner takes all*), тобто збуджується тільки той нейрон, для якого величина z_j (12.1) буде найбільшою. На виході цього нейрона з'являється одиничний сигнал, на виходах інших — нульові. За наявності декількох нейронів з однаковим z_j серед них визначається переможець (випадковим способом або за будь-яким іншим принципом). Отже,

$$y_j' = \begin{cases} 1, \text{ якщо } z_j = \max(z_e); \\ 0 \text{ в іншому випадку.} \end{cases} \tag{12.2}$$

При практичній реалізації цей нейрон-переможець визначається шляхом перегляду (аналізу) всіх нейронів шару K.

Зважені з вагами v_{jg} $(j = \overline{1,k};\ g = \overline{1,M})$ вихідні сигнали нейронів шару K подаються на входи нейронів шару G

$$z_g' = \sum_{j \in K} y_j' v_{jg}, \quad g = \overline{1,M}. \tag{12.3}$$

Оскільки за аналогією із вхідним шаром у вихідному шарі не використовуються функції активації, на виході мережі (виходах нейронів шару G) з'являються сигнали

$$y_g = z_g', \quad g = \overline{1,M}. \tag{12.4}$$

А оскільки тільки один нейрон шару K, наприклад i-й, видає одиничний сигнал, а інші нульовий, то сума (12.3) спрощується

$$y_i = y_i' v_{ij} = v_{ij}. \tag{12.5}$$

Отже, на виході j-го нейрона шару G з'являється сигнал, рівний v_{ij}, тобто рівний вазі зв'язку між цим j-м нейроном і єдиним активним нейроном шару K.

12.2. Навчання шару Когонена

Мережа Когонена докладно розглянута в розділі 11. Тут же ми коротко зупинимося на застосуванні цієї мережі у вигляді шару ШНМ зустрічного поширення.

Шар K класифікує (розподіляє) по класах вхідні вектори, для чого його ваги мають бути настроєні так, щоб подібні вхідні сигнали активували однакові нейрони даного шару. Шар G на підставі цього повинен видати необхідний вихідний сигнал. Корекція ваг шару K здійснюється шляхом самонавчання.

Процес навчання в цьому шарі відбувається після того, як проведена попередня обробка вхідних сигналів й обрані початкові значення (ініціалізовані) елементи вагової матриці W.

12.2.1. Нормування входів

Попередня обробка вхідних сигналів звичайно полягає в їхній нормалізації, здійснюваній відповідно до формули (11.2)

$$\hat{x}_i = \frac{x_i}{\sqrt{x_1^2 + x_2^2 + \ldots + x_N^2}}, \quad i = \overline{1,N}. \tag{12.6}$$

12.2.2. Корекція вагових коефіцієнтів

У процесі навчання нейронів шару Когонена вхідний сигнал мережі зважується відповідно до (12.1) і визначається нейрон-переможець (12.2). Зазначимо, що внаслідок нормалізації векторів x й w_i величина z_j (12.1), що є їхнім скалярним добутком, може бути мірою подібності цих векторів. Максимальне значення z_j, що визначає нейрон-переможець, свідчить про те, що ваги w_{ij} саме цього нейрона найбільшою мірою відповідають поданому вхідному сигналу. Тому вагові коефіцієнти нейрона-переможця корегуються за правилом

$$w_{ij}(k+1) = w_{ij}(k) + \alpha(k)(x_i(k) - w_{ij}(k)), \qquad (12.7)$$

що відповідає обертанню вагового вектора $\boldsymbol{w}_j$ у напрямку вектора $\boldsymbol{x}$ (*див.* рис. 10.1).

Ефект навчання полягає в тому, що вектор ваг $\boldsymbol{w}_j(k)$ переміщується у бік $\boldsymbol{x}$ без істотної зміни його довжини (відбувається деяке скорочування вектора внаслідок $w_j(k+1) < 1$, але це скорочування незначне, оскільки $\boldsymbol{x}$ й $\boldsymbol{w}_j$ розташовані близько один від одного). Коефіцієнт α вибирають спочатку порядку 0,7 і поступово зменшують у процесі навчання. При подачі тільки одного вхідного вектора на нейрон Когонена, як можна побачити, у випадку $\alpha = 1$ навчання відбувається відразу, тому що $\boldsymbol{w}_j(k+1) = \boldsymbol{x}(k)$. Зазвичай ряд векторів, що навчають, містить багато близьких векторів, і мережа повинна для всіх цих векторів активувати однакові нейрони. Це означає, що вектор ваг нейронів має бути розташований посередині кожного класу (*див.* розділ 11). Останнє досягається використанням малих значень α, що зменшують вплив окремих вхідних векторів.

12.2.3. Ініціалізація елементів вагової матриці

Звичайно початковим значенням векторів ваг нейронів шару K ставлять у відповідність випадкові значення. Крім того, їх нормують, оскільки після навчання вони мають збігатися з відповідними нормованими вхідними векторами. При початковому нормуванні вони перебувають ближче до свого кінцевого стану, що до того ж скорочує процес навчання.

Однак при такому випадковому заданні векторів можуть виникнути серйозні проблеми. Наприклад, у багатовимірному просторі при нерівномірному розподілі векторів може виявитися, що вони згрупуються в деякій малій області. Вектори всередині цієї групи будуть розташовані настільки далеко від входів, що ніколи не наблизяться до них, оскільки постійно на їхньому виході буде нульовий сигнал. Інших же векторів, які розташовані ближче, буде недостатньо, щоб навчитися всім поданим образам (*див.* розділ 11).

Вибір початкових значень ваг пов'язаний ще з наступним.

При правильному розпізнаванні вхідних векторів, розташованих близько, але що відносяться до різних класів, мають бути активовані різні нейрони шару K, тобто щільність вагових векторів у цій області має бути великою. Якщо ж близько розташовані вхідні вектори відносяться до одного класу, то при їхній подачі має бути активований той самий нейрон шару Когонена.

Тому дуже важливим є початковий розподіл вагових векторів відповідно до щільності вхідних векторів. На практиці використовуються такі методи.

1. *Комбінаційний метод*

Усім вагам привласнюють значення $N^{-\frac{1}{2}}$, де N — кількість нейронів вхідного шару (що дорівнює кількості компонентів вагового вектора). При цьому всі вони мають одиничну довжину та є ідентичними. Потім подаються x_i й обчислюють

$$\hat{x}_i = \alpha x_i + \frac{1}{\sqrt{N}}(1-\alpha), \qquad (12.8)$$

де $0 < \alpha < 1$. Спочатку вибирають значення α невеликими, поступово збільшуючи їх у процесі навчання до 1. Завдяки цьому спочатку всі вхідні вектори мають одиничну довжину й збігаються із ваговими векторами. Із підвищенням α відбувається поділ вхідних векторів і повільне встановлення їхніх дійсних значень. Вагові вектори, повертаючись, прямують за вхідними векторами. Цей метод досить ефективний, однак вимагає для своєї реалізації багато часу.

2. *Введення шуму у вхідні вектори*

Відбуваються випадкові флуктуації вхідних векторів, що визначають зрештою ваговий вектор. Однак цей метод є ще повільнішим, ніж попередній.

3. *Початкова зміна ваг усіх нейронів шару K*

Відповідно до цього методу спочатку змінюються ваги не тільки нейрона-переможця, але й усіх інших нейронів, що наближає всі ваги до вхідного вектора. Потім у процесі навчання змінюються ваги тільки тих нейронів, які перебувають на певній відстані від нейрона-переможця. Далі ця відстань зменшується доти, поки не виявиться, що рухається ваговий вектор тільки нейрона-переможця.

4. *Використання механізму пам'яті*

Кожен нейрон має пам'ять (*conscience factor*), і нейрон-переможець, що найчастіше корегує свої ваги (наприклад, шляхом використання алгоритму (10.9)), штрафується, даючи тим самим можливість перемогти іншим нейронам.

На завершення необхідно зазначити таке. У розглянутому вище випадку при подачі кожного вхідного сигналу активувався тільки один нейрон Когонена. Однак можлива така реалізація мережі, коли вихідний сигнал шару K формують p нейронів цього шару, що мають найвищу активність. Виходи всіх нейронів цієї групи є нормованим вектором

$$\hat{y}_i = \frac{y_i}{\sqrt{y_1^2 + \ldots + y_p^2}}, \qquad (12.9)$$

що передається на шар G. Виходи інших нейронів K є нульовими.

Перевага такого підходу полягає в більш високій точності результату (наприклад, під час розгляду складних образів). Труднощі ж реалізації полягають у тому, що число нейронів-переможців p залежить від розв'язуваної задачі й не може бути заздалегідь визначено.

12.3. Навчання шару Ґроссберґа

На відміну від шару K нейрони шару G навчаються із учителем (*supervised learning*).

Після визначення нейрона-переможця й навчання шару Когонена відбувається навчання шару Ґроссберґа, у процесі якого корегуються елементи вагової матриці V, що пов'язує виходи шару із входами шару G, за алгоритмом

$$v_{ji}((k+1) = v_{ji}(k) + \gamma(k) y_j'(k)(y_i^*(k) - v_{ji}(k)), \qquad (12.10)$$

де $y_j'(k)$ — значення вихідного сигналу нейрона-переможця; $y_i^*(k)$ — необхідне значення вихідного сигналу i-го нейрона шару Ґроссберґа (мережі); $\gamma(k)$ — коефіцієнт підсилення, що змінюється в часі (аналогічний $\alpha(k)$ у (12.7)).

Змінюються ваги тільки тих нейронів шару G, які отримують відмінний від нуля сигнал нейрона шару K. Коефіцієнт навчання γ спочатку задають $\gamma \approx 0{,}1$ і поступово зменшують при навчанні. Завдяки цьому вагові коефіцієнти шару G повільно збігаються до середнього значення бажаних виходів y^*.

12.4. Повна мережа зустрічного поширення

Повна мережа зустрічного поширення, яку наведено на рис. 12.2, має унікальні апроксимуючі можливості.

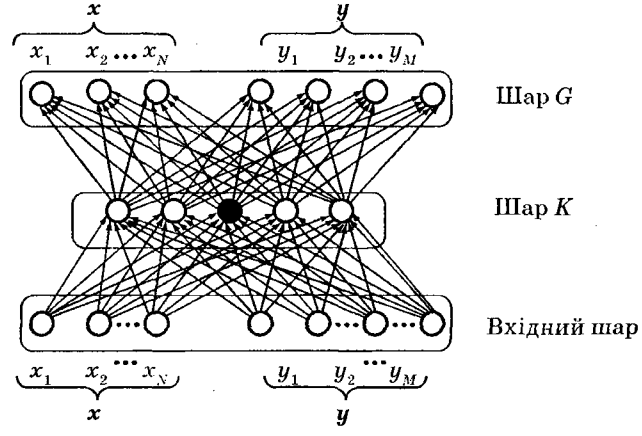


Рис. 12.2. Повна мережа зустрічного поширення

Для реалізації залежності $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ на вхід подаються нормалізовані вектори x й y і бажаний вихід, тобто вектори (x, y) мають перетворитися мережею у вектори $(\boldsymbol{x}, \boldsymbol{y}^*)$. У результаті навчання мережа вироблятиме копії $(\boldsymbol{x}, \boldsymbol{y})$. Цікавою є властивість мережі, яка полягає в тому, що при подачі на вхід тільки $\boldsymbol{x}$, тобто $(\boldsymbol{x}, 0)$, на виході з'являється як $\boldsymbol{x}$, так й $\boldsymbol{y}$, оскільки при сигналі $(\boldsymbol{x}, 0)$ буде

активований той самий нейрон Когонена, що й при подачі (x, y). Отже, якщо f — функція, що перетворить x в y, мережа апроксимує її. Якщо існує зворотна функція f^{-1}, то при подачі на вхід вектора y на виході з'являється x, тобто $f^{-1}(y) = x$ тому, що при $(0, y)$ активується знову той самий нейрон Когонена. Це означає, що мережа зустрічного поширення може одночасно апроксимувати як функцію f, так і зворотну функцію f^{-1}.

Контрольні запитання

1. Які основні властивості мережі зустрічного поширення?
2. Яка структура мережі?
3. У чому полягає навчання шару Когонена?
4. У який спосіб здійснюється ініціалізація елементів вагової матриці нейронів шару Когонена?
5. Як відбувається навчання шару Ґроссберґа?
6. У чому полягають апроксимуючі можливості повної мережі зустрічного поширення?

13. МАШИНА БОЛЬЦМАНА

Мережа Гопфілда збігається до локального екстремуму, що може відрізнятися від глобального. Якщо скористатися введеним раніше поняттям енергетичного ландшафту, то це відповідає застряганню кульки, пущеної з гори, яка прагне зайняти місце з мінімальною енергією в деякій западині (рис. 13.1, *а*).

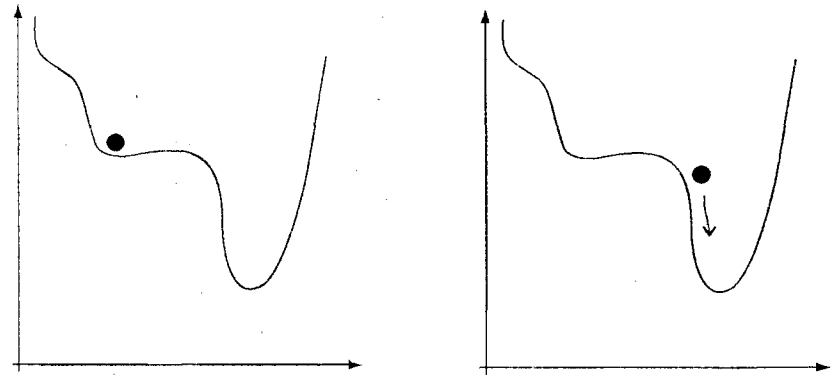


Рис. 13.1. Локальний і глобальний екстремуми

Для того, щоб кулька могла проскочити локальні екстремуми й досягти глобального, їй необхідно було б додати деяку внутрішню енергію.

З рис. 13.1, *б* видно, що для того щоб вийти з локального екстремуму, кулька має перескочити локальний максимум, що супроводжується зростанням його енергії. А оскільки рух кульки відбувається в багатовимірному просторі, виникає запитання: з якою силою й у який бік її слід виштовхувати? Очевидно, що тут, по-перше, має бути присутнім елемент випадковості й, по-друге, швидкість кульки має зменшуватися в міру її наближення до

глобального екстремуму. Ця ідея й реалізована в *машині Больцмана*, що поєднує детерміновану мережу Гопфілда з імовірнісним правилом навчання.

13.1. Структура машини Больцмана

Цей тип ШНМ запропоновано і досліджено у роботах [96–98]. Структуру цієї мережі зображено на рис. 13.2.

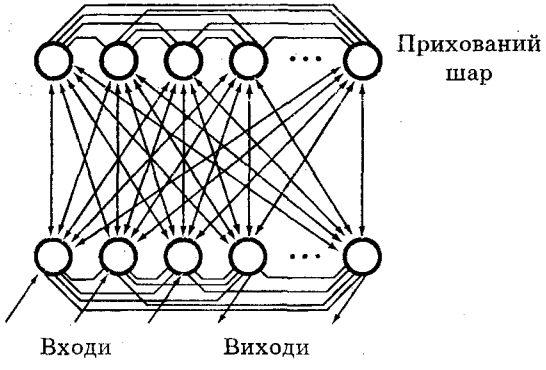


Рис. 13.2. Структура машини Больцмана

Поряд зі спостережуваними елементами машина Больцмана може містити й приховані, які виконують функції стохастичних детекторів ознак і поліпшують завдяки цьому властивості мережі.

У загальному випадку не всі елементи ШНМ пов'язані між собою, тобто в матриці ваг можуть бути й нульові елементи. Як і в мережі Гопфілда, діагональні елементи є нульовими $w_{ij} = 0$, а ненульові — недіагональні симетричні: $w_{ij} = w_{ji}$.

Мережевий вхід для j-го елемента машини Больцмана може бути поданий виразом

$$X_j = \sum_{i=S_j} w_{ij} y_i - \theta, \qquad (13.1)$$

де w_{ij} — вага зв'язку i-го й j-го елементів; y_i — двійкова вихідна функція активації j; θ — поріг (може дорівнювати нулю); S_j — множина елементів мережі, пов'язаних з j-м.

Значення y_i є бінарними (0, 1) або біполярними (−1, 1) і задаються у такий спосіб:

$$y_i = \begin{cases} +1 & \text{з імовірністю } p(x_j); \\ -1 & \text{з імовірністю } 1 - p(x_j). \end{cases} \qquad (13.2)$$

При цьому приймається, що випадкові величини x мають сигмоїдальну функцію розподілу

$$p(x) = \frac{1}{1 + e^{-\frac{x}{T}}}. \qquad (13.3)$$

Параметр T, що міститься у даному виразі, є керованим параметром «температури», що адаптується під час роботи мережі.

13.2. Навчання в машині Больцмана

Як ми вже зазначали, процес навчання цієї ШНМ є стохастичним. Значення вагових коефіцієнтів вибираються випадковим способом і для їхнього настроювання застосовується модифіковане правило Гебба. Процес навчання в машині Больцмана заснований на використанні моделі, що називається в термодинаміці «*моделлю відпалювання*» (*annealing*).

13.2.1. Модель відпалювання

Математична *модель відпалювання* металу розроблена в 50-ті роки XX ст. й описує такі процеси.

Розплавлений метал повільно охолоджують, знижуючи його температуру, у результаті чого частки металу впорядковуються в певну кристалічну структуру. У процесі кристалізації метал з рідкої фази проходить неперервний ряд станів, кожний з яких характеризується своїм значенням енергії. Кінцева структура — єдиний кристал — відповідає стабільному мінімальному енергетичному стану. При високих температурах атоми можуть рухатися, що призводить до переходу в стани з більшими значеннями енергії. У процесі поступового охолодження металу виникають стани з усе більш низькими значеннями енергії. Температура має знижуватися повільно, щоб після охолодження твердий метал досягав рівноважного стану. В іншому випадку з'являються явища вкраплення порожнин, що порушують бажану структуру металу.

Важливим при цьому є якомога довше перебувати поблизу так званої температури фазового переходу для того, щоб було достатньо часу для формування правильної кристалічної структури.

Якщо позначити енергію твердої речовини в стані m як E_m, то термічна рівновага в цьому стані виникає з імовірністю, визначеною *розподілом Больцмана*

$$P_m = \frac{1}{Z(T)} e^{-\frac{E_m}{Tk_\beta}}, \qquad (13.4)$$

де $k_\beta = 1{,}38 \cdot 10^{-16}$ ерг/К (кельвін) — стала Больцмана; $Z(T) = \sum_m e^{-\frac{E_m}{Tk_\beta}}$ — функція, що визначає всі 2^N можливих станів бінарної моделі, що містить N часток і відіграє роль нормалізатора.

При моделюванні процесу охолодження, здійснюваному методом Монте-Карло, модель проходить ряд станів, кожному з яких відповідає певний напрямок усіх часток. Потім у стан вноситься збурювання шляхом зміни розміщення якоїсь випадковим способом обраної частки. У бінарній моделі це збурювання відповідає зміні бінарного стану. Якщо результуюче збільшення енергії ΔE при цьому буде негативним, переходять до нового стану. Якщо ж $\Delta E > 0$, вибирають новий стан, обумовлений імовірністю

$$P = \frac{1}{1 + e^{\frac{\Delta E}{Tk_\beta}}}. \qquad (13.5)$$

Внесення збурювань у систему триває до досягнення нею рівноважного стійкого стану.

З (12.5) видно, що при дуже високій температурі T і $\Delta E > 0$ імовірність досягнення стану, що характеризується більшим значенням енергії, дорівнює 0,5. З іншого боку, при $T \to 0$ $P \approx 1$, тобто невизначеність у системі зникає. Отже, при високих температурах можливий перехід з більш низького енергетичного стану в більш високий. Зі зменшенням температури ймовірність такого переходу зменшується.

Очевидно, система може встановлюватися в один із великої кількості глобальних енергетичних станів, розподіл яких задається розподілом Больцмана. Якщо позначити деякі енергетичні стани $E(\alpha)$ і $E(\beta)$, то їхні ймовірності визначаться відповідно до формули

$$P(\alpha) = k \cdot e^{-\frac{E(\alpha)}{T}}, \quad P(\beta) = k \cdot e^{-\frac{E(\beta)}{T}}.$$

Тоді

$$\frac{P(\alpha)}{P(\beta)} = \frac{e^{-\frac{E(\alpha)}{T}}}{e^{-\frac{E(\beta)}{T}}} = e^{-\frac{(E(\alpha)-E(\beta))}{T}}. \qquad (13.6)$$

Система перебуватиме в тепловій рівновазі, якщо ймовірності станів більше не змінюються. Нехай енергія стану α буде менше енергії стану β

$$E(\alpha) < E(\beta).$$

Тоді з (13.6) випливає, що

$$e^{-\frac{(E(\alpha)-E(\beta))}{T}} > 1,$$

тобто

$$P(\alpha) > P(\beta). \qquad (13.7)$$

Це означає, що при наближенні до теплової рівноваги найбільш імовірними є стани з низьким енергетичним рівнем. При високих температурах рівновага досягається швидко, однак ці рівноважні стани є нестійкими.

Як було показано вище, високі температури дозволяють практично з рівною ймовірністю переходити як у високоенергетичні стани, так і в низькоенергетичні. При зниженні температури ймовірність переходу з більш високих енергетичних мінімумів зменшується, однак імовірність зворотного переходу спадає ще швидше. В остаточному підсумку при низькій температурі система встановлюється в тепловій рівновазі. Вплив температури на ймовірність P, визначену формулою (13.5), для рідких значень ΔE наочно показано на рис. 13.3.

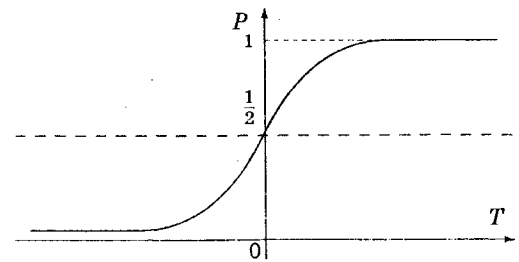


Рис. 13.3. Вплив температури на ймовірність переходу

Наведений опис імітованого відпалювання є значно спрощеним, оскільки насправді енергетичний ландшафт є простором великої розмірності, а енергетичний бар'єр між станами не має чітких меж. Це означає, що існує багато способів проходження від одного стану до іншого (їхня кількість зростає експоненціально зі збільшенням теплового шуму, що додається до системи).

Як зазначалося вище, важливе значення має швидкість зниження температури. Якщо температура знижується занадто швидко, система не має достатньої можливості вийти з локального енергетичного мінімуму й задовільний розв'язок може бути не досягнуто. Якщо ж температура знижується дуже повільно, система може вийти з локального мінімуму, але для цього їй знадобиться тривалий час, що також може виявитися неприйнятним.

Моделювання роботи машини Больцмана здійснюється аналогічно моделюванню процесу відпалювання, а оскільки вагова матриця є симетричною, аналіз її може бути проведений за аналогією з мережею Гопфілда з використанням енергетичної функції типу (5.5), що має мінімуми в стійких станах мережі. При цьому стани мережі є аналогами станів металу, що досягаються ним у процесі відпалювання, а енергетична функція — функцією вартості або критерієм класифікації. Параметр температури поступово (потактово) зменшується відповідно до обраної схеми охолодження до досягнення мережею деякого стійкого стану.

Нескладно показати, що машина Больцмана задовольняє рівнянню (13.5). Оскільки при переході мережі зі стану α у стан β стан k-го нейрона змінюється з x_k на x'_k, енергетична функція в станах α і β відповідно дорівнюватиме

$$E_\alpha = -\frac{1}{2}\sum_{i\neq k}\sum_{j\neq k} w_{ij}x_i x_j + \sum_{i\neq k}\theta_i x_i - \sum_i w_{ki}x_i x_k + \theta_k x_k, \qquad (13.8)$$

$$E_\beta = -\frac{1}{2}\sum_{i\neq k}\sum_{j\neq k} w_{ij}x_i x_j + \sum_{i\neq k}\theta_i x_i - \sum_i w_{ki}x_i x'_k + \theta'_k x'_k, \qquad (13.9)$$

а різниця $\Delta E_{\beta\alpha} = E_\alpha - E_\beta$ визначається виразом

$$\Delta E_{\beta\alpha} = -\sum_i w_{ki}x_i\left(x'_k - x_k\right) + \theta_k\left(x'_k - x_k\right). \qquad (13.10)$$

Внаслідок зміни стану k-го нейрона $x'_k \neq x_k$. У випадку бінарних значень при $x_k = 0$ стан $x'_k = 1$ буде досягнуто з імовірністю

$$P_{\alpha\to\beta} = \frac{1}{1+e^{-\left(\sum w_{ki}x_i - \theta_k\right)}}. \qquad (13.11)$$

При цьому зміна енергетичної функції відповідно до (13.10) дорівнюватиме

$$E_\beta - E_\alpha = -\sum_i w_{ki}x_i + \theta_k. \qquad (13.12)$$

Якщо ж змінні приймають біполярні значення (+1,–1), то

$$\Delta E_{\beta\alpha} = 2\left(\sum w_{ik}x_i - \theta_k x_k\right), \qquad (13.13)$$

а ймовірність переходу в новий стан β обчислюється за формулою (13.11) з $\Delta E_{\beta\alpha}$, визначеного співвідношенням (13.13). Очевидно, що формула (13.11) еквівалентна (13.5). Отже, у даної ШНМ розподіл Больцмана характеризує її стійкий статистичний стан, а процес навчання мережі називають *імітацією відпалювання*.

Навчання машини Больцмана є контрольованим і полягає в поданні мережі навчальних пар $\{\boldsymbol{x}^p, \boldsymbol{y}^p\}$ $p = \overline{1,P}$, що являють собою вхідні $\boldsymbol{x}^p$ і відповідні вихідні $\boldsymbol{y}^p$ образи. Слід зазначити, що мережа може навчатися як в *авто-*, так й у *гетероасоціативному* режимі. У першому випадку необхідний образ є вхідним, і мережа застосовується з метою знаходження збереженого образу, якщо поданий їй образ є спотвореним або містить у собі неточності. У другому випадку поданий (вхідний) і необхідний образи, як правило, різні.

13.2.2. Алгоритм навчання машини Больцмана

Алгоритм відпалювання — варіант ітеративного підходу до розв'язання оптимізаційних задач, у якому, як у фізичному відпалюванні, дозволяються кроки, що підвищують значення мінімізувального функціонала (енергії). «Теплові флуктуації», закладені в алгоритм, дають можливість не затримуватися в локальних екстремумах, завдяки чому може бути знайдений глобальний мінімум енергетичного рельєфу нейронної мережі.

На відміну від мережі Гопфілда, загальна кількість нейронів N машини Больцмана містить n видимих вхідних, m видимих вихідних й k прихованих нейронів ($N = n + m + k$). Надалі позначатимемо будь-який i-й з видимих нейронів безвідносно виконуваної їм функції як z_i ($i = 1, 2, \ldots, n + m$). Для зручності приймемо, що значення вхідних сигналів є біполярними (+1, –1). Слід відзначити також особливість процесу навчання даної ШНМ, що полягає в тому, що тут ще перед процедурою корекції матриці вагових коефіцієнтів необхідно здійснити ряд обчислень, пов'язаних із застосуванням алгоритму відпалювання для кожного поданого образу до досягнення мережею відповідного стійкого стану рівноваги, при якому

вихідні сигнали мережі збігатимуться з необхідними при заданому наборі вхідних сигналів. Всі стани рівноваги запам'ятовуються й використовуються для оцінювання їхніх імовірностей. Процес повністю повторюється, якщо при поданні вхідного сигналу стану вихідних нейронів мають змінитися або є «вільні» локальні мінімуми. Імовірності «вільних» рівноважних станів обчислюються на основі характеристик запам'ятованих мережею станів. Корекція ваг здійснюється шляхом мінімізації ентропії, що є мірою неузгодженості між бажаним і фактичним станом мережі. Весь процес повторюється доти, поки не будуть визначені рівноважні стани для всіх пар образів, що навчають.

Формально алгоритм навчання може бути описаний у такий спосіб:

1. Усім вагам мережі привласнюються невеликі, вибрані випадковим способом, початкові значення. Початкове значення температурного коефіцієнта T_0 вибирається досить великим.

2. На спостережувані нейрони мережі подаються навчальні пари із заданої множини образів $\{\boldsymbol{x}^p, \boldsymbol{y}^p\}$ доти, поки можливі зміни станів прихованих нейронів.

3. Випадковим способом вибирається деякий прихований нейрон l і змінюється його стан з x_l на x'_l, що призводить до зміни енергетичної функції на $\Delta E_l = E'_l - E_l$, визначеному виразом (9.13).

Якщо $\Delta E_l < 0$ (енергія зменшується), здійснюється довільна активація l-го нейрона незалежного від його попереднього стану.

Якщо $\Delta E_l > 0$ (енергія зростає) і

$$P_l = \frac{1}{1+e^{-\frac{\Delta E_l}{T}}} \geq P_0, \qquad (13.14)$$

де P_0 — деяка апріорно задана ймовірність, здійснюється активація l-го нейрона. Якщо $P_l < P_0$, стан l-го нейрона не змінюється.

4. Крок 3 повторюється для всіх m вихідних нейронів доти, поки в середньому стани всіх прихованих нейронів не змінилися хоча б один раз.

5. Значення температурного коефіцієнта зменшується відповідно до обраної схеми охолодження, наприклад

$$T(k+1) = \beta \cdot T(k), \qquad (13.15)$$

де $T(k)$ — температура на k-му такті; $0 < \beta < 1$ — коефіцієнт охолодження.

6. Кроки 3–5 повторюються до досягнення значення $T_{\min}$, при якому система перебуває в рівновазі, а енергія мінімальна.

7. Стани всіх прихованих нейронів, що відповідають рівноважному стану мережі для поданого p-го образу, запам'ятовуються у вигляді вектора r_c^p.

8. Після здійснення перерахованих вище кроків для всіх образів обчислюються оцінки r_{ij}^c кореляцій p_{ij}^c для всіх пар нейронів

$$r_{ij}^c = \frac{1}{p}\sum_{p=1}^{P} \varphi\left(z_i^p, h_{ci}^p\right), \quad i,j = \overline{1,N}, \quad i \neq j, \qquad (13.16)$$

де $\varphi(x,y) = \begin{cases} 1, & \text{якщо } x = y; \\ 0 & \text{в іншому випадку.} \end{cases}$

9. Кроки 2–8 повторюються для вільних вихідних нейронів, тобто не враховуються ті вихідні нейрони, які відповідають необхідним значенням цільової функції, й обчислюються оцінки r_{ij}^f кореляції p_{ij}^f пара вільних нейронів

$$r_{ij}^f = \frac{1}{p}\sum_{p=1}^{P} \varphi\left(z_i^p, h_{fi}^p\right), \quad i,j = \overline{1,N}, \quad i \neq j. \qquad (13.17)$$

10. Корегуються вагові коефіцієнти мережі за правилом

$$\Delta w_{ij} = \alpha\left[r_{ij}^c - r_{ij}^f\right], \quad i,j = \overline{1,N} \quad i \neq j, \qquad (13.18)$$

де α — параметр навчання.

11. Весь процес навчання повторюється для всіх i, j й p доти, поки необхідні зміни ваг (13.18) не стануть нульовими або досить малими.

13.2.3. Обговорення алгоритму навчання

Розглянемо деякі особливості описаного вище алгоритму навчання. Так, на другому кроці роботи алгоритму тому самому вхідному образу можуть відповідати кілька бажаних реакцій мережі. Така ситуація може виникати у випадку, якщо зв'язок між вхідними й вихідними образами не відомий або заданий нечітко, що характерно, наприклад, для експертних систем діагностики. У цьому випадку пари, що навчають, подаються багаторазово.

Крок 3 свідчить про можливості зростання енергії в ШНМ, що забезпечує проскакування локального екстремуму й гарантує збіжність до глобального мінімуму принаймні в асимптотиці. У роботі [100] показано, що при виборі схеми охолодження за правилом

$$T(k) \geq \frac{T_0}{1+\log k},$$

де T_0 — досить велика величина, алгоритм збігається в асимптотиці з імовірністю одиниця.

Хоча асимптотичні оцінки через скінченність інтервалу спостереження на практиці не досягаються, моделювання процесу навчання свідчить про отримання прийнятних результатів на кінцевих інтервалах при вдалому виборі схеми охолодження.

Із приводу вибору початкового значення температурного коефіцієнта слід зазначити таке. Це значення має бути значно більшим, щоб забезпечити приблизно однакову ймовірність усіх станів мережі. Потім значення параметра T має повільно зменшуватися, забезпечуючи тим самим зростання ймовірності переходу в низький енергетичний стан. При наближенні T до нуля ця ймовірність прямує до одиниці. Процес переходу в наступний стан з досягнутого рівноважного стану є ітераційним, причому досягнутий кінцевий стан мережі й відповідне значення T виступають як нові початкові умови. Істотним є досягнення рівноважного стану до зменшення температурного коефіцієнта, оскільки в іншому випадку глобальний мінімум досягнутим не буде.

Охолодження, здійснюване відповідно до (13.15), називається *експонентним*. Іншими найбільш використовуваними схемами є

— *лінійна*

$$T(k+1) = T(k) - \delta; \tag{13.19}$$

— *гіперболічна*

$$\frac{1}{T(k+1)} = \frac{1}{T(k)} + \delta. \tag{13.20}$$

Як свідчить практика, при виборі малих значень параметра навчання α правило корекції ваг (13.18) еквівалентне зростанню градієнта відносної ентропії H між упорядкованими й неупорядкованими (вільними) нейронами. Якщо позначити спільну ймовірність активних i-го й j-го нейронів, коли спостережувані нейрони впорядковані, як P_{ij}^c, а спільну ймовірність активних i-го й j-го нейронів, коли впорядковані тільки вхідні нейрони, а вихідні є вільними, як P_{ij}^f, то ваги мають бути підібрані так, щоб обидва розподіли мало відрізнялися один від одного. Один із можливих шляхів реалізації алгоритму настроювання ваг полягає в мінімізації відносної ентропії цих розподілів, визначеної за формулою

$$H\left(\frac{p^c}{p^f}\right) = \sum_{i,j} P_{ij}^c \log \frac{P_{ij}^c}{P_{ij}^f}. \tag{13.21}$$

При цьому P_{ij}^c характеризує бажані ймовірності стану мережі, а P_{ij}^f — дійсні. Ентропія (13.21) є невід'ємною величиною й дорівнює нулю при рівності ймовірностей P_{ij}^c й P_{ij}^f.

Відповідне правило корекції вагових коефіцієнтів w_{ij} набуває вигляду

$$\Delta w_{ij} = -\alpha \frac{\partial H}{\partial w_{ij}}. \tag{13.22}$$

Щодо використовуваної в (13.22) частинної похідної $\partial H / \partial w_{ij}$ слід зазначити, що імовірність P_{ij}^c, яка входить у H, не залежить від ваг w_{ij}, оскільки вхідні нейрони впорядковані відповідно до вхідних образів, а ймовірність P_{ij}^f залежить від ваг і для обчислень ваг користуються розподілом Больцмана (13.4).

Після навчання на заданій множині образів мережа може бути застосована для класифікації (розпізнавання) невідомих або спотворених образів. Алгоритм функціонування мережі при цьому буде дещо простіше розглянутого й міститиме, як й алгоритм навчання, елемент стохастичності.

13.2.4. Алгоритм класифікації (розпізнавання)

1. Усім вихідним сигналам випадковим способом привласнюються значення ±1 і задається велике (початкове) значення температурного параметра T_0. Подається вхідний образ $\boldsymbol{x}$. При цьому нейрони вихідного й прихованого шарів мають змінити свої стани.

2. Вибирається деякий j-й нейрон вихідного або прихованого шару й обчислюється за аналогією з (13.1) вхідний сигнал

$$x_i = \sum_{i \in S_i} w_{ij} y_i. \tag{13.23}$$

3. Незалежно від поточного стану стан j-го нейрона приймається рівним одиниці, якщо

$$P(x_j) = \left[1 + e^{\frac{-x_j}{T}}\right]^{-1} > P_0, \tag{13.24}$$

де P_0 — наперед задане значення ймовірності.

Якщо ж нерівність (13.24) не виконується, поточний стан j-го нейрона не змінюється.

4. Кроки 3 й 4 циклічно повторюються доти, поки всі нейрони вихідного й прихованого шарів хоча б один раз не змінили свого стану.

5. Цикл 4 повторюється багаторазово до досягнення рівноважного стану.

6. Зменшується температурний коефіцієнт відповідно до правила

$$T(k+1)=\beta T(k), \qquad (13.25)$$

де $\beta \in (0, 1)$,

$$T(k)=\frac{T_0}{1+\log(k)}.$$

7. Кроки 2–6 повторюються доти, поки мережа не досягне стійкого стану, що свідчить про розпізнавання мережею поданого вхідного образу.

На завершення необхідно зазначити таке. Будучи по суті стохастичною моделлю Гопфілда, машина Больцмана покликана забезпечити проскакування локального екстремуму й знаходження глобального. Хоча при цьому ймовірність знаходження такого екстремуму досить велика, не завжди ця ШНМ дозволяє його знайти. Це відбувається у тих випадках, коли глобальний екстремум відрізняється від локального незначною мірою (як, наприклад, у задачі про комівояжера).

Машина Больцмана викликає великий теоретичний інтерес, оскільки за її допомогою можна моделювати деякі властивості біологічних мереж.

Проте під час розв'язання практичних завдань нерідко переходять до простішої порівняно з машиною Больцмана *стохастичної мережі Гопфілда*, використовуючи для цього на вході моделі Гопфілда додатковий випадковий сигнал. Якщо цей сигнал має нормальний розподіл, то ця мережа називається *машиною Гаусса*.

Іншим способом спрощення мережі зі збереженням її позитивних властивостей є використання замість розподілу Больцмана (13.4) деякого іншого розподілу, зокрема, розподілу Коші. Отримувана при цьому мережа називається *машиною Коші*.

13.3. Машина Коші

Маючи ряд переваг порівняно з багатошаровим персептроном (у першу чергу — можливістю виходу з локальних екстремумів),

машина Больцмана, однак, не знайшла настільки значного поширення, як персептрон.

Непопулярність цієї ШНМ під час розв'язання практичних задач пояснюється істотними обчислювальними витратами, що супроводжують реалізації алгоритму відпалювання й пов'язаними з тим, що для досягнення мережею стану рівноваги необхідне проведення великої кількості обчислень для кожного значення температури.

Істотного спрощення алгоритму відпалювання вдається досягти, якщо замість розподілу Больцмана (13.4) використати *розподіл Коші*

$$p(x,\theta)=\frac{1}{\pi\left|1+(x-\theta)^2\right|}, \qquad (13.26)$$

де θ — центр симетричного розподілу.

ШНМ, що використовує цей розподіл, називається *машиною Коші*. Розподіл (13.26) відноситься до розподілів з важкими «хвостами», другі моменти яких є нескінченними. Однак використання (13.26) замість (13.4) в алгоритмі відпалювання дозволяє робити значні кроки при навчанні ШНМ, прискорюючи тим самим схему охолодження. При цьому значення температурного коефіцієнта $T(k)$ змінюється не відповідно до (13.19), а за правилом

$$T(k)=\frac{T_0}{1+k}, \qquad (13.27)$$

де T_0 призначається так само, як і в (13.19).

Хоча в машині Коші вдається істотно скоротити час навчання, на відміну від машини Больцмана й у порівнянні з еквівалентним багатошаровим персептроном, що навчається за допомогою алгоритму зворотного поширення, воно залишається дуже великим, що перевищує час навчання останнього в 100 разів [34].

Контрольні запитання

1. Розгляньте структуру машини Больцмана. У чому відмінність цієї мережі від мереж Хеммінга й Гопфілда?
2. Що таке модель відпалювання? Як ця модель використовується в алгоритмі навчання машини Больцмана?
3. Поясніть суть алгоритму навчання машини Больцмана.
4. Що являє собою машина Коші? У чому її відмінність від машини Больцмана?

14. СТОХАСТИЧНІ МЕРЕЖІ

Хоча ці мережі й називаються стохастичними (*probabilistic*), функціонування їх, на відміну від машини Больцмана (*див.* розділ 13), є детермінованим. Основним же їхнім призначенням є стохастичне моделювання.

Стохастичні мережі (*СМ*, *PNN*), введені *Д. Шпехтом* [101, 102], мають цілий ряд цікавих властивостей, що роблять їх особливо привабливими для розв'язання прикладних задач розпізнавання образів. До таких властивостей відносять:

- — високу швидкість навчання, що досягається за рахунок відмови від застосування ітеративного оптимізаційного процесу настроювання внутрішніх параметрів;
- — оптимальність, яка полягає в тому, що за наявності достатнього для конкретного завдання обсягу даних отримуваний розв'язок збігається з розв'язком, що дається байєсівським класифікатором;
- — можливість отримання ефективного розв'язку за наявності пропущених або зашумлених даних.

Обмежені донедавна можливості обчислювальної техніки (малий обсяг пам'яті й невисока швидкодія) були причиною недостатньо пильної уваги до цих мереж. Бурхливий розвиток обчислювальних засобів дав новий поштовх як до досліджень СМ, так і до розширення сфер їхнього застосування.

Запропонована Шпехтом мережа моделювала роботу байєсівського класифікатора, що забезпечує мінімум середнього ризику класифікації.

14.1. Байєсівський класифікатор

Робота класифікатора може бути подана у такий спосіб. Нехай заданий вхідний N-вимірний вектор $\boldsymbol{x} = (1, x_1, ..., x_N)^T$, що

описує об'єкти, які можуть бути віднесені до одного з k класів, характеризованих відповідно функціями розподілу ймовірностей $f_1(\boldsymbol{x})$, $f_2(\boldsymbol{x})$, ..., $f_k(\boldsymbol{x})$. Нехай p_0, p_1, ..., p_k — апріорні ймовірності приналежності вектора $\boldsymbol{x}$ до відповідного класу. Необхідно задати деякий критерій класифікації (функцію) $d(\boldsymbol{x}) = C_i\ (i = \overline{1,k})$, що характеризує правильність віднесення вхідного вектора $\boldsymbol{x}$ до одного із заданих класів $C_i\ (i = \overline{1,k})$. Як такий можуть бути взяті, наприклад, критерій мінімального ризику або ж вартісний критерій, що визначає штраф за невірний розв'язок. Нехай будь-яким способом задано функції втрат L_1, L_2, ..., L_k, що характеризують помилкове віднесення поданого вектора до одного з класів. При цьому функція втрат L_i відповідає випадку віднесення вектора $\boldsymbol{x}$ до класу $\boldsymbol{i}$, у той час як насправді він відноситься до класу j, тобто L_i відповідає $d(\boldsymbol{x}) = C_i$, $\boldsymbol{x} \in C_j$, $i \neq j$. Природно, що при правильній класифікації $L = 0$.

Байєсівський класифікатор заснований на порівнянні величин $p_1L_1f_1(\boldsymbol{x})$, $p_2L_2f_2(\boldsymbol{x})$, ..., $p_NL_Nf_N(\boldsymbol{x})$, а вхідний сигнал, що надходить, він відносить до того класу, для якого така величина буде найбільшою. Таким чином, вектор $\boldsymbol{x}$ буде віднесений до класу C_i, якщо

$$p_iL_if_i(\boldsymbol{x}) > p_jL_jf_j(\boldsymbol{x}) \text{ для } j = \overline{1,k},\ \ i \neq j. \tag{14.1}$$

Використання зазначеного правила припускає, що всі співмножники добутків $p_iL_if(\boldsymbol{x})$ відомі. На практиці ж усе відбувається дещо інакше. Якщо апріорні ймовірності $p_i\ (i = \overline{1,k})$ або відомі, або можуть бути достатньо просто оцінені, то функції розподілу ймовірностей $f_i(\boldsymbol{x})\ (i = \overline{1,k})$, як правило, невідомі, а їхнє оцінювання є досить складним завданням. Звичайно для спрощення припускають, що вони є нормальними й оцінюють параметри цих розподілів. Більш доцільним, однак, є застосування непараметричних методів оцінювання, зокрема *вікон* (*методу*) *Парзена* [103, 104].

14.2. Вікна Парзена

Вікна Парзена застосовуються для побудови сімейства оцінок, які задають у загальному випадку співвідношенням

$$f_n(\boldsymbol{x}) = \frac{1}{n\sigma}\sum_{i=1}^{n}\varphi\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{\sigma}\right). \tag{14.2}$$

Тут x_i — незалежні випадкові змінні, що мають ідентичні неперервні функції розподілу; $\varphi(y)$ — функції, що задовольняють умові

$$\int_{-\infty}^{\infty}|\varphi(y)|dy<\infty;\ \lim_{y\to\infty}|y\varphi(y)|=0;\ \int_{-\infty}^{\infty}\varphi(y)dy=1;$$

$\sigma=\sigma(n)$ — функція, що задовольняє умові

$$\lim_{n\to\infty}\sigma(n)=0;\ \lim_{n\to\infty}n\sigma(n)=\infty.$$

Е. Парзен показав [104], що оцінка (14.2) асимптотично збігається до розподілу, що лежить у її основі, якщо він є гладким і неперервним.

Графічно (рис. 14.1) це може бути проілюстровано у такий спосіб.

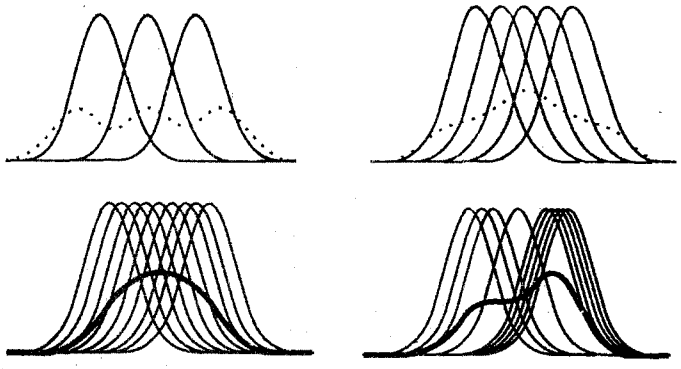


Рис. 14.1. Вікна Парзена

На осі відкладаються наявні значення випадкової величини x. Для кожного значення x будується функція щільності (наприклад, гауссівська, як на наведеному рисунку), максимум якої досягається при даному значенні x. Усереднення кривих дає нову криву (рис. 14.1, *а*) — усереднену функцію щільності, що із збільшенням кількості значень x, а відповідно й функцій щільності, може як завгодно точно апроксимувати дійсну функцію щільності.

Результат Парзена був узагальнений *Т. Какулосом* [105] на багатовимірний випадок. Для найчастіше використовуваного на практиці багатовимірного нормального розподілу оцінка (14.2) набуває вигляду

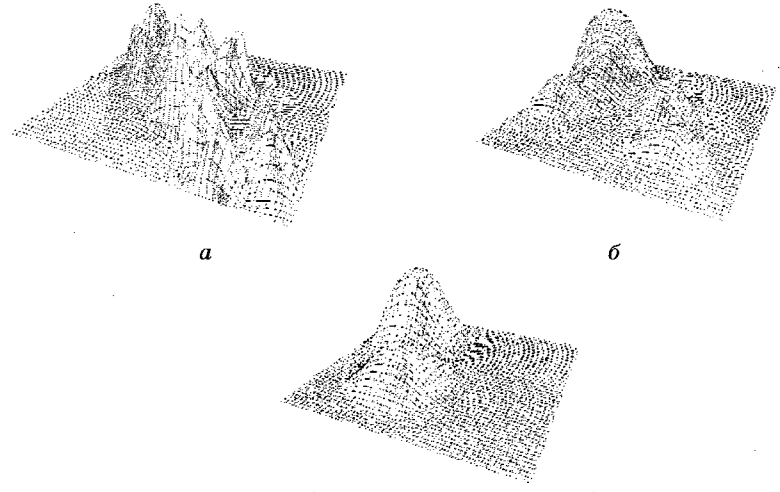
$$f_n(\boldsymbol{x})=\left(2\pi\sigma^2\right)^{-\frac{n}{2}}\frac{1}{n}\sum_{i=1}^{n}\exp\left(-\frac{\left(\boldsymbol{x}-\boldsymbol{x}_i\right)^T\left(\boldsymbol{x}-\boldsymbol{x}_i\right)}{2\sigma^2}\right). \qquad (14.3)$$

Стосовно завдання класифікації оцінка (14.3) може бути записана у такий спосіб:

$$f_i(\boldsymbol{x})=\left(2\pi\sigma^2\right)^{-\frac{n}{2}}k_i^{-1}\sum_{j=1}^{k_i}\exp\left(-\frac{\left(\boldsymbol{x}-\boldsymbol{x}_{ij}\right)^T\left(\boldsymbol{x}-\boldsymbol{x}_{ij}\right)}{2\sigma^2}\right), \qquad (14.4)$$

де n — розмірність випадкових змінних; k_i — кількість точок образів класу C_i; $\boldsymbol{x}_{ij}$ — j-й образ, що відноситься до класу i.

В оцінці (14.4) σ^2 є параметром згладжування, що визначається експериментально. Зазначимо, що величина σ^2 впливає на вид оцінки Парзена (*див.* рис. 14.2), однак сама оцінка не надто чутлива до цієї величини.



а *б* *в*

Рис. 14.2. Вплив величини σ^2 на вид оцінки Парзена:
а — мале σ^2; *б* — більше σ^2; *в* — велике σ^2

14.3. Структура стохастичної мережі

Запропонована *Шпехтом* [101] структура стохастичної ШНМ (рис. 14.3) є аналогом байєсівського класифікатора, що відновлює на основі (14.4) розподіл імовірностей поданих класів образів.

Застосування такого підходу дозволяє будувати нелінійні роздільні поверхні, що забезпечують високу якість класифікації.

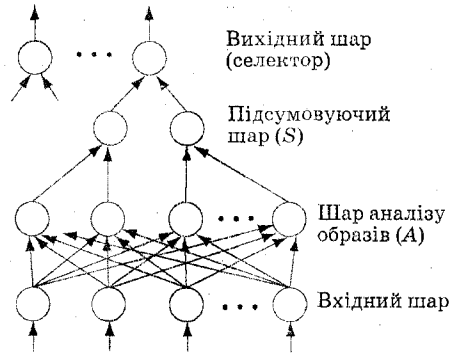


Рис. 14.3. Архітектура стохастичної мережі

Кількість нейронів вхідного шару відповідає розмірності вхідного векторного простору.

Усі подані мережі образи попередньо нормалізуються. Кількість нейронів шару аналізу образів відповідає кількості самих образів.

У режимі навчання встановлюються й запам'ятовуються елементи вагової матриці W. Так, якщо $\boldsymbol{x}_l \in C_i$ (l-й елемент множини образів належить класу C_i), то після завершення режиму навчання маємо $\boldsymbol{w}_i = \boldsymbol{x}_l$. Оскільки вектор $\boldsymbol{x}$ нормалізований, вектор $\boldsymbol{w}$ буде також нормалізованим. Після цього встановлюється зв'язок між l-м нейроном шару аналізу образів a_l й j-м нейроном підсумовуючого шару S_j. Таким чином, процес навчання мережі полягає в присвоєнні відповідних значень вхідних сигналів елементам вагової матриці W і встановленні зв'язків між нейронами шару аналізу образів і підсумовуючого шару.

У режимі класифікації на вхід мережі подається сигнал $\boldsymbol{x}$ і здійснюється активація нейронів шару аналізу образів. При цьому обчислюється зважена сума

$$z_j = \sum_{i=1}^{N} x_i w_{ij}.$$

При активації нейрона шару аналізу образів обчислюється величина

$$f(z_j) = \exp\left\{\frac{z_j - 1}{\sigma^2}\right\}. \qquad (14.5)$$

Кількість нейронів підсумовуючого шару відповідає кількості класів, що розрізняють.

Нейрони шару аналізу образів зв'язуються з нейронами підсумовуючого шару не довільно, а залежно від того, до якого класу відноситься даний образ. Нейрони підсумовуючого шару просто підсумовують відповідно до (14.4) m сигналів ($0 \le m \le M$, де M — кількість нейронів шару аналізу образів), що надходять на їхні входи , тобто кожен нейрон S_k цього шару обчислює величину

$$S_k = \sum_{p=1}^{m} f_p(z_j). \qquad (14.6)$$

А оскільки сигнали $f_p(z_j)$ відносяться до одного класу, то отримувані в результаті підсумовування вхідні сигнали нейрона цього шару будуть оцінками ймовірностей розподілу відповідних класів.

У загальному випадку вихідний шар є селектором, що вибирає нейрон підсумовуючого шару з максимальним значенням вихідного сигналу й відносить його до відповідного класу.

Приклад 14.1. Під час розв'язання задачі дихотомії, тобто розбивання образів на два класи, нейрон вихідного шару має два вхідних сигнали, що надходять із підсумовуючого шару. Ці нейрони обчислюють добутки вихідних сигналів підсумовуючого шару й вагових коефіцієнтів c_k (k = 1,2), визначених за формулою

$$c_k = \frac{p_2 L_2}{p_1 L_1}\frac{k_1}{k_2},$$

де k_1, k_2 — кількість образів, що належать до першого й другого класів відповідно.

Якщо $c_1 > c_2$, поданий образ відносять до першого класу, в іншому випадку — до другого. Архітектуру ШНМ для розв'язання задачі дихотомії наведено на рис. 14.4.

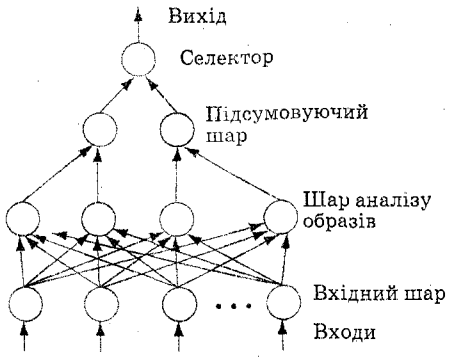


Рис. 14.4. Стохастична ШНМ для розв'язання задачі дихотомії

У цьому випадку підсумовуючий шар містить 2 нейрони, а вихідний — один (селектор).

14.4. Узагальнено-регресійна ШНМ

14.4.1. Рівняння регресії

Головна задача, розв'язувана за допомогою регресійного аналізу [83], — визначення математичних моделей досліджуваних об'єктів або явищ на основі експериментів або спостережень. Ці моделі є певними математичними співвідношеннями між деякими характеристиками спостережуваного об'єкта або явища $y_1, y_2, ..., y_m$ та їхніми обумовленими величинами, $x_1, x_2, ..., x_N$, що називаються відповідно залежними (вихідними) і незалежними (вхідними) змінними. Оскільки всі моделі через наявність у спостереженнях випадкових збурювань відбивають досліджуване явище з деяким наближенням, під моделлю розуміють співвідношення, що надає умовне математичне сподівання залежної змінної при заданих значеннях незалежних змінних

$$f(x_1, x_2, ..., x_N) = M\{y | x_1, x_2, ..., x_N\}, \qquad (14.7)$$

що називається *регресією* (*рівнянням регресії*). Співвідношення (14.7) показує зміну середнього значення залежної змінної об'єкта при змінах незалежних змінних.

Фактично вимірювана вихідна змінна y містить деяку перешкоду ξ (найчастіше припускають, що дія на об'єкт множини випадкових збурювань еквівалентна дії одного єдиного збурювання ξ), тобто рівняння (14.7) має вигляд:

$$y = f(x_1, x_2, ..., x_N) + \xi. \qquad (14.8)$$

Існує велика кількість регресійних моделей, визначених конкретним видом функції $f(x_1, x_2, ..., x_N)$ і значеннями використовуваних у них коефіцієнтів $\beta_1, \beta_2, ..., \beta_k$, які необхідно визначити за експериментальним даними. Залежно від того, як ці коефіцієнти входять у рівняння регресії, лінійно або нелінійно, розрізняють відповідно лінійні й нелінійні за параметрами моделі.

Найбільш дослідженою є модель *лінійної регресії*

$$y(t) = \beta^T x(t) + \xi(t), \qquad (14.9)$$

де $y(t)$ — вихідна (залежна) змінна; $x(t) = (x_1(t), x_2(t), ..., x_N(t))^T$ — вектор незалежних змінних $N \times 1$; $\beta = (\beta_1, \beta_2, ..., \beta_N)^T$ — вектор параметрів $N \times 1$, що підлягає визначенню; $\xi(t)$ — випадкова неспостережувана завада.

У класичному регресійному аналізі оцінювання вектора параметрів β (параметричне оцінювання) здійснюється шляхом мінімізації якогось наперед обраного опуклого (звичайно квадратичного) критерію оцінювання. Основним методом оцінювання в лінійній регресії є метод найменших квадратів (МНК).

Вибір лінійних моделей зовсім не є настільки обмежувальним, як це може здатися на перший погляд. Так, багато функцій декількох змінних є приблизно лінійними в досить малих областях або можуть бути приведені до лінійного за допомогою відповідного перетворення. Однак у ряді випадків таке спрощення не дозволяє досягти необхідного результату, зокрема при використанні регресійних моделей у задачах прогнозування. У цих випадках надзвичайно важливою є задача вибору виду функціональної залежності залежних й незалежних змінних, тобто завдання вибору структури моделі (14.8).

Задача параметричного оцінювання в нелінійній регресії істотно ускладнюється. Так, оцінка МНК для нелінійної регресії взагалі може не існувати, а може існувати кілька оцінок МНК, що призводять до того самого значення критерію, що мінімізується.

Використання вікон Парзена дозволяє ефективно вирішувати задачі побудови *нелінійної регресійної моделі*.

14.4.2. Узагальнено-регресійна мережа

Нехай $f(x,y)$ — спільна щільність розподілу випадкового вектора $\boldsymbol{x}$ і величини y. Регресія за визначенням є умовним середнім

$$M\{y \mid x\} = \frac{\int_{-\infty}^{\infty} yf(\boldsymbol{x},y)dy}{\int_{-\infty}^{\infty} f(\boldsymbol{x},y)dy}. \qquad (14.10)$$

Для оцінки невідомої функції $f(\boldsymbol{x},y)$ можуть бути застосовані різні методи. Одним із найбільш ефективних є непараметрична оцінка Парзена

$$f_k(\boldsymbol{x},y) = (2\pi\sigma^2)^{-\frac{N}{2}} k^{-1} \sum_{i=1}^{k} \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{2\sigma^2}\right) \exp\left(-\frac{\|y-y_i\|^2}{2\sigma^2}\right), \qquad (14.11)$$

де k — кількість образів; N — розмірність вектора i-го образу x_i; σ — згладжувальний параметр.

Із збільшенням кількості образів k точність оцінки (14.11) зростає.

Використання в (14.10) замість $f(\boldsymbol{x},y)$ оцінки (14.11) призводить до такої оцінки регресії [102]:

$$\hat{y} = \frac{\sum_{i=1}^{k} y_i \exp\left(-D_i^2\left(2\sigma^2\right)^{-1}\right)}{\sum_{i=1}^{k} \exp\left(-D_i^2\left(2\sigma^2\right)^{-1}\right)}, \qquad (14.12)$$

де $D_i^2 = \|\boldsymbol{x}-\boldsymbol{x}_i\|^2$.

Узагальнено-регресійна (*УР*) ШНМ (*Generalized Regression Neural Network, GRNN*) була запропонована *Шпехтом* [102] для побудови *узагальнених* (лінійних і нелінійних) *регресій*. На відміну від класичного регресійного аналізу, застосування цього підходу не вимагає задання функціональної залежності, що пов'язує вхідні й вихідні змінні. Мережа видає оцінку функції розподілу ймовірностей поданих їй образів.

Усі вхідні образи будь-яким способом розбивають на певну кількість кластерів. Якщо кількість образів невелика, кожен образ може утворити свій кластер. У загальному ж випадку кластери містять кілька образів. Вхідні вектори, як правило, нормують. Потім вибирається структура мережі й здійснюється її навчання,

як це було описано вище. Зазначимо, що оскільки співвідношення (14.12) описує оцінку для окремого образу, при побудові узагальненої регресії використовують оцінку

$$\hat{y} = \frac{\sum_{i=1}^{k} A_i \exp\left(-D_i^2\left(2\sigma^2\right)^{-1}\right)}{\sum_{i=1}^{k} B_i \exp\left(-D_i^2\left(2\sigma^2\right)^{-1}\right)}, \qquad (14.13)$$

де $A_i \equiv A_i(k) = A_i(k-1) + y_i$; $B_i \equiv B_i(k) = B_i(k-1) + 1$ є значенням коефіцієнтів для i-го кластера після k спостережень. Структура УР-мережі збігається зі структурою стохастичної мережі, зображеної на рис. 14.3.

Вхідні образи передаються вхідним шаром на шар аналізу образів, що містить k нейронів за кількістю кластерів. Елементи вагової матриці приймаються рівними відповідним компонентам вхідних векторів або відповідних координат векторів центрів кластерів. При поданні мережі вхідного образу $\boldsymbol{x}$ обчислюється відстань між цим образом і вектором вагових коефіцієнтів мережі, що є аргументом функції активації нейронів шару аналізу образів. Звичайно як функція активації застосовується експонентна (гауссівська) функція. У цьому випадку при використанні квадратичної функції відхилення ваг і входів мережі вихідні сигнали нейронів шару аналізу образів мають вигляд (14.5). Далі ці вихідні сигнали надходять у підсумовуючий шар, що містить нейрони двох типів.

Нейрони одного типу використовуються для подання «бажаної» регресії, на їхніх виходах формується сигнал $\hat{y}f(x)\alpha$, де $\hat{y}$ — оцінка середнього значення y для заданого $x(M\{y|x\})$; $f(x)$ — функція щільності x; α — деякий коефіцієнт, що залежить від використовуваних вікон Парзена. Ваги нейронів другого типу дорівнюють кількості образів в одному кластері. У цих нейронах підсумовуються $f(x)k$. У вихідному шарі здійснюється розподіл вихідних сигналів нейронів першого типу на вихідні сигнали нейронів другого типу, внаслідок чого отримуємо оцінку регресії (14.12) або (14.13).

Кілька слів про вибір параметра згладжування σ. Як видно з рис. 14.2, збільшення σ призводить до згладжування роздільних гіперповерхонь. У більшості випадків позитивних результатів вдається досягти при виборі $2 \le \sigma \le 6$ [102].

Слід зазначити, що процес навчання УР-мереж характеризується простотою й високою швидкістю. У цьому випадку не використовується потактова корекція (рекурентне настроювання)

елементів вагової матриці, оскільки мережа будує оцінку регресії на основі поданих образів. Точність оцінювання зростає із збільшенням кількості образів. Крім того, мережі даного типу можуть застосовуватися у процесі обробки інформації в реальному часі.

Контрольні запитання

1. Які основні властивості стохастичних мереж?
2. Поясніть роботу байєсівського класифікатора.
3. Для побудови яких оцінок використовуються вікна Парзена?
4. Як впливає параметр згладжування на вид оцінки Парзена?
5. Що являє собою структура стохастичної мережі?
6. Що називається регресією?
7. Яким співвідношенням описується непараметрична оцінка Парзена?
8. Які активаційні функції використовуються в узагальнено-регресійній мережі?
9. Які задачі можна вирішити за допомогою узагальнено-регресійної мережі?

15. РАДІАЛЬНО-БАЗИСНА МЕРЕЖА

Радіально-базисні мережі (*Radial Basis Function Nets, RBFN*) запропоновані для апроксимації функцій багатьох змінних [106–109]. Як зазначено в цих роботах, за допомогою радіально-базисних функцій можна як завгодно точно апроксимувати задану функцію. Як і багатошаровий персептрон, радіально-базисна мережа (РБМ) є *універсальним апроксиматором*.

Математичну основу РБ-мережі становить метод потенціальних функцій, розроблений *М. А. Айзерманом, Е. М. Браверманом* і *Л. І. Розоноером* [110], що дозволяє подати деяку функцію $y(\boldsymbol{x})$ у вигляді суперпозиції *потенціальних* або *базисних функцій* $f_i(\boldsymbol{x})$

$$y(\boldsymbol{x}) = \sum_{i=1}^{N} a_i f_i(\boldsymbol{x}) = \boldsymbol{a}^T \boldsymbol{f}(\boldsymbol{x}), \tag{15.1}$$

де $\boldsymbol{a}_i(t) = (a_1, a_2, ..., a_N)^T$ — вектор параметрів, які підлягають визначенню; $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_N(\boldsymbol{x}))^T$ — вектор базисних функцій.

У РБМ як базисні вибираються деякі функції відстані між векторами

$$f_i(\boldsymbol{x}) = f(\| \boldsymbol{x} - \boldsymbol{c}_i \|). \tag{15.2}$$

Вектори $\boldsymbol{c}_i$ називають *центрами базисних функцій*. Функції $f_i(\boldsymbol{x})$ вибираються невід'ємними й зростаючими при зменшенні $\| \boldsymbol{x} - \boldsymbol{c}_i \|$. Як міра близькості векторів $\boldsymbol{x}$ й $\boldsymbol{c}_i$ вибираються звичайно або *евклідова метрика* $\| \boldsymbol{x} - \boldsymbol{c}_i \| = \left(\sum_{j=1}^{N} (x_j - c_{ij})^2 \right)^{\frac{1}{2}}$, або *манхетенська* $\| \boldsymbol{x} - \boldsymbol{c}_i \| = \sum_{j=1}^{N} | x_j - c_{ij} |$, де $| x_j - c_{ij} | = (x_j - c_{ij}) \operatorname{sgn}(x_j - c_{ij})$,

$$\operatorname{sgn}(x_j - c_{ij}) = \begin{cases} 1, \text{ якщо } (x_j - c_{ij}) > 0; \\ 0, \text{ якщо } (x_j - c_{ij}) = 0; \\ -1, \text{ якщо } (x_j - c_{ij}) < 0. \end{cases} \tag{15.3}$$

Радіально-базисні мережі мають багато спільного зі стохастичними мережами (*див.* розділ 14). Як і стохастичні мережі, РБМ мають високу швидкість навчання. Слід також зазначити, що при їхньому навчанні не виникає проблем з «застряганням» у локальних мінімумах. Однак у зв'язку з тим, що при виконанні безпосередньо класифікації здійснюються досить складні обчислення, зростає час отримання результату.

15.1. Архітектура РБМ

Особливістю цих мереж є наявність *радіально-симетричного шаблонного шару*.

15.1.1. Структура мережі

Структура РБМ відповідає мережі прямого поширення першого порядку (рис. 15.1).

Інформація про образи передається із вхідного шару на прихований, що є шаблонним і містить p нейронів. Кожен нейрон шаблонного шару, отримуючи повну інформацію про вхідні сигнали $\boldsymbol{x}$, обчислює функцію

$$f_i(\boldsymbol{x}) = f((\boldsymbol{x}-\boldsymbol{c}_i)^T R^{-1}(\boldsymbol{x}-\boldsymbol{c}_i)),\ i=\overline{1,p}, \qquad (15.4)$$

де $\boldsymbol{x}$ — вектор вхідних сигналів $(N\times 1)$; $\boldsymbol{c}_i$ — вектор центрів $(N\times 1)$; R — вагова матриця.

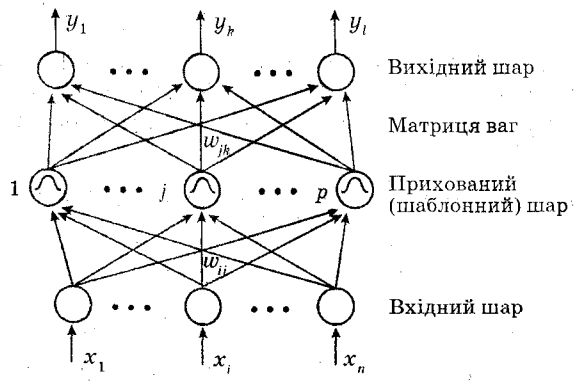


Рис. 15.1. Структура радіально-базисної мережі

Як ми вже зазначали, особливістю даних мереж є наявність радіально-симетричного шаблонного шару, у якому аналізується відстань $(\boldsymbol{x}-\boldsymbol{c}_i)^T R^{-1}(\boldsymbol{x}-\boldsymbol{c}_i)$ між вхідним вектором і центром, поданим у вигляді вектора у вхідному просторі. Вектор центрів визначається за навчальною вибіркою й зберігається в просторі ваг від вхідного шару до шару шаблонів.

15.1.2. Нейрон шаблонного шару мережі

На рис. 15.2 зображено i-й нейрон шаблонного шару РБ-мережі. Обробку інформації, що надходить на нього, умовно можна поділити на два етапи: на першому обчислюється відстань між поданим образом $\boldsymbol{x}$ і вектором центрів $\boldsymbol{c}_i$ з урахуванням обраної метрики й норми матриці R, на другому ця відстань перетворюється нелінійною активаційною функцією $f(\boldsymbol{x})$. Подвійні стрілки на рисунку позначають векторні сигнали, а потрійні — матричний сигнал.

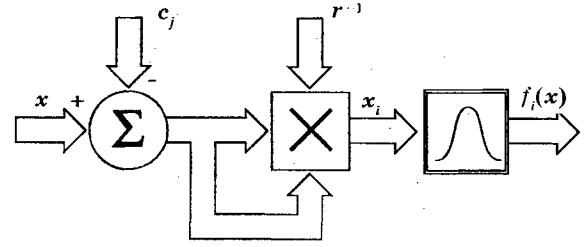


Рис. 15.2. Нейрон шаблонного шару РБМ

Як функція перетворення $f(\cdot)$ найчастіше вибираються такі:

— гауссова функція (*див.* рис. 15.3)

$$f(x) = \exp\left\{-\frac{(x-c)^2}{2\sigma^2}\right\}; \qquad (15.5)$$

— мультиквадратична функція (*див.* рис. 15.4)

$$f(x) = \left[\frac{(x-c)^2}{\sigma^2} + a^2\right]^{\frac{1}{2}}; \qquad (15.6)$$

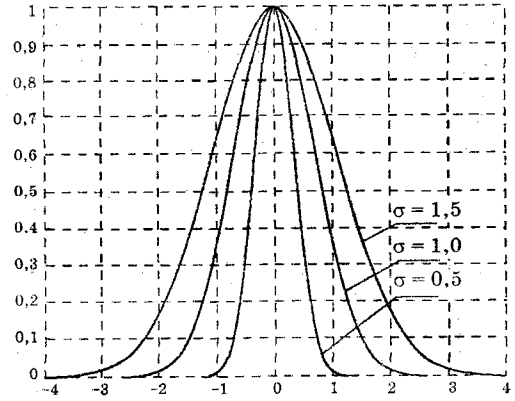


Рис. 15.3. Гауссова функція

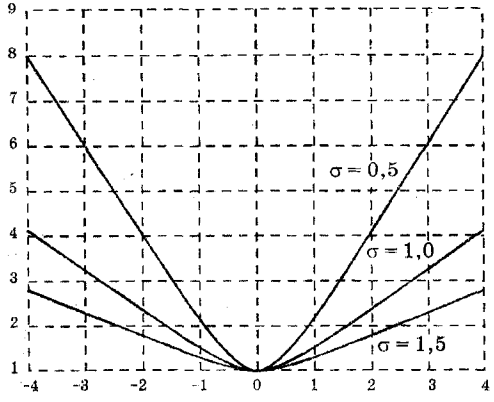


Рис. 15.4. Мультиквадратична функція

— зворотна мультиквадратична функція (рис. 15.5)

$$f(x)=\left[\frac{(x-c)^2}{\sigma^2}+a^2\right]^{\frac{1}{2}}; \qquad (15.7)$$

— сплайн-функція (рис. 15.6)

$$f(x)=x^2\log(x); \qquad (15.8)$$

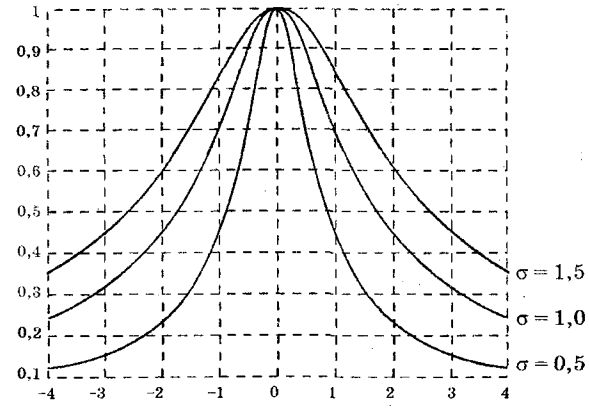


Рис. 15.5. Зворотна мультиквадратична функція ($a = 1$, $c = 0$)

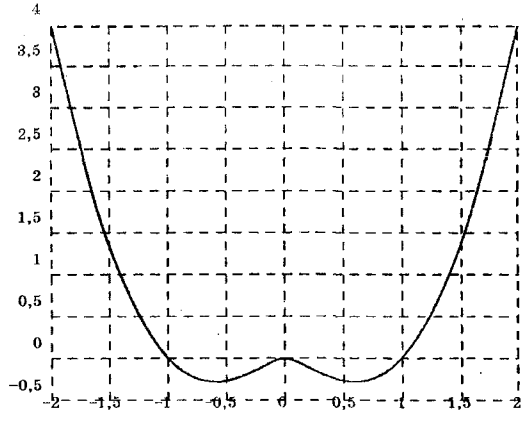


Рис. 15.6. Сплайн-функція

— функція Коші (рис. 15.7)

$$f(x)=(1+|x|)^{-1}. \qquad (15.9)$$

Слід зазначити, що ці функції допускають різні модифікації. Приклади таких модифікацій для гауссової функції $f(x)=\exp\{-P(x)\}$ наведено в табл. 15.1.

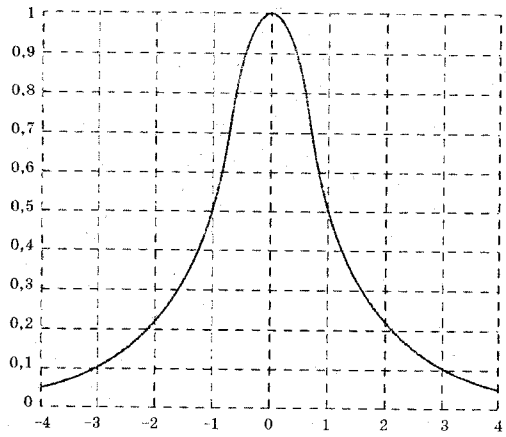


Рис. 15.7. Функція Коші

Таблиця 15.1

№	$P_k(x)$
1	$\lambda_k^2 \|\boldsymbol{x}-\boldsymbol{c}_k\|^2 = \lambda_k^2\left[(x_1-c_{k1})^2+(x_2-c_{k2})^2\right]$
2	$\lambda_k^2 \|D(\boldsymbol{x}-\boldsymbol{c}_k)\|^2 = \lambda_k^2\left[d_1^2(x_1-c_{k1})^2+d_1^2(x_2-c_{k2})^2\right]$
3	$\lambda_k^2 \|W^T(\boldsymbol{x}-\boldsymbol{c}_k)\|^2 = \lambda_k^2\left[(w_{11}(x_1-c_{k1})^2+w_{21}(x_2-c_{k2}))^2\times\right.$ $\left.\times(w_{12}(x_1-c_{k1})^2+w_{22}(x_2-c_{k2}))^2\right]$
4	$\|D_k(\boldsymbol{x}-\boldsymbol{c}_k)\|^2 = d_{k1}^2(x_1-c_{k1})^2+d_{k2}^2(x_2-c_{k2})^2$
5	$\|D_k W^T(\boldsymbol{x}-\boldsymbol{c}_k)\|^2 = \left[d_{k1}(w_{11}(x_1-c_{k1})^2+w_{21}(x_2-c_{k2}))\right]^2+$ $+\left[d_{k2}(w_{12}(x_1-c_{k1})^2+w_{22}(x_2-c_{k2}))\right]^2$
6	$\|D_k(\boldsymbol{x}-\boldsymbol{c}_k)\|^2 = d_{k1}^2(x_1-c_{k1})^2+d_{k2}^2(x_2-c_{k2})^2$, де $d_{ki}=\begin{cases}d_{ki}^+, & x_i \ge c_{ki};\\ d_{ki}^-, & x_i < c_{ki}.\end{cases}$

Норма матриці R^{-1} визначає розміщення осей у просторі. У загальному вигляді матриця R^{-1} може бути подана у такий спосіб:

$$R^{-1}=\begin{bmatrix} r_{11} & r_{12} & \dots & r_{1p} \\ r_{21} & r_{22} & \dots & r_{2p} \\ \dots & \dots & \dots & \dots \\ r_{p1} & r_{p2} & \dots & r_{pp} \end{bmatrix}. \qquad (15.10)$$

Вагову матрицю R_1 також називають *зворотною коваріаційною матрицею*. Елементи цієї матриці дорівнюють

$$r_{ij}=\sigma_{ij}^{-2}, \quad i,j=\overline{1,p}. \qquad (15.11)$$

Тут $\sigma_{ij}^{-2}=\sigma_{ji}^{-2}$ — деякі керовані параметри.

Нерідко матриця R^{-1} вибирається діагональною, тобто $r_{ij}=0$ для $i\neq j$, і більш того, приймають $r_{ii}=\sigma_{ii}^{-2}=\sigma^{-2}=\text{const}$. У цьому випадку для гауссової функції $f(x)$ (15.5) σ є стандартне відхилення.

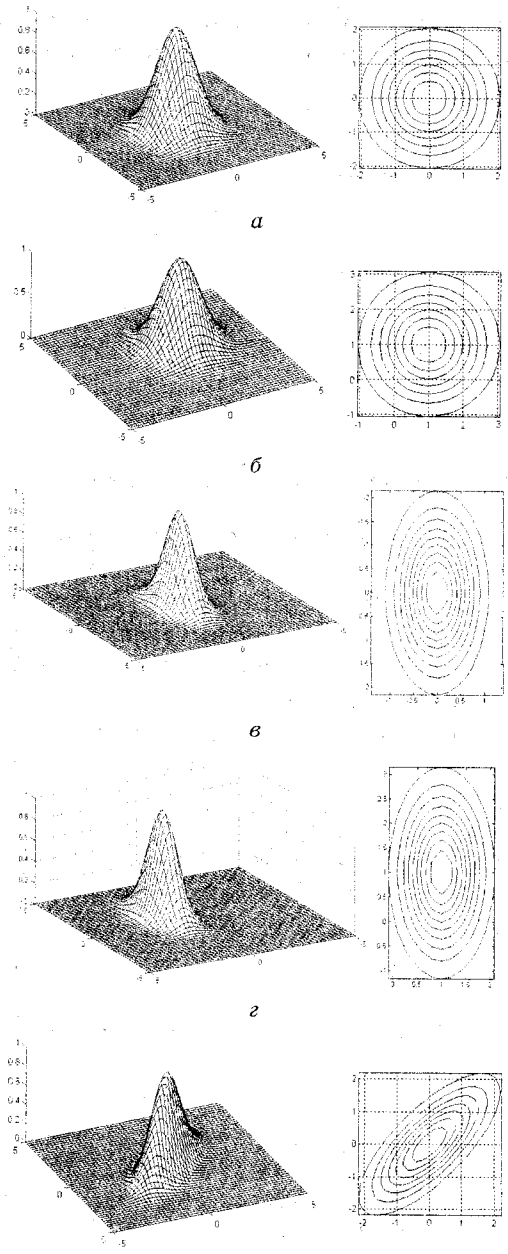
Параметр σ_i визначається експериментально, хоча мережа не є настільки критичною до його вибору. Залежність базисних функцій від σ показано на рис. 15.3–15.5.

На рис. 15.8 показано вплив вибору величин $\boldsymbol{c}$ й σ_{ij} на вид базисної функції i-го нейрона прихованого шару для двовходової РБ-мережі.

Рис. 15.8, *а*, *б* відбивають випадок $R^{-1}=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$ із $\boldsymbol{c}=(0,0)^T$ і $\boldsymbol{c}=(1,1)^T$ відповідно; рис. 15.8, *в*, *г* — випадок $R^{-1}=\begin{pmatrix}1 & 0\\ 0 & 4\end{pmatrix}$ і $\boldsymbol{c}=(0,0)^T$, $\boldsymbol{c}=(1,1)^T$; рис. 15.8, *д* — випадок $R^{-1}=\begin{pmatrix}2{,}0 & -1{,}5\\ 1{,}5 & 2{,}0\end{pmatrix}$ і $\boldsymbol{c}=(0,0)^T$.

Величина сигналу j-го нейрона вихідного шару y_j залежить від того, наскільки близький пропонований вхідний сигнал x запам'ятовуваному цим нейроном центру $\boldsymbol{c}_j$. Значення y_j визначається як зважена сума функцій (15.1), тобто

$$y_j=\sum_{i=1}^{P} f_i(x)w_{ij}. \qquad (15.12)$$



а *б* *в* *г* *д*

Рис. 15.8. Вплив вибору величин c й σ_{ij} на вигляд базисної функції

Звичайно вихідними сигналами мережі є нормалізовані значення $\hat{y}_i$, обчислені за формулою

$$\hat{y}_j = \frac{y_j}{\sum_{i=1}^{P} f_i(x)}. \qquad (15.13)$$

Приклад 15.1. На рис. 15.9 наведено приклад апроксимації функції $\varphi(x)$ (показана пунктирною лінією) трьома базисними експонентними функціями (показано суцільними лініями)

$$\varphi(x) = \sum_{i=1}^{3} w_i e^{-(x-c_i)^2} = \boldsymbol{w}^T f(x),$$

де $\boldsymbol{w} = (0{,}5,\ -0{,}3,\ 0{,}4)^T$; $c_1 = 1$; $c_2 = 3$; $c_3 = 4$.

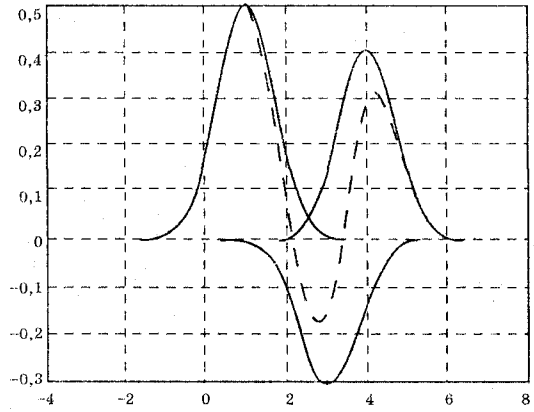


Рис. 15.9. Апроксимація функції $\varphi(x)$

Таким чином, ця мережа є нейромережевою реалізацією конкретної апроксимовної функції, при якій кожному поданому образу відповідає свій нейрон шаблонного шару. Визначення вагових параметрів нейронів вихідного шару здійснюється шляхом розв'язання системи лінійних алгебраїчних рівнянь, аналогічних (4.25)

$$\boldsymbol{y}^* = F\boldsymbol{w}, \qquad (15.14)$$

де $\boldsymbol{y}^* = (y_1^*, y_2^*, \ldots, y_N^*)^T$ — вектор заданих значень апроксимовної функції в опорних точках $c_1, c_2, \ldots, c_N$; $f_{ij} = f\left(\left\|\boldsymbol{x}_i - \boldsymbol{c}_j\right\|\right)$ — базисні функції; $\boldsymbol{w} = (w_1, w_2, \ldots, w_N)^T$ — вектор вагових параметрів мережі.

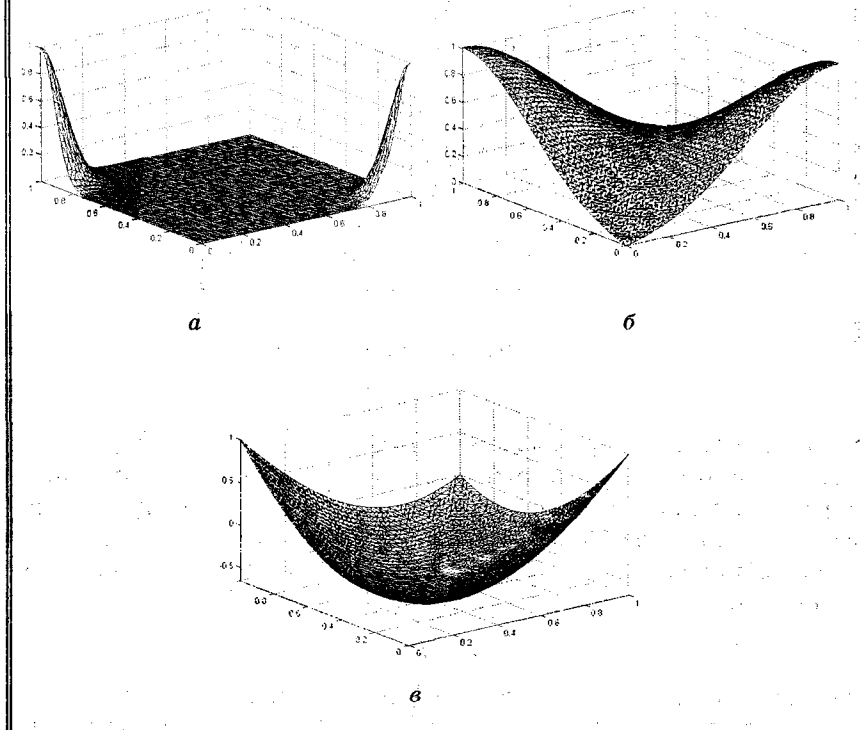
Як ми вже зазначали, розв'язання рівняння (15.14) має вигляд

$$\boldsymbol{w} = F^{+}\boldsymbol{y}^{*}, \tag{15.15}$$

яке у випадку квадратичної ($N \times N$) матриці F приймає відому форму оцінки МНК.

$$\boldsymbol{w} = \left[F^{T}F\right]^{-1}F^{T}\boldsymbol{y}^{*}. \tag{15.16}$$

Приклад 15.2. Реалізація функції «що виключає АБО» на РБМ із випадково заданими значеннями $\boldsymbol{w}$ наведена на рис. 15.10, причому рис. 15.10, *а* відповідає випадку $\sigma = 0{,}1$; рис. 15.10, *б* — $\sigma = 0{,}5$; рис. 15.10, *в* — $\sigma = 1{,}0$.



а *б* *в*

Рис. 15.10. Реалізація функції «що виключає АБО» РБ-мережею

15.2. Навчання радіально-базисної мережі

Як випливає з вищевикладеного, РБ-мережу характеризують три типи параметрів:

— *лінійні вагові параметри* вихідного шару w_{ij} (входять в опис мережі лінійно);

— *центри* $\boldsymbol{c}_i$ — нелінійні (входять в опис нелінійно) параметри прихованого шару;

— *відхилення* (*радіуси базисних функцій*) σ_{ij} — нелінійні параметри прихованого шару.

Навчання мережі, що полягає у визначенні цих параметрів, може зводитися до одного з варіантів:

1. Задаються центри й відхилення, а обчислюються тільки ваги вихідного шару.

2. Визначаються шляхом самонавчання центри й відхилення, а для корекції ваг вихідного шару використовується навчання із учителем.

3. Визначаються всі параметри мережі за допомогою навчання із учителем.

Перші два варіанти застосовуються в мережах, що використовують базисні функції з жорстко заданим радіусом (відхиленням). Третій варіант, будучи найбільш складним і трудомістким у реалізації, припускає використання будь-яких базисних функцій.

Отже, навчання мережі полягає в такому:

— визначаються центри $\boldsymbol{c}_i$;

— вибираються параметри σ_j;

— обчислюються елементи матриці ваг W.

15.2.1. Вибір параметрів центрів і відхилень σ

Центри $\boldsymbol{c}_i$ визначають точки, через які має проходити апроксимовна функція. Оскільки велика навчальна вибірка призводить до затягування процесу навчання, у РБ-мережах широко використовується кластеризація образів, при якій подібні вектори поєднуються в *кластери*, що подаються потім у процесі навчання тільки одним вектором. Сьогодні існує досить велика кількість ефективних алгоритмів кластеризації.

Використання кластеризації відображується на формулах (15.12), (15.13) у такий спосіб:

$$\hat{y}_i = \frac{\sum_{i=1}^{P} m_i f_i(x) w_{ij}}{\sum_{i=1}^{P} m_i f_i(x)}, \qquad (15.17)$$

де m_i — кількість вхідних векторів в i-му кластері.

У найбільш простому варіанті алгоритм кластеризації k-середнього, направляє кожен образ у кластер, що має найближчий до даного образу центр [103]. Якщо кількість центрів заздалегідь задана або визначена, алгоритм, обробляючи на кожному такті вхідний вектор мережі, формує в просторі входів мережі центри кластерів. Зі збільшенням кількості тактів ці центри збігаються до центрів даних. Кандидатами в центри є всі виходи прихованого шару, однак у результаті роботи алгоритму буде сформована підмножина найбільш істотних виходів.

Як уже зазначалося, параметр σ_i, що входить у формули для функцій перетворення, визначає розкид щодо центра $\boldsymbol{c}_i$. Варіюючи параметри $\boldsymbol{c}_i$ й σ_i, намагаються перекрити весь простір образів, не залишаючи порожнин. Використовуючи метод k-найближчих сусідів, визначають k сусідів центру $\boldsymbol{c}_i$ й, усереднюючи, обчислюють середнє значення $\hat{\boldsymbol{c}}_i$. Величина відхилення $\hat{\boldsymbol{c}}_i$ від $\boldsymbol{c}_i$ служить підставою для вибору параметра σ_i. На практиці часто виправдовує себе вибір

$$\sigma = \frac{d}{\sqrt{2p}},$$

де $d = \max(c_i - c_k)$ — максимальна відстань між обраними центрами; p — кількість нейронів шаблонного шару (образів).

Самі ж вагові коефіцієнти нейронів вихідного шару визначаються зі співвідношень (15.15) або (15.16).

Якщо якість апроксимації є незадовільною, вибір параметрів $\boldsymbol{c}_i$ й σ, а також визначення ваг $\boldsymbol{w}$ повторюють доти, поки отримане розв'язання не виявиться задовільним.

15.2.2. Самонавчання параметрів центрів

Настроювання параметрів центрів може відбуватися з допомогою методів, аналогічних методам *LVQ* (*див.* розділ 10), або за допомогою *мап Когонена* (розділ 11). При цьому найчастіше всі образи, що навчають, попередньо нормалізують, що дозволяє використати скалярний добуток вхідного вектора (образу) і вектора ваг як їхню

міру відповідності. Компонентам вагового вектора присвоюються випадкові початкові значення. Корекція компонентів вектора центрів $\boldsymbol{c}_i(k)$, $i = \overline{1,p}$, розташованого найближче до поданого образу $\boldsymbol{x}(k)$, що навчає, відбувається за алгоритмом

$$\boldsymbol{c}_i(k+1) = \boldsymbol{c}_i(k) + \alpha(k)\delta_{ij}[\boldsymbol{x}(k) - \boldsymbol{c}_i(k)], \quad i = \overline{1,p}, \qquad (15.18)$$

де $i = \arg\min_j \{d_j(k)\}$;

$$d_j(k) = \| \boldsymbol{x}(k) - \boldsymbol{c}_j(k) \|.$$

Коефіцієнт посилення $\alpha(k) \in (0, 1)$, що змінюється в часі, задається співвідношенням

$$\alpha(k) = \frac{\alpha(k-1)}{\sqrt{1 + \mathrm{int}\left(\frac{k}{N+p+1}\right)}},$$

де $\mathrm{int}(x)$ — ціла частина x.

Як свідчать експерименти, алгоритм (15.18) не критичний до вибору початкового значення $\alpha(0)$, і позитивні результати забезпечуються вибором $\alpha(0) \approx 1$.

Більш простим правилом завдання зміни $\alpha(k)$ є таке:

$$\alpha(k) = k^{-1}.$$

Обчислені значення центрів $\boldsymbol{c}_i(k)$, $i = \overline{1,p}$ використовуються потім для визначення функції перетворення f_i й обчислення виходів мережі (15.13).

У процесі використання самоорганізувальних мап Когонена змінюються параметри центру не тільки нейрона-переможця шаблонного шару, але і його сусідів, що перебувають у цьому ж шарі. Використання як методів LVQ, так і мап Когонена має за мету таке розміщення в просторі векторів ваг, щоб їхня щільність розподілу відповідала щільності розподілу образів, що навчають.

15.2.3. Навчання мережі із учителем

Як і в випадку будь-якого навчання із учителем, мережі послідовно подаються навчальні пари $(\boldsymbol{x}_i, \boldsymbol{y}_i^*)$ $i = \overline{1,k}$, де $\boldsymbol{x}_i$ й $\boldsymbol{y}_i^*$ — вхідний і бажаний вихідний вектори i-ї навчальної пари відповідно. На виході мережі з'являється вектор $\boldsymbol{y}_i = (y_{1i}, y_{2i}, ..., y_{Li})^T$. Цей

вектор порівнюється з необхідним y_i^* й обчислюється неузгодженість $(y_i - y_i^*)$ використовувана в заздалегідь обраному опуклому функціоналі якості, мінімізація якого дає необхідні значення елементів вагової матриці.

Звичайно за функціонал, що мінімізується, приймається квадратична форма

$$I = \sum_{i=1}^{k}(y_i^* - y_i)^T(y_i^* - y_i) = \sum_{i=k}^{k}(y_i^* - Wf(x_i))^T(y_i^* - Wf(x_i)). \quad (15.19)$$

Так, усі пари $(x_i,\ y_i^*)$ $i=\overline{1,k}$ утворюють матриці вхідних сигналів $X(k)$ розмірності $N \times k$ і матриці необхідних вихідних сигналів $Y^*(k)$ розмірності $L \times k$, РБ-мережа описується таким матричним рівнянням:

$$Y(k) = WF(k), \quad (15.20)$$

де $Y(k)$ — $(L \times k)$-матриця вихідних сигналів; W — $(L \times p)$-матриця ваг; $F(k)$ — $(p \times k)$-матриця використовуваних у мережі функцій перетворення, а критерій (15.19) може бути записаний так:

$$I = tr[(Y(k) - WF(k))^T(Y(k) - WF(k))]. \quad (15.21)$$

Тут $tr[\cdot]$ — слід матриці $[\cdot]$.

Зазначимо, що (15.21) є узагальненням (15.14) на випадок k образів. Мінімізація (15.21), здійснювана шляхом розв'язання системи лінійних алгебраїчних рівнянь

$$\frac{\partial I}{\partial w_{ij}} = 0, \quad (15.22)$$

призводить після подання k навчальних пар до отримання оцінки МНК, аналогічній (15.15)

$$W = F^+(k)Y(k) \quad (15.23)$$

або

$$W(k) = Y(k)F^T(k)[F(k)F^T(k)]^{-1}. \quad (15.24)$$

При поданні нової навчальної пари $\{x_{k+1}, y_{k+1}^*\}$ оцінка матриці вагових коефіцієнтів за аналогією (15.24) записується у вигляді

$$W(k+1) = Y(k+1)F^T(k+1)[F(k+1)F^T(k+1)]^{-1}. \quad (15.25)$$

Оскільки розмірності матриць $Y(k+1)$ і $F(k+1)$ збільшуються, ускладнюється процедура обчислення зворотної матриці й зростає загальна кількість арифметичних операцій, необхідних для обчислення оцінки (15.25).

Істотного спрощення можна досягти, якщо врахувати, що використовувані в (15.24) матриці формуються у такий спосіб:

$$Y(k+1) = [Y(k) \vdots y_{k+1}],\ F(k+1) = [F(k) \vdots f_{k+1}], \quad (15.26)$$

тобто є блоковими.

Припустимо, що матриця $[F(k+1)\ F^T(k+1)]$ не вироджена. Позначимо

$$P^{-1}(k) = F(k)F^T(k) = \sum_{i=1}^{k} f_i f_i^T. \quad (15.27)$$

Тоді з урахуванням (15.26), (15.27)

$$P^{-1}(k+1) = F(k+1)F^T(k+1) = \sum_{i=1}^{k+1} f_i f_i^T = \\ = \sum_{i=1}^{k} f_i f_i^T + f_{k+1}f_{k+1}^T = P^{-1}(k) + f_{k+1}f_{k+1}^T. \quad (15.28)$$

Застосування до (15.28) леми про обернення матриць [70] дає

$$P(k+1) = P(k) - P(k)f_{k+1}(1 + f_{k+1}^T P(k) f_{k+1})^{-1} f_{k+1}^T P(k). \quad (15.29)$$

Підстановка в (15.25) співвідношень (15.26) і (15.28) та нескладні перетворення призводять до такої формули:

$$W(k+1) = W(k) + [y(k+1) - W(k)f_{k+1}]f_{k+1}^T P(k). \quad (15.30)$$

Отже, співвідношення (15.29), (15.30) описують рекурентну процедуру обчислення оцінки матриці вагових коефіцієнтів при додаванні нової навчальної пари $\{x_{k+1}, y_{k+1}^*\}$.

Важливим питанням, що виникає під час реалізації подібних алгоритмів, є вибір початкових умов. Оскільки передбачається, що ранг матриці $F(k)F^T(k)$ дорівнює p, то необхідно, щоб навчання відбувалося з використанням числа навчальних пар, не меншого p. Не вдаючись у подробиці, зазначимо тільки, що звичайно як початкове значення матриці P_0 приймається $P_0 = \alpha I$,

де $\alpha >> 1$. У цьому випадку рекурентна оцінка (15.29), (15.30) в асимптотиці збігається з нерекурентною оцінкою (15.25).

Однак у ряді робіт показано, що *online*-навчання мережі шляхом рекурентного настроювання її параметрів є нестійким. Більш стійким є навчання в режимі *offline*, коли для знаходження параметрів мережі використовується критерій (15.21). Детальніше навчання мережі в режимі *offline* буде розглянуто нижче.

15.2.4. Зміна кількості нейронів шаблонного шару

Безсумнівно, важливим є питання вибору кількості нейронів прихованого шаблонного шару. Принципово може бути реалізована РБ-мережа із варіювальною кількістю цих нейронів, що збільшується, наприклад, у випадку, якщо досягнута після навчання помилка апроксимації перевищуватиме припустиму. Такий підхід може бути застосований також у випадку, якщо не проведене початкове розбивання образів на кластери.

Розглянемо процедуру введення додаткового нейрона прихованого шару.

Позначимо матрицю ваг мережі W, що описується рівнянням (15.20) і містить p нейронів прихованого шару, як W_p. При введенні нового $(p+1)$-го нейрона модель (15.12) набуває вигляду

$$y_j = \sum_{i=1}^{P+1} f_i(x) w_{ij} \qquad (15.31)$$

або в матричному вигляді

$$Y(k) = W_{P+1} F_{P+1}(k), \qquad (15.32)$$

де W_{p+1} — $(L \times (p+1))$-матриця ваг; $F_{p+1}(k)$ — $((p+1) \times k)$-матриця використовуваних у мережі $(p+1)$-ї функцій перетворення.

Розмірність вектора виходів мережі не змінюється.

За аналогією з (15.24) МНК-оцінка записується у вигляді

$$W_{P+1}(k) = Y(k) F_{P+1}^T(k) \left[F_{P+1}(k) F_{P+1}^T(k) \right]^{-1}. \qquad (15.33)$$

І в цьому випадку оцінка може обчислюватися рекурентно. Формула для корекції елементів вагової матриці при введенні додаткового $(p+1)$-го нейрона прихованого шару має вигляд

$$W_{P+1} = (W_P : 0) - (W_P f_{P+1} b_{p+1}^T : 0) + Y(f_{P+1} b_{P+1}^T : F_P^T a_{P+1} + f_{P+1} d_{P+1}). \qquad (15.34)$$

У ряді випадків при одержанні оцінки елементів матриці вагових коефіцієнтів виникає необхідність вилучити з будь-якої причини із усієї множини навчальних пар якусь певну пару (наприклад, навчальна пара, що містила помилкову інформацію). При цьому, природно, оцінка також має бути скорегована. І в цьому випадку доцільно скористатися рекурентними обчисленнями. Нехай для простоти необхідно вилучити останню $(k+1)$-ту навчальну пару. Тоді з (15.28) маємо співвідношення

$$P^{-1}(k) = P^{-1}(k+1) - f_{k+1} f_{k+1}^T,$$

застосування до якого леми про обернення матриці дає

$$P(k) = P(k+1) + P(k+1) f_{k+1} (1 - f_{k+1}^T P(k+1) f_{k+1})^{-1} f_{k+1}^T P(k+1).$$

Аналогічно для оцінки матриці отримуємо

$$W(k) = W(k+1) - [y(k+1) - W(k+1) f_{k+1}] f_{k+1}^T P(k).$$

На завершення необхідно зазначити таке: як і багатошаровий персептрон, РБ-мережа є універсальним апроксиматором.

Приклад 15.3. На рис. 15.11 наведено результати апроксимації РБ-мережею функції, заданої у прикладі 3.11

$$f(x,y) = 0{,}5 \exp\left[-\frac{(9x-2)^2 + (9y-2)^2}{4} \right] +$$

$$+ 0{,}75 \exp\left[-\frac{(9x+1)^2}{49} - \frac{(9y+1)^2}{10} \right] +$$

$$+ 0{,}5 \exp\left[-\frac{(9x-7)^2 + (9y-3)^2}{4} \right] -$$

$$- 0{,}2 \exp\left[-(9x-4)^2 - (9y-7)^2 \right].$$

Істотного скорочення часу навчання вдається досягти, якщо розбити мережу на ряд мереж, кожна з яких навчається окремо. На рис. 15.12 наведено результати моделювання роботи

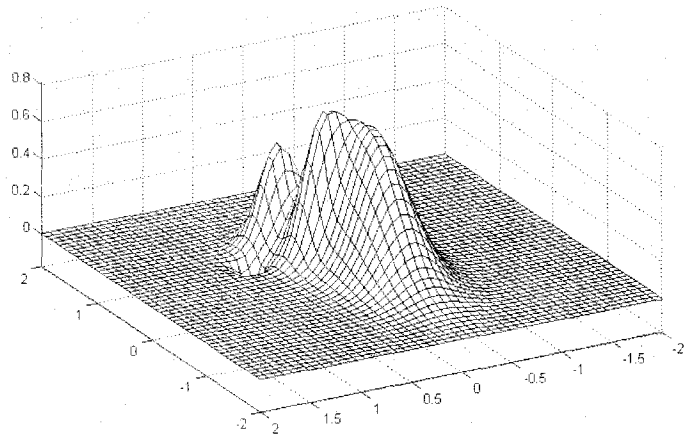
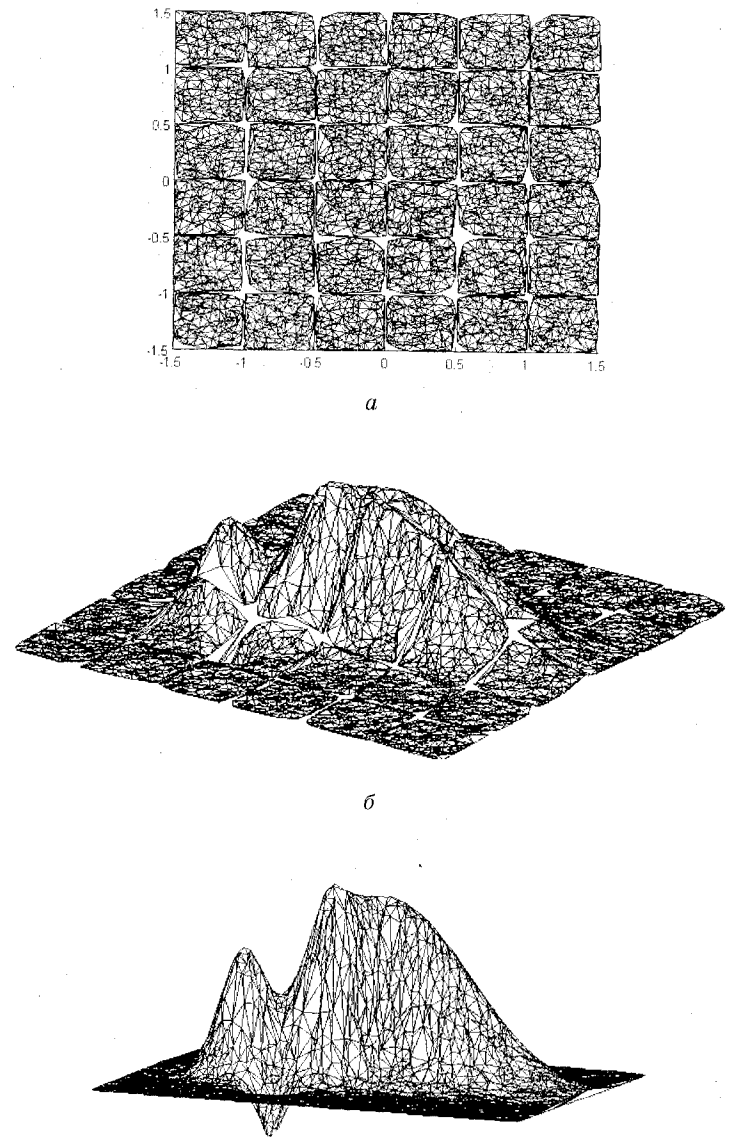


Рис. 15.11. Апроксимація функції $f(x,y)$ РБ-мережею

мережі, що складається з 36 мереж, для навчання кожної з яких використовувалося 150 навчальних пар (рис. 15.12, *а*). Для заданої похибки апроксимації (10^{-4}) загальна кількість нейронів мережі дорівнює 751, а час навчання скоротилося приблизно в 10 разів. Результат апроксимації наведений на рис. 15.12,*б*.

Результати моделювання свідчать про те, що точність апроксимації за допомогою РБ-мережі, вимагаючи менших обчислювальних витрат, після одноразового навчання за всіма спостереженнями не гірше тієї, яка забезпечується багатошаровим персептроном.

У ряді випадків більш ефективним виявляється персептрон, а в ряді випадків — РБ-мережа. Приклади таких випадків наведено на рис. 15.13. Рис. 15.13, *а* відповідає випадку, коли розпізнані образи легко можуть бути розділені на два класи за допомогою персептрона, у той час як для такого ж розбивання на класи потрібна була б складна РБ-мережа (рис. 15.13, *б*). Водночас для побудови кластера, що має форму еліпса, потрібно було б багато персептронів (рис. 15.13, *в*), а РБ-мережа містила б тільки один нейрон шаблонного шару (рис. 15.13, *г*).



а

б

в

Рис. 15.12. Апроксимація функції $f(x, y)$ РБ-мережею, поданою 36 мережами

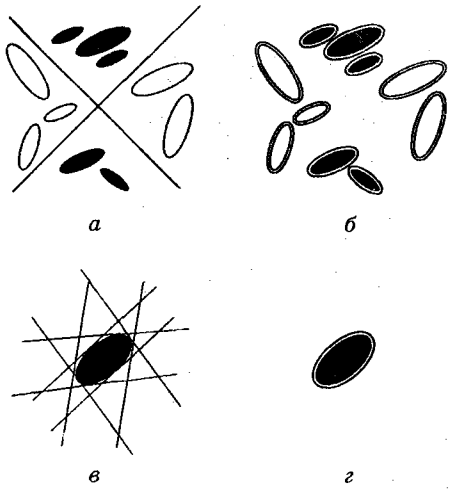


Рис. 15.13. Ілюстрація класифікації образів багатошаровим персептроном і РБ-мережею

15.3. Нормалізована радіально-базисна мережа

У даній мережі використовуються *нормалізовані базисні функції* (НБФ), що обчислюють у такий спосіб:

$$\bar{f}_i(x) = \frac{f_i(x)}{\sum_{j=1}^{M} f_j(x)}, \qquad (15.35)$$

де M — кількість нейронів у мережі.

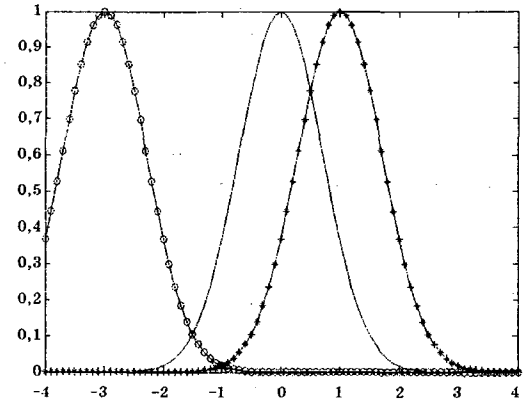
Властивість даної мережі полягає в тому, що для будь-якої точки вхідного сигналу сума всіх базисних функцій дорівнює одиниці, тобто

$$\sum_{i=1}^{M} \bar{f}_i(x) = 1, \qquad (15.36)$$

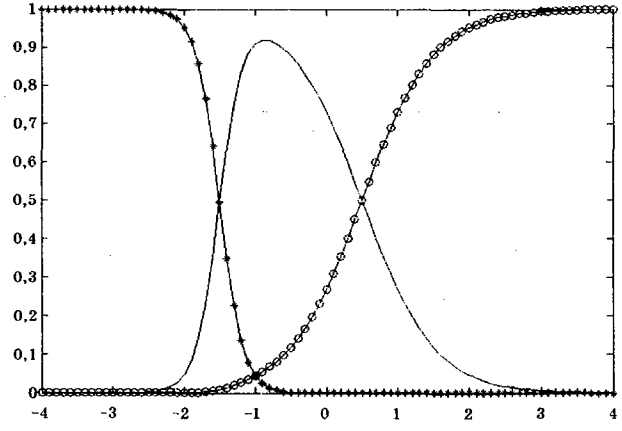
що робить їх привабливими для багатьох застосувань. Як і РБФ, нормалізовані РБФ (НРБФ) є універсальними апроксиматорами.

Слід зазначити, що нормалізація (15.35) істотно впливає на вигляд базисної функції, змінюючи тим самим властивості мережі.

На рис. 15.14 наведено ненормалізовані (рис. 15.14, *а*) і нормалізовані (рис. 15.14, *б*) гауссівські функції з параметрами $c_1 = -3$, $c_2 = 0$, $c_3 = 1$ й $\sigma_1^2 = \sigma_2^2 = \sigma_3^2 = 1$.



а



б

Рис. 15.14. Ненормалізовані та нормалізовані гауссівські функції

Вплив величини дисперсії σ_i^2 $(i = \overline{1,3})$ на вид гауссівських базисних функцій наведено на рис. 15.15.

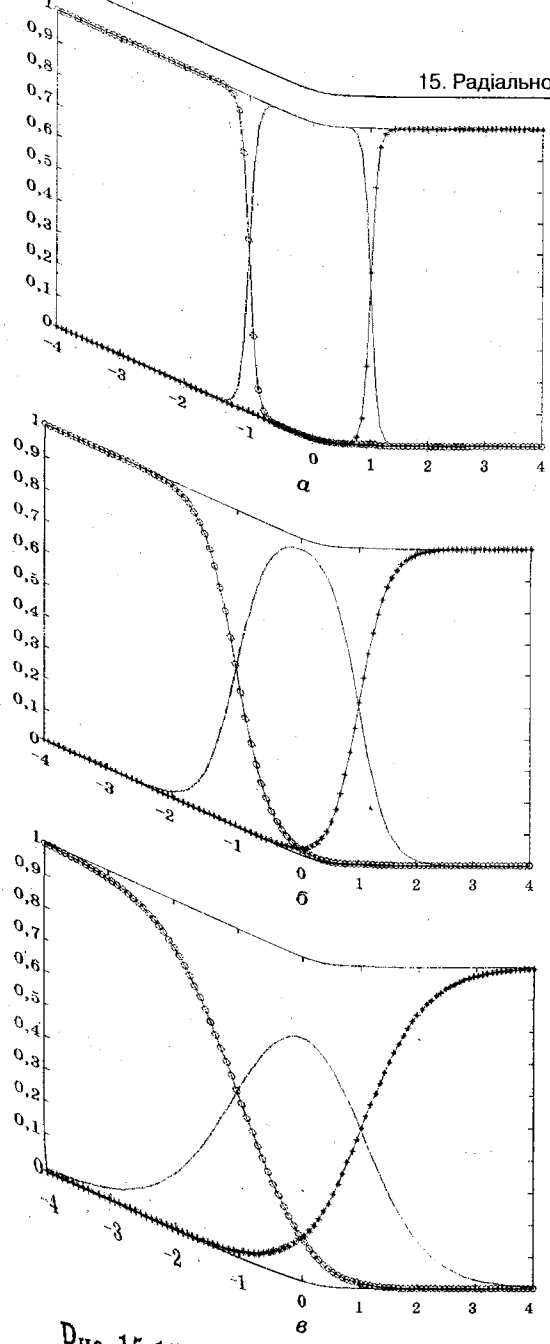


Рис. 15.15. Вплив величини σ_i^2 на вигляд нормалізованих гауссівських функцій:
а — $\sigma_i^2 = 0{,}5$ $(i = \overline{1,3})$; *б* — $\sigma_i^2 = 1{,}0$ $(i = \overline{1,3})$; *в* — $\sigma_i^2 = 1{,}5$ $(i = \overline{1,3})$

Нарешті, на рис. 15.16 наведено ненормалізовані й нормалізовані гауссівські функції із $c_1 = -1$; $\sigma_1^2 = 0{,}5$; $c_2 = 0$; $\sigma_2^2 = 1$; $c_3 = 1$; $\sigma_3^2 = 1{,}5$.

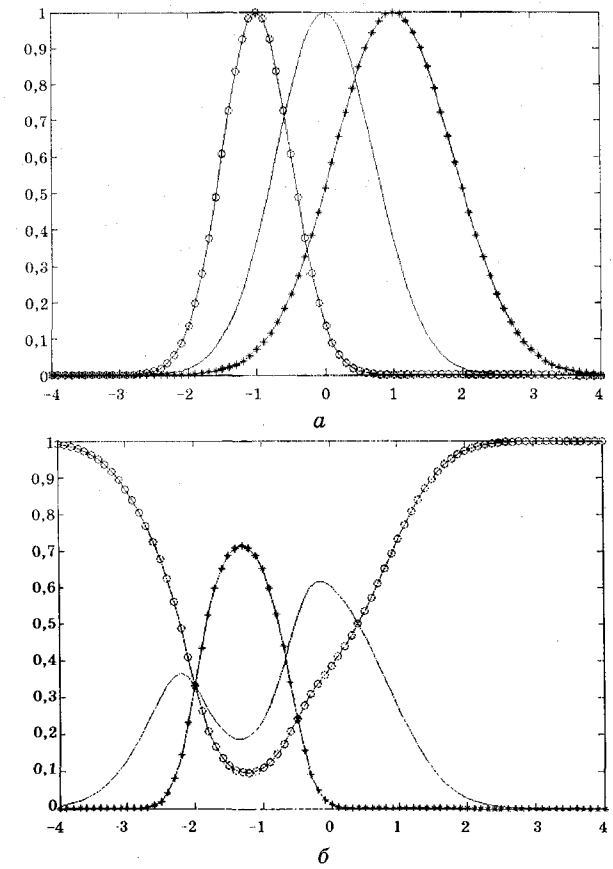


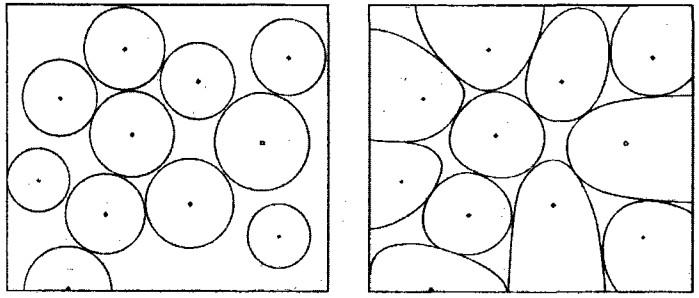
Рис. 15.16. Ненормалізовані й нормалізовані гауссівські функції

Як видно з наведених рисунків, нормалізація істотно змінює форму, причому на цю форму впливає як ширина вихідних БФ (при зменшенні σ_i^2 форма нормалізованих функцій стає більш прямокутною), так і їхнє взаємне розташування, що характеризується параметрами c_i. Як випливає з формули (15.35), кожна нормалізована БФ є функцією всіх вихідних БФ. Тому зміна виду будь-якої

вихідної БФ або введення додаткової БФ (що характерно для початкового етапу функціонування РБМ, коли визначається її структура й відбувається навчання) призводить до зміни всіх нормалізованих БФ. Це є досить незручним, якщо передбачається навчання й функціонування мережі в режимі *online*. Крім того, додаткові незручності застосування нормалізованих БФ обумовлені тим, що при нерівномірному розподілі центрів вихідних БФ або при різній їхній ширині, максимуми НБФ будуть опущені стосовно центрів, а самі НБФ можуть стати не монотонно спадними. Останнє призводить до неправильної реакції нейрона на вхідні сигнали: так, на сигнали, що знаходяться далі від центра БФ нейрона, реакція його буде сильнішою, ніж на сигнали, які розташовані ближче до центра.

При використанні НРБМ у задачах великої розмірності необхідно також враховувати, що із зростанням розмірності завдання збільшується кількість використовуваних НБФ, що призводить не тільки до зазначених вище наслідків, але й до різкого зменшення значень максимумів НБФ.

Слід зазначити, що НРБМ є досить зручними й досить ефективними, якщо центри БФ рівномірно розподілені в просторі вхідних сигналів (наприклад, шляхом апріорного завдання), а ширина БФ приблизно однакова. У цьому випадку в процесі навчання мережі ні центри, ні ширина БФ не змінюються, а корегуються лише її вагові коефіцієнти w. Застосування НБФ, на відміну від ненормалізованих БФ, забезпечує однаковою мірою покриття всіх точок вхідного простору, роблячи мережу менш чутливою до невдалого вибору центрів. Рис. 15.17 ілюструє різні можливості покриття двовимірного простору вхідних сигналів мережами РБМ і НРБМ при тих самих завданнях центрів БФ.

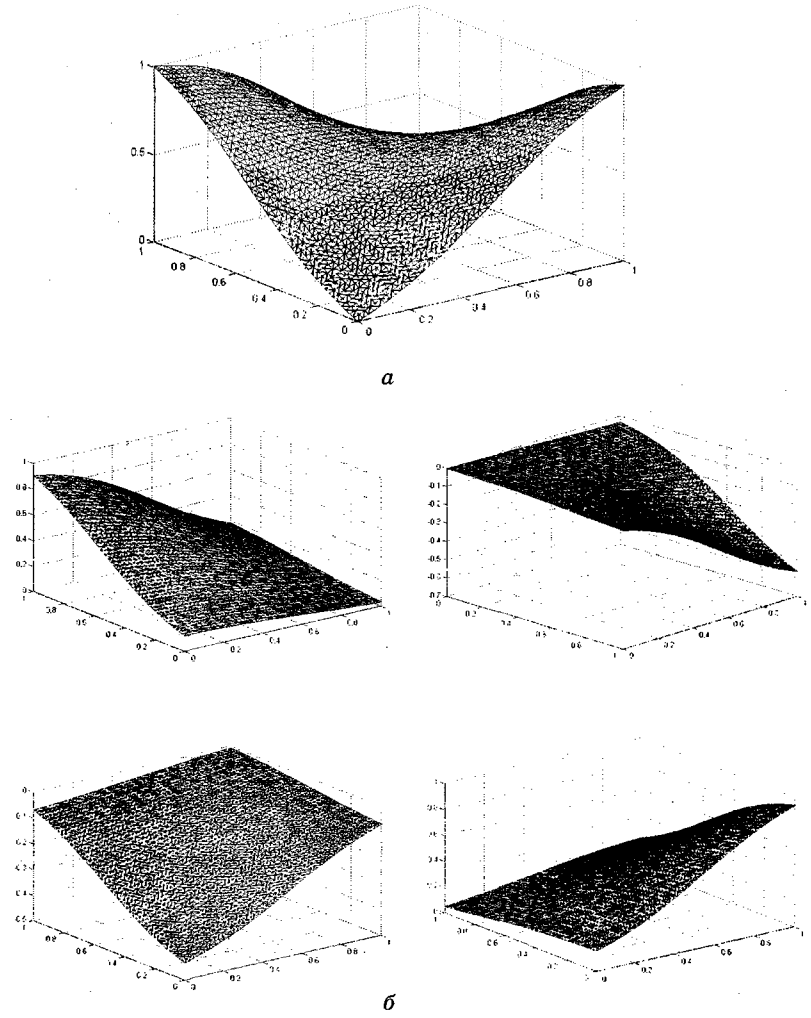


а *б*

Рис. 15.17. Покриття двовимірного простору ненормалізованими (*а*) і нормалізованими (*б*) РБФ

Приклад 15.4. Результати реалізації функції «що виключає АБО» за допомогою НРБМ наведені на рис. 15.18.

На рис. 15.18, *а* зображена сама функція, а на рис. 15.18, *б* — НБФ чотирьох нейронів, використовуваних для побудови мережі.



а

б

Рис. 15.18. Результати реалізації функції «що виключає АБО» за допомогою НРБМ

15.4. Гіпербазисна мережа

У розглянутих радіально-базисних мережах вихідні нейрони реалізовували зважену суму сигналів, що на них надходять, тобто мали фактично лінійні функції активації. Як свідчать дослідження, значно кращих результатів можна досягти, якщо активаційні функції вихідних нейронів будуть сигмоїдальними. Отже, заміною функцій активації вихідних нейронів можлива модифікація радіально-базисних мереж.

Крім того, можливе зважування як відстані від образів до відповідних центрів $\|\boldsymbol{x} - \boldsymbol{c}_i\|$ кожної базисної функції шляхом введення деякого параметра q, так й окремих його компонентів $(x_j - c_{ji})$, що означає у випадку евклідової метрики перехід від гіперсфер до гіпереліпсоїдів. Можуть бути використані також різні величини радіусів і різний напрямок осей для базисних функцій.

Нарешті, базисна функція, використовувана для апроксимації досліджуваної функції F, може бути доповнена поліномом якогось, зокрема першого, степеня й постійною величиною θ, аналогічною порогу (зміщенню).

Модифіковані у такий спосіб радіальні базисні функції називають *гіпербазисними*, а відповідну мережу — *гіпербазисною мережею*.

15.4.1. Структура мережі

У гіпербазисній мережі сигнал i-го нейрона вихідного шару обчислюється у такий спосіб:

$$y_i(\boldsymbol{x}) = \sigma(F_j(\boldsymbol{x})) =$$

$$= \sigma\left(\sum_{i=1}^{L} w_{ij} f\left(\|\boldsymbol{x} - \boldsymbol{c}\|_i, q_i\right) + \sum_{n=1}^{N} d_{nj} x_n + \theta_j\right), \quad j = \overline{1,M}, \qquad (15.37)$$

де $\sigma(\cdot)$ — сигмоїдальна активаційна функція; q_i — параметр зміни центра функции; θ_i — поріг.

Така модифікація мережі проявляється в тому, що в ній крім зміненого вихідного нейрона з'являються так звані «*short-cut*»-зв'язки (*див.* розділ 2) між нейронами вхідного й вихідного шарів, обумовлені ваговими коефіцієнтами d_{nj} $(n = \overline{1,N};\ j = \overline{1,M})$. Структуру цієї мережі наведено на рис. 15.19.

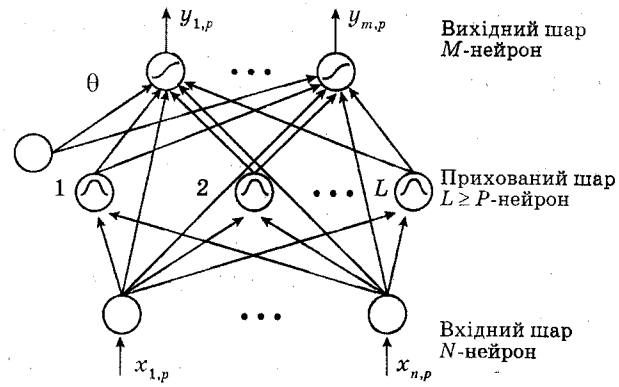


Рис. 15.19. Структура мережі гіпербазисної основи

15.4.2. Навчання мережі

У процесі навчання мережі настроюються всі параметри, що входять у (15.37). Слід зазначити, що звичайно первинний вибір центрів $\boldsymbol{c}_j$ не є оптимальним. Тому при зміні (перерахуванні) параметрів центрів мають бути перелічені й ваги w_{ij}. Для навчання мережі використовується метод зворотного поширення помилки, що полягає в мінімізації квадратичного функціонала помилки реакції мережі за всією множиною поданих образів P, тобто

$$I = \frac{1}{2}\sum_{P=1}^{P}\sum_{j=1}^{M} e_{pj}^2 = \frac{1}{2}\sum_{P=1}^{P}\sum_{j=1}^{M}(y_{pj}^* - y_{pj})^2, \qquad (15.38)$$

де y_{pj}^* — бажана реакція j-го нейрона вихідного шару мережі на p-й образ. Відповідно до цього методу (*див.* розділ 3) корекція параметрів мережі здійснюється за формулами

$$\Delta w_{ij} = \gamma_1 \sum_{P=1}^{P} e_{pj}\sigma'(F_j(\boldsymbol{x}_p)) f(\varepsilon_{pi}, q_i); \qquad (15.39)$$

$$\Delta c_i = -\gamma_2 \sum_{P=1}^{P} f_\xi(\varepsilon_{pi}, q_i)(x_p - w_i)\sum_{j=1}^{M} w_{ij} e_{pj}\sigma'(F_j(\boldsymbol{x}_p)); \qquad (15.40)$$

$$\Delta q_i = \gamma_3 \sum_{P=1}^{P} f_{q_i}(\varepsilon_{pi}, q_i)\sum_{j=1}^{M} w_{ij} e_{pj}(F_j(\boldsymbol{x}_p)); \qquad (15.41)$$

$$\Delta d_{nj} = \gamma_4 \sum_{P=1}^{P} e_{pj}\sigma'(F_j(\boldsymbol{x}_p)) x_{np}; \qquad (15.42)$$

$$\Delta\theta_j = \gamma_5 \sum_{P=1}^{P} e_{pj}\sigma'(F_j(\boldsymbol{x}_p)) \; ; \; i=\overline{1,L}; \; j=\overline{1,M}; \; n=\overline{1,N}, \qquad (15.43)$$

де $f(\varepsilon_{pi},q_i) = f(\|\boldsymbol{x}_p - \boldsymbol{w}_i\|,q_i)$

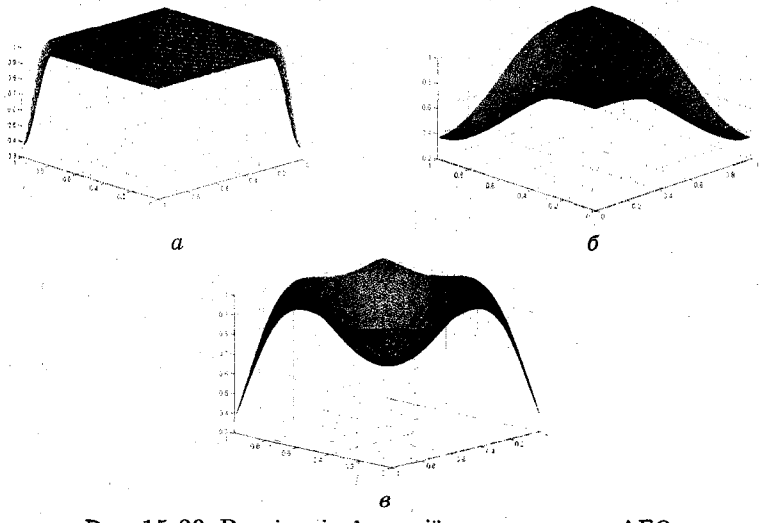
$$f(\varepsilon_{pi},q_i) = \frac{\partial f(\varepsilon_{pi},q_i)}{\partial \varepsilon_{pi}};$$

$$f_{q_i}(\varepsilon_{pi},q_i) = \frac{\partial f(\varepsilon_{pi},q_i)}{\partial q_i};$$

$\gamma_k = (k = 1, 2, \ldots, 5)$ — коефіцієнти навчання (можуть бути обрані, наприклад, однаковими).

Підсумовування за всіма поданими образами p у наведених формулах означає, що навчання мережі відбувається в режимі *offline*. Навчання в режимі *online*, коли параметри уточнюються потактово після подання кожного образу, у даній мережі, на відміну від мереж інших типів, є нестійким. З метою підвищення стійкості в алгоритмах застосовують регуляризуючі добавки, що адитивно вводять у (15.39)–(15.43) зважені зміни цих же ваг на попередньому такті.

Пример 15.5. На рис. 15.20 наведено реалізацію функції «що виключає АБО» гіпербазисною мережею. Рис. 15.20, *а* відповідає вибору $\sigma = 0{,}1$; рис. 15.20, *б* — $\sigma = 0{,}5$; рис. 15.20, *в* — $\sigma = 1$.



а *б* *в*

Рис. 15.20. Реалізація функції «що виключає АБО» гіпербазисною мережею

Контрольні запитання

1. Дайте визначення радіально-базисній мережі.
2. Що являє собою нейрон шаблонного шару РБМ?
3. Наведіть приклади базисних функцій, використовуваних у РБМ.
4. Яким чином здійснюється навчання РБМ? У чому суть самонавчання?
5. У який спосіб можна змінювати структуру РБМ? З якою метою може здійснюватися зміна структури мережі?
6. Як впливає розмірність досліджуваного завдання на складність реалізації мережі?
7. Що являє собою нормалізована РБМ?
8. Як впливає вибір параметрів c й σ^2 на вид НРБФ?
9. У чому переваги НРБФ порівняно з ненормалізованими РБФ?
10. Що являє собою мережа гіпербазисної основи? Яким чином здійснюється її навчання?
11. Порівняйте ефективність розв'язання задачі апроксимації за допомогою персептрона й РБМ.

16. НЕЙРОННА МЕРЕЖА CMAC

У 1972 р. *Дж. Альбусом* запропонована модель, що описує процеси керування рухом, що відбуваються в мозочку, яка була реалізована в нейромережевому регуляторі для керування роботом-маніпулятором, названим ним *CMAC — Cerebellar Model Articulation Controller* (церебральна модель артикуляційного контроллера) [111, 112]. Згодом модель стала основою для розробки нейромережевої асоціативної пам'яті, що називається також CMAC, однак у цьому випадку абревіатура нерідко розшифровується як *Cerebellar Model Arithmetic Computer* (церебральна модель арифметичного комп'ютера). Отже, призначена спочатку для керування роботом-маніпулятором мережа стала згодом застосовуватися переважно як асоціативна пам'ять у задачах апроксимації, відновлення й інтерполяції одно- і багатовимірних функцій. Зазначимо, що на відміну від інших мереж традиційна CMAC здійснює кусково-сталу апроксимацію.

16.1. Принцип роботи мережі

Принцип роботи даної мережі як асоціативної пам'яті полягає в наступному. Функція $y = f(x)$, що запам'ятовується (апроксимована), задається обмеженою кількістю точок (значень аргументів) $\boldsymbol{x}$, що утворюють N-вимірний простір вхідних сигналів. Даний простір розбивається на M підпросторів, утворених вхідними сигналами $\boldsymbol{x}(i)$ $(i=\overline{1,M})$. При побудові моделі мозочка Альбус виходив з того факту, що поява збуджувального сигналу активує не один його нейрон, а декілька, тобто з появою вхідного сигналу збуджується певна ділянка мозочка, або рецептивне поле, характеризоване деяким параметром ρ. Тому для зберігання значень функції $y(i)$ (вихідних сигналів мережі), що відповідають $\boldsymbol{x}(i)$ $(i=\overline{1,M})$, використовується ρ комірок пам'яті, кількість яких є постійною для всіх векторів вхідних сигналів даної мережі. При надходженні на вхід

мережі деякого сигналу $\boldsymbol{x}(i)$ на її виході з'являється сигнал $y(i)$, що являє собою суму вмісту ρ адресованих комірок. Модель CMAC для випадку ρ = 4 наведено на рис. 16.1.

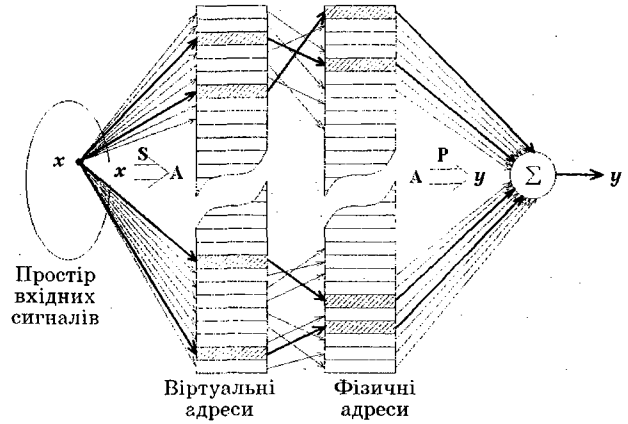


Рис. 16.1. Модель CMAC

Асоціативні властивості CMAC проявляються у використовуваному вигляді адресації, заснованому на спеціальному кодуванні вхідної інформації.

16.2. Структура мережі CMAC

Мережа, приклад якої наведено на рис. 16.2, складається із вхідного, прихованого і вихідного шарів, позначених L1, L2, L3 відповідно.

Шар L1 містить R_i рівнів квантування з відповідним кроком квантування r_i за кожним компонентом вхідного сигналу. На рис. 16.2 для випадку $x = (0{,}83\ \ 1{,}36)^T$, тобто $N = 2$, наведено однакову кількість рівнів квантування $R_1 = R_2 = 19$ із кроками квантування відповідно r_1 й r_2 (тут $r_1 = r_2 = 0{,}1$, що відповідає рівномірній дискретизації). Крім того, кожен рівень квантування i-ї компоненти має однакову кількість ρ степенів квантування ($C_1^i, C_2^i, \ldots, C_\rho^i$), кожна з яких включає $\rho' \le \rho$ областей квантування, позначуваних

проміжними змінними $A, B, ..., a, b, ...$ і які є деякими заздалегідь обраними базисними функціями. Області квантування можуть розглядатися як нейрони вхідного шару, а базисні функції — як їхні функції активації. N-вимірний сигнал $\boldsymbol{x}$, що надходить на вхід мережі, активізує $N\rho$ нейронів шару L1 (кожен компонент $\boldsymbol{x}_i$ збуджує ρ нейронів — за одним нейроном у кожному степені квантування для відповідного рівня квантування). Так, у прикладі, наведеному на рис. 16.2, збудженими є нейрони $B, G, K, O, T, c, h, l, q, u$. На виходах збуджених нейронів з'являються сигнали, величина яких визначається значенням прийнятої функції активації, на виходах незбуджених нейронів сигналу немає. У традиційній мережі CMAC базисні функції вибиралися прямокутними, що призводить до кусково-сталої апроксимації. Отже, вихідними сигналами нейронів шару L1 і вхідними шару L2 є значення використовуваних функцій активації.

Другий шар L2 утворюють асоціативні нейрони, з'єднані з певними нейронами шару L1 і позначувані комбінаціями змінних, складених з позначень відповідних нейронів вхідного шару. Максимальна кількість асоціативних нейронів можна визначити за формулою

$$n_{\max} = \left\rfloor \rho \left(\frac{R-1}{\rho} + 1 \right)^{N} \right\lfloor , \qquad (16.1)$$

де R — використовуване число рівнів квантування вхідних сигналів; N — розмірність вхідного вектора; $\rfloor \cdot \lfloor$ означає заокруглення у бік найближчого більшого цілого числа.

Кожний нейрон шару L2 має своє рецептивне поле, що містить крім цього нейрона ($\rho - 1$) асоціативний нейрон даного шару. На рис. 16.2 пунктиром зображено рецептивні поля збуджених асоціативних нейронів.

Поява на вході шару L2 відмінних від нуля сигналів призводить до збудження ρ асоціативних нейронів і формування на його виході сигналу, що є добутоком сигналів, які надійшли від нейронів шару L1. На рис. 16.2 наведено вектор асоціацій, п'ять компонент якого ($\rho = 5$), позначених хрестиками, відмінні від нуля. Значення цих компонент залежать від обраних активаційних функцій нейронів.

Істотного зменшення використовуваної пам'яті можна досягти шляхом гешування інформації [10, 11]. При використанні гешування отриманий у шарах L1 й L2 вектор асоціацій $\boldsymbol{a}$ необхідно перетворити, застосовуючи деякий алгоритм гешування, у новий вектор $\boldsymbol{a}' = H(\boldsymbol{a})$ меншої розмірності. Тут $H(\cdot)$ — перетворення,

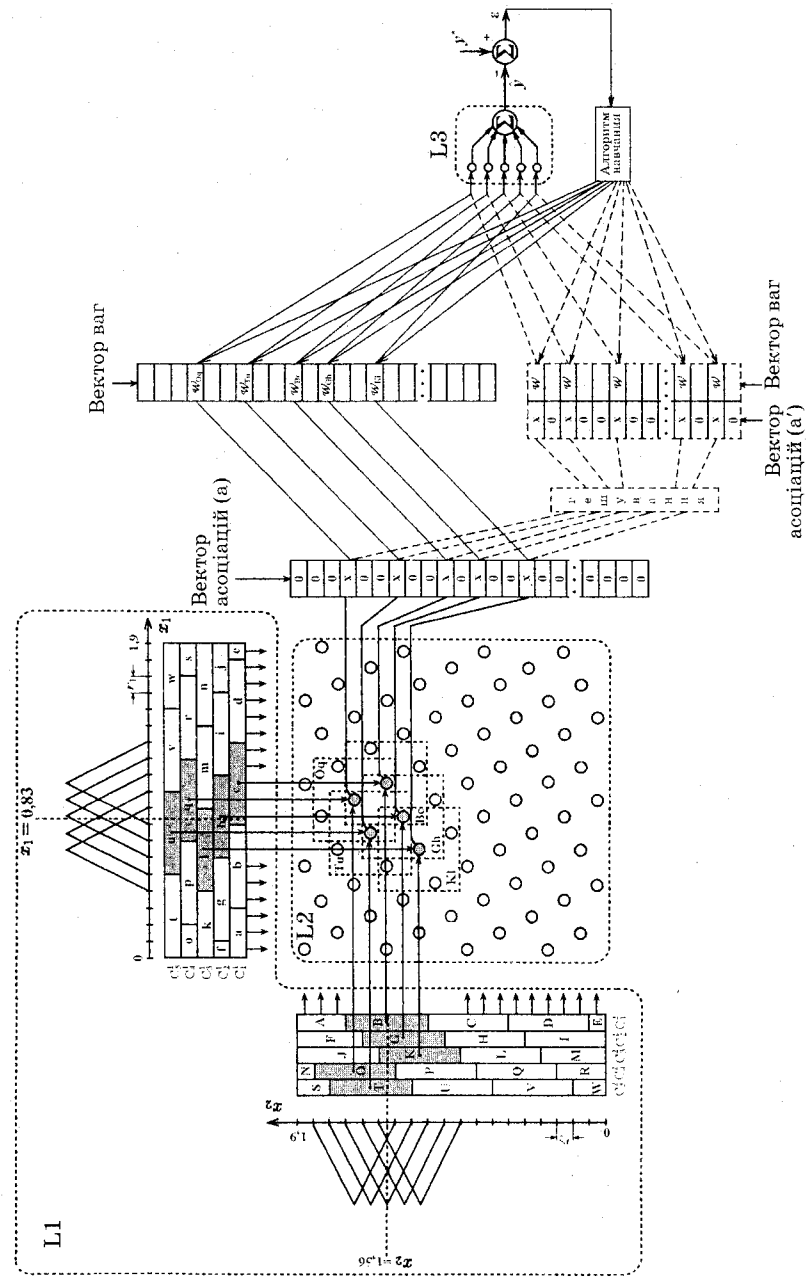


Рис. 16.2. Приклад мережі CMAC

здійснюване алгоритмом гешування. У векторі a' число ненульових компонентів може бути менше ρ (відповідні блоки й зв'язки при гешуванні показано у правій нижній частині рис. 16.2 пунктирними лініями).

За відсутності гешування кожен нейрон вихідного шару L3 з'єднаний з асоціативними нейронами другого шару. Кожен такий зв'язок має власну вагу w_i $(i=\overline{1,n})$, яка є настроюваним параметром мережі, що обчислюється у процесі її навчання. Отже, нейрони третього шару обчислюють зважену суму вихідних сигналів нейронів другого шару.

При використанні ж алгоритмів гешування нейрони вихідного шару вже не пов'язані безпосередньо з асоціативними нейронами. У цьому випадку нейрони вихідного шару обчислюють зважену суму ненульових компонентів вектора a'.

Отже, у загальному випадку мережа CMAC здійснює перетворення

$$S: X \Rightarrow A, \tag{16.2}$$

$$H: A \Rightarrow A', \tag{16.3}$$

$$P: A' \Rightarrow y, \tag{16.4}$$

де X — N-вимірний простір неперервних вхідних сигналів; A — n-вимірний простір асоціацій; A' — перетворене алгоритмом гешування простір асоціацій; y — вектор вихідних сигналів.

Перетворення (16.2) відповідає кодуванню інформації (шари L1 й L2)

$$a = S(x), \tag{16.5}$$

(16.3) — гешуванню

$$a' = H(a),$$

а (16.4) — обчисленню вихідного сигналу (шар L3)

$$\hat{y} = P(a') = (a')^T w = (H(a))^T w. \tag{16.6}$$

Вираз (16.6) описує перетворення, здійснюване в традиційній CMAC із використанням гешування інформації при виборі прямокутних базисних функцій. Якщо ж у мережі використовуються нейрони з активаційними функціями, відмінними від прямокутної, перетворення (16.6) набуває вигляду

$$\hat{y} = H(a^T \Phi(x))w, \tag{16.7}$$

де $\Phi(x) = \begin{bmatrix} \Phi_1(x) & 0 & \dots & 0 \\ 0 & \Phi_2(x) & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \Phi_n(x) \end{bmatrix}$.

Тут $\Phi_i(x) = \prod_{j=1}^{N} \phi_{ij}(x_j)$; $\phi_{ij}(x_i)$ — значення обраної базисної функції у точці x_j.

Для традиційної CMAC $\Phi(x) = I$ (I — одинична матриця).

16.3. Кодування інформації

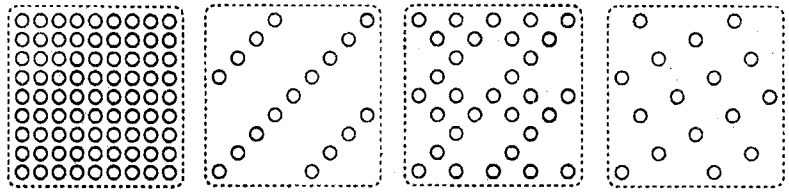
Характерною рисою CMAC є використання в ній спеціального кодування, що дозволяє різко зменшити обсяг пам'яті, необхідний для зберігання оброблюваної інформації.

Кодування інформації, здійснюване в шарах L1 й L2, полягає в тому, що кожному N-вимірному вхідному вектору $x(i)$ ставиться у відповідність n-вимірний вектор асоціацій $a(i)$, що формується шаром асоціативних нейронів, елементи якого можуть приймати значення з інтервалу [0, 1]. При цьому тільки $\rho << n$ елементів даного вектора мають відмінні від нуля значення, тобто тільки ρ елементів пам'яті є активними.

Кодування здійснюється у два етапи: на першому — у шарі L1 неперервна множина вхідних сигналів шляхом дискретизації (квантування за рівнем) перетвориться в дискретну, а на другому (у шарі L2) відбувається обчислення коду вхідного сигналу. При цьому важливим є взаємне розташування нейронів вхідного шару, що відносяться до різних степенів квантування, оскільки воно однозначно визначає схему кодування інформації. Проблема вибору схеми кодування інформації виникає під час використання базисних функцій з формою, відмінною від прямокутної.

На рис. 16.3, *а* зображено схему, що відповідає табличному методу апроксимації функції, отримуваної у результаті вибору $\rho = 1$. Витрати пам'яті в цьому випадку будуть максимальними тому, що для зберігання інформації про апросимовну функцію знадобиться R^N (*див.* формулу (16.1)) комірок пам'яті. Перевагою такої схеми є відсутність необхідності навчання мережі, тому що в цьому випадку достатньо записати в комірки пам'яті відповідні значення функції.

Зображена на рис. 16.3, *б* схема кодування є найпростішою. Однак під час її використання рецептивні поля асоціативних нейронів, що перебувають на сусідніх діагоналях, не перетинаються, у результаті чого згладжувальні властивості мережі значно погіршуються.



а *б* *в* *г*

Рис. 16.3. Приклади різних схем кодування інформації

На сьогодні не існує рекомендацій щодо вибору оптимальної схеми кодування інформації. Її вибір здійснюється експериментальним шляхом. При цьому, однак, необхідно дотримувати обмеження на розташування нейронів вхідного шару, яке полягає в тому, що будь-який нейрон деякого степеня C_k^i повинен мати зв'язки не більш ніж з $(\rho - 1)$ нейроном сусідніх степенів C_{k-1}^i й C_{k+1}^i. Це відповідає обмеженню на максимально припустиму загальну кількість компонент вектора асоціацій, що дорівнює $(\rho - 1)$, використовуване для кодування двох різних векторів вхідних сигналів, при якому ще можливо їхнє розпізнавання.

Кожний степінь містить не більше ρ областей квантування й характеризується відповідною матрицею асоціації $A^i\,(i=\overline{1,\rho})$, тільки один елемент якої відрізняється від нуля. Побудова вектора асоціацій здійснюється у такий спосіб. Для всієї заданої множини вхідних сигналів формуються матриці асоціацій кожного степеня квантування $A^i\,(i=\overline{1,\rho})$, стовпці яких й утворять вектори асоціацій $\boldsymbol{a}_i\,(i=\overline{1,\rho})$. Розмірність цих векторів n, рівна сумі всіх матриць $A^i\,(i=\overline{1,\rho})$, може бути обчислена за формулою (16.1).

Приклад 16.1. Розглянемо процес кодування двовимірних векторів $(N = 2)$ $\boldsymbol{x}(1) = (2\ 1)^T$; $\boldsymbol{x}(2) = (2\ 2)^T$; $\boldsymbol{x}(3) = (3\ 2)^T$; $\boldsymbol{x}(4) = (4\ 6)^T$, якщо $\rho = 3$ (рис. 16.4).

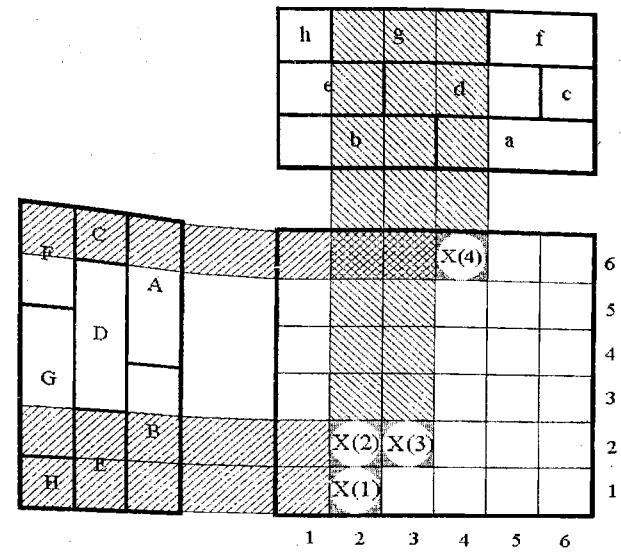


Рис. 16.4. Кодування двовимірного вектора

Відповідні матриці й вектори асоціацій мають вигляд

$\boldsymbol{x}(1) = (2\ 1)^T$

$$A_1^1 = \begin{array}{c} \begin{matrix} A & B \end{matrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \end{array} \begin{matrix} a \\ b \end{matrix}; \quad A_2^1 = \begin{array}{c} \begin{matrix} C & D & E \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} c \\ d \\ e \end{matrix}; \quad A_3^1 = \begin{array}{c} \begin{matrix} F & G & H \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} f \\ g \\ h \end{matrix};$$

$$\boldsymbol{a}(1) = (0\,0 \vdots 0\,1 \vdots 0\,0\,0 \vdots 0\,0\,0 \vdots 0\,0\,1 \vdots 0\,0\,0 \vdots 0\,0\,0 \vdots 0\,1\,0)^T;$$

$\boldsymbol{x}(2) = (2\ 2)^T$

$$A_1^2 = \begin{array}{c} \begin{matrix} A & B \end{matrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \end{array} \begin{matrix} a \\ b \end{matrix}; \quad A_2^2 = \begin{array}{c} \begin{matrix} C & D & E \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} c \\ d \\ e \end{matrix}; \quad A_3^2 = \begin{array}{c} \begin{matrix} F & G & H \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} f \\ g \\ h \end{matrix};$$

$$\boldsymbol{a}(2) = (0\,0 \vdots 0\,1 \vdots 0\,0\,0 \vdots 0\,0\,0 \vdots 0\,0\,1 \vdots 0\,0\,0 \vdots 0\,1\,0 \vdots 0\,0\,0)^T;$$

$\underline{x(3) = (3\ 2)^T}$

$$A_1^3 = \begin{matrix} & A & B \\ a & 0 & 0 \\ b & 0 & 1 \end{matrix}; \quad A_2^3 = \begin{matrix} & C & D & E \\ c & 0 & 0 & 0 \\ d & 0 & 0 & 1 \\ e & 0 & 0 & 0 \end{matrix}; \quad A_3^3 = \begin{matrix} & F & G & H \\ f & 0 & 0 & 0 \\ g & 0 & 1 & 0 \\ h & 0 & 0 & 0 \end{matrix};$$

$$a(3) = (0\,0 \vdots 0\,1 \vdots 0\,0\,0 \vdots 0\,0\,0 \vdots 0\,1\,0 \vdots 0\,0\,0 \vdots 0\,1\,0 \vdots 0\,0\,0)^T;$$

$\underline{x(4) = (4\ 6)^T}$

$$A_1^4 = \begin{matrix} & A & B \\ a & 1 & 0 \\ b & 0 & 0 \end{matrix}; \quad A_2^4 = \begin{matrix} & C & D & E \\ c & 0 & 0 & 0 \\ d & 1 & 0 & 0 \\ e & 0 & 0 & 0 \end{matrix}; \quad A_3^4 = \begin{matrix} & F & G & H \\ f & 0 & 0 & 0 \\ g & 1 & 0 & 0 \\ h & 0 & 0 & 0 \end{matrix};$$

$$a(4) = (1\,0 \vdots 0\,0 \vdots 0\,1\,0 \vdots 0\,0\,0 \vdots 0\,0\,0 \vdots 0\,1\,0 \vdots 0\,0\,0 \vdots 0\,0\,0)^T.$$

Отримані вектори асоціацій містять по три одиниці (ρ = 3) і мають розмірність 22 × 1.

У розташованих близько векторів $x(1)$, $x(2)$ матриці асоціацій $A_1^1 = A_1^2$, $A_2^1 = A_2^2$, тобто відповідні компоненти векторів асоціацій $a(1)$ і $a(2)$, співпадають і зберігаються в одних комірках. Менше збігів у $x(1)$ і $x(3)$, і нарешті, далеко розташований вектор $x(4)$ не має з $x(1)$, $x(2)$, $x(3)$ загальних компонент, тому його компоненти зберігаються в окремих комірках.

16.4. Вибір базисних функцій

Вибір базисних функцій нейронів шару L1 істотно впливає на апроксимуючі властивості мережі CMAC. Як ми вже зазначали, традиційна CMAC здійснює кусково-сталу апроксимацію, що є наслідком використання в ній нейронів із прямокутною активаційною функцією.

Під час вибору базисних функцій прямокутної форми (рис. 16.5, *а*) обчислювальні витрати будуть мінімальними. Це обумовлено тим, що у вхідному шарі замість обчислень значень базисних функцій для кожного збудженого нейрона достатньо визначити,

які нейрони збуджені, і подати на їхні виходи сигнал, рівний 1. У шарі асоціативних нейронів також не відбувається обчислень, а визначаються збуджені нейрони й з їхніх виходів знімається сигнал «1». Внаслідок цього значно скорочується час реакції мережі на вхідний сигнал, що надійшов. Компоненти вектора асоціацій у цьому випадку можуть приймати значення «0» або «1». Швидкість навчання мережі під час вибору базисних функцій прямокутної форми буде максимальною. Істотним також є той факт, що в мережі CMAC із базисними функціями прямокутної форми схема кодування інформації не впливає на точність апроксимації, тобто будь-яка схема кодування є прийнятною.

На рис. 16.5 наведено активаційні функції збуджених нейронів *c*, *h*, *l*, *q*, *u* (*див.* рис. 16.2), що беруть участь у кодуванні координати $x = (0{,}83\ \ 1{,}36)^T$, подані у вигляді B-сплайнів першого (рис. 16.5, *а*), другого (рис. 16.5, *б*) і четвертого (рис. 16.5, *в*) порядків. Безсумнівною перевагою B-сплайнів є можливість рекурентного обчислення як самих B-сплайнів відповідно до формули

$$B_{n,j}(x) = \left[\frac{x - \lambda_{j-n}}{\lambda_{j-1} - \lambda_{j-n}}\right] B_{n-1,j-1}(x) + \left[\frac{\lambda_j - x}{\lambda_j - \lambda_{j-n+1}}\right] B_{n-1,j}(x), \qquad (16.8)$$

так і їхніх похідних δ-го порядку

$${}^{(\delta)}B_{n,j}(x) = \frac{(n-1)}{(n-\delta-1)} \left\{ \left[\frac{x - \lambda_{j-n}}{\lambda_{j-1} - \lambda_{j-n}}\right]^{(\delta)} B_{n-1,j-1}(x) + \left[\frac{\lambda_j - x}{\lambda_j - \lambda_{j-n+1}}\right]^{(\delta)} B_{n-1,j}(x) \right\}. \qquad (16.9)$$

$$\text{Тут } B_{0,j}(x) = \begin{cases} 1, & \text{якщо } x \in [\lambda_{j-1}, \lambda_j); \\ 0, & \text{в іншому випадку;} \end{cases}$$

$${}^{(0)}B_{0,j}(x) = \begin{cases} 1, & \text{якщо } x \in [\lambda_{j-1}, \lambda_j); \\ 0, & \text{в іншому випадку;} \end{cases}$$

$${}^{(\delta)}B_{n,j}(x) = \left[\frac{{}^{(\delta-1)}B_{n-1,j-1}(x)}{\lambda_{j-1} - \lambda_{j-n}}\right] - \left[\frac{{}^{(\delta-1)}B_{n-1,j}(x)}{\lambda_j - \lambda_{j-n+1}}\right];$$

λ_j — *j*-й вузол сплайна (центр області квантування).

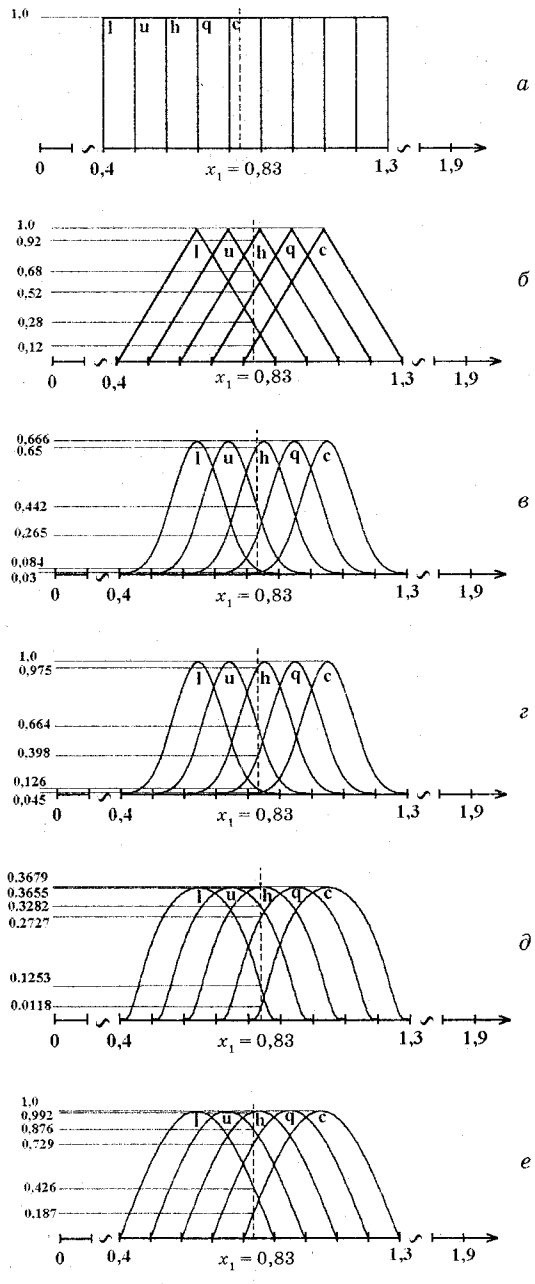


Рис. 16.5. Активаційні функції збуджених нейронів

Таким чином, після визначення активного інтервалу $(\lambda_{j-1},\lambda_j]$ для B-сплайна першого порядку дані вирази можуть бути використані для отримання значень усіх B-сплайнів більш високих порядків й, за необхідності, їхніх похідних.

На рис. 16.5, *г* наведено гауссівські активаційні функції цих же збуджених нейронів, що мають вигляд

$$\Phi_i(x_j) = \exp\left\{-\frac{(x_j-\mu_i)^2}{\sigma^2}\right\}. \qquad (16.10)$$

Неважко побачити, що вони також допускають досить просте обчислення похідних $\frac{\partial\Phi_i}{\partial x_j}; \frac{\partial^2\Phi_i}{\partial x_j^2} \ldots$.

Однак хоча гауссівська функція й має властивість локального збудження, досить складно чітко виділити її межі збудження, що є важливим для реалізації кодування інформації в CMAC. З метою усунення цього недоліку можливе використання модифікованої гауссівської функції, що має вигляд

$$\Phi_i(x) = \begin{cases} \exp\left\{-\dfrac{(\lambda_2-\lambda_1)^2/4}{(x-\lambda_1)(\lambda_2-x)}\right\} & \text{при } x \in (\lambda_1,\lambda_2); \\ 0 \quad \text{в іншому випадку.} \end{cases} \qquad (16.11)$$

Як видно з виразу (16.11), ця функція точно визначена в інтервалі (λ_1, λ_2), що спрощує процедуру масштабування базисних функцій при зміні таких параметрів мережі, як R й ρ. На рис. 16.5, *д* показані модифіковані гауссівської активаційні функції збуджених нейронів *c*, *h*, *l*, *q*, *u*.

Аналогічний вид матимуть й базисні функції збуджених нейронів *T*, *O*, *K*, *G*, *B*, що беруть участь у кодуванні другої координати x_2 (у нашому прикладі x_2 = 1,36). Оскільки значення компонент вектора асоціацій, що з'являються на виході L2, отримується перемножуванням відповідних значень базисних функцій збуджених нейронів, то для прикладу, наведеного на рис. 16.2, вектори асоціацій матимуть вигляд:

(0 1 0 ...0 1 0 0 1 0 1 0 0 1 0...0) — при використанні базисних функцій прямокутної форми (B-сплайнів першого порядку);

(0 0,125 0 ...0 0,435 0 0 0,115 0 0,515 0 0 0,045 0...0) — B-сплайнів другого порядку;

(0 0,017 0 ...0 0,172 0 0 0,019 0 0,196 0 0 0,003 0...0) — B-сплайнів четвертого порядку;

(0 0,039 0 ...0 0,395 0 0 0,045 0 0,449 0 0 0,007 0...0) — гауссівської функції;

(0 0,025 0 ...0 0,104 0 0 0,0043 0 0,106 0 0 0,0041 0...0) — модифікованої гауссівської функції.

Як видно з наведених значень, ці базисні функції мають поблизу межі області квантування (зони чутливості нейрона вхідного шару) порівняно невеликі значення (підкреслені), що негативно позначається на результатах апроксимації. Для запобігання зазначеного недоліку в якості базисних можливе використання тригонометричних функцій, наприклад, косинусоїдної, що має вигляд

$$\Phi_i(x_j)=\begin{cases}\cos\left(\dfrac{\pi}{\rho r_j}\left(x_j-\lambda_i\right)\right) \text{ при } x_j\in\left(\lambda_i-\dfrac{\rho r_j}{2},\lambda_i+\dfrac{\rho r_j}{2}\right];\\ 0 \quad \text{в іншому випадку,}\end{cases}$$

де λ_i — i-й центр області квантування; r_j — крок квантування за j-ю компонентою вхідного сигналу.

Вигляд даної функції наведений на рис. 16.5, *е*. Тригонометричні функції також дозволяють легко обчислювати їхні похідні. У цьому випадку для нашого прикладу вектор асоціацій матиме вигляд

(0 0,268 0 ...0 0,739 0 0 0,186 0 0,763 0 0 0,106 0...0),

тобто мінімальні значення його компонентів збільшилися.

16.5. Гешування інформації

Гешування полягає у відображенні великої необхідної пам'яті в меншу, фізично реалізовану [113]. Обсяг більшої пам'яті визначається розмірністю вектора асоціацій, отриманого на етапі кодування. Якщо адреси комірок більшої пам'яті лежать в інтервалі [1, M], а меншої — в інтервалі [1, m], то функція перетворення адреси (геш-функція) має задовольняти вимозі

$$1 \le H(k) < m,\ \forall\ 1 \le k \le M.$$

16.5.1. Алгоритми гешування інформації у CMAC

Вибір функції $H(k)$ є досить складним завданням, однак численні дослідження довели, що позитивні результати забезпечує застосування двох основних типів геш-функцій, один із яких полягає в розподілі, а інший — у множенні [113, 114].

Геш-функція, що реалізує метод, заснований на розподілі, має вигляд

$$H(k) = 1 + k \bmod m, \tag{16.12}$$

де $k \bmod m$ означає розподіл за модулем m.

Адреса комірки пам'яті з меншим обсягом при методі гешування, заснованому на множенні, обчислюється за формулою:

$$H(k) = 1 + \left\lfloor m\left[\left(\frac{F}{w}k\right)\bmod 1\right]\right\rfloor, \tag{16.13}$$

де w — розмір машинного слова; F — деяка ціла константа, взаємно проста з w; $[\cdot]$ означає заокруглення у бік найближчого цілого числа.

Як F можливе використання чисел Фібоначчі.

Геш-функція, заснована на розподілі, рівномірно розподіляє більшу пам'ять у меншу відповідно до адреси.

Функція, описувана рівнянням (16.13), «тасує» елементи пам'яті.

16.5.2. Геш-колізії

Який би метод гешування не застосовувався, неможливо повністю усунути проблему геш-колізії. При гешуванні інформації в мережі CMAC можливе виникнення двох типів колізій. Колізії першого типу виникають при активації одним вхідним сигналом однієї комірки пам'яті більше одного разу. Імовірність того, що один вхідний сигнал звернеться два й більше рази до однієї комірки фізичної пам'яті, можна визначити за формулою

$$P = \sum_{i=2}^{\rho-1}\frac{i}{m}, \tag{16.14}$$

де m — обсяг меншої пам'яті.

Так, наприклад, для випадку $\rho = 20$, $m = 2000$ імовірність багаторазового звертання до однієї комірки дорівнюватиме приблизно 0,1, тобто є досить малою. Виникнення такої колізії означає, що значення, яке зберігається в багаторазово активізованій комірці, кілька разів бере участь у формуванні вихідного сигналу.

Причиною виникнення колізій другого типу є активації однієї й тієї самої комірки пам'яті різними віддаленими один від одного (у значенні відстані Хеммінга) вхідними сигналами. Хоча ці колізії призводять до більш серйозної проблеми — появи небажаної кореляції двома далеко розташованими (у значенні відстані Хеммінга) вхідними сигналами, вони не можуть істотно вплинути на роботу мережі, поки виконується умова

$$a'_1 \wedge a'_2 << \rho,$$

де a'_1 й a'_2 — перетворені геш-функцією вектори асоціацій двох різних вхідних сигналів.

Для будь-яких двох вхідних сигналів, для яких не виконується $a'_1 \wedge a'_2 = \phi$ (тут ϕ — порожня множина), імовірності звертання до загальних комірок можна визначити у такий спосіб:

$$P_n(k) = C_n^k p^k q^{n-k}, \quad k = \overline{1,n}, \tag{16.15}$$

де C_n^k — кількість поєднань; $p = 1 - q$.

Для нашого прикладу $q = \frac{20}{2000}$. У цьому випадку ймовірність того, що $a'_1 \wedge a'_2 = 0$, тобто $P_{20}(20)$, дорівнює 0,818. Імовірності ж того, що $a'_1 \wedge a'_2 = 1$, $a'_1 \wedge a'_2 = 2$, $a'_1 \wedge a'_2 = 3$, дорівнюватимуть відповідно 0,165; 0,016; 0,001.

16.6. Навчання мережі

Навчання мережі CMAC, як практично й всіх інших ШНМ, полягає в настроюванні вектора її вагових параметрів w розмірності $n \times 1$, тобто в даній мережі настроюються тільки ваги зв'язків між асоціативними нейронами й нейронами вихідного шару. Використання спеціального кодування дозволяє зберігати в ρ комірках

пам'яті ваги мережі. Якби різні вхідні вектори при кодуванні не мали загальних комірок, то ваги мережі могли бути визначені й без її навчання, аналітично, у такий спосіб:

$$w_{ik} = \frac{y(k)}{\rho}. \tag{16.16}$$

Наявність загальних областей пам'яті для зберігання різних векторів не дозволяє обмежитися таким простим обчисленням ваг й обумовлює необхідність навчання мережі. Компоненти вектора ваг мають фізичні адреси (*див.* рис. 16.1) і вибираються таким чином, щоб середнє арифметичне значення активних ваг, тобто ваг, що відповідають одиничним компонентам вектора асоціацій, дорівнювало значенню накопичувальної функції $y(i)$. У традиційній CMAC [111], що використовує нейрони із прямокутною активаційною функцією, на кожному такті навчання мережі подаються навчальні пари $\{x(k), y(k)\}$, де $y(k)$ — значення функції, що відповідає $x(k)$,і коректуються лише ті її ρ ваг, які відповідають одиничним компонентам вектора асоціацій для даного вектора $x(k)$. При цьому правило навчання для всіх i, j, для яких $a_i(k) = a_j(k) = 1$, має вигляд

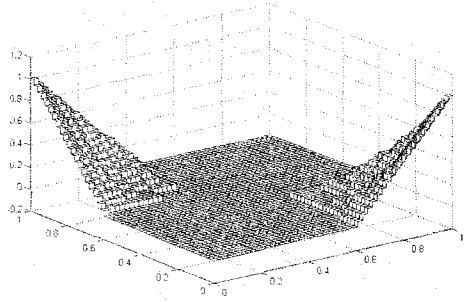
$$w_j(k+1) = w_j(k) + \gamma\left(y(k) - \frac{1}{\rho}\sum_{i=1}^{n} w_i(k)\right), \tag{16.17}$$

де $\gamma \in (0, 1]$— параметр, що впливає на швидкість навчання.

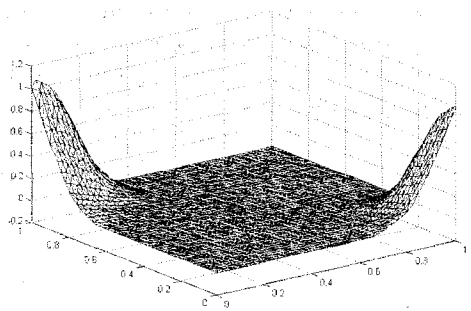
Для нейронів з відмінною від прямокутної активаційною функцією даний алгоритм може бути записаний у такий спосіб:

$$w(k+1) = w(k) + \gamma\left(\frac{y(k) - a^T(k)\Phi(x)w(k)}{\|\Phi(x)a(k)\|^2}\Phi(x)a(k)\right). \tag{16.18}$$

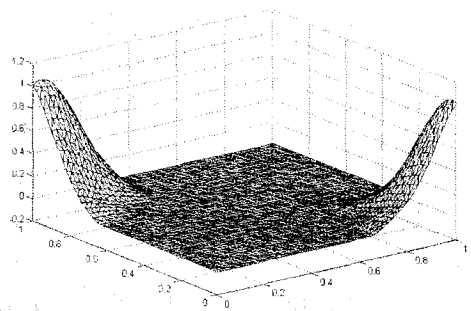
Приклад 16.2. На рис. 16.6 наведено реалізацію мережею CMAC функції «що виключає АБО». При цьому рис. 16.6, *а* відповідає вибору прямокутних, рис. 16.6, *б* — трикутних, а рис. 16.6, *в* — косинусоїдних функцій активації.



а



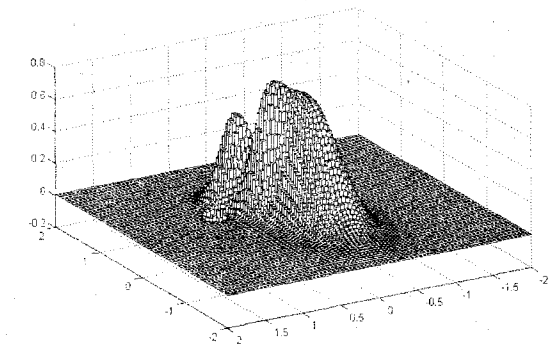
б



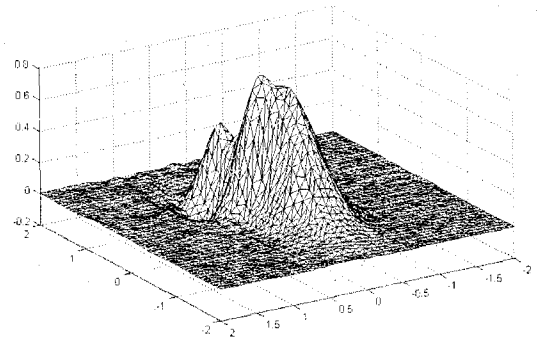
в

Рис. 16.6. Реалізація мережею CMAC функції «що виключає АБО»

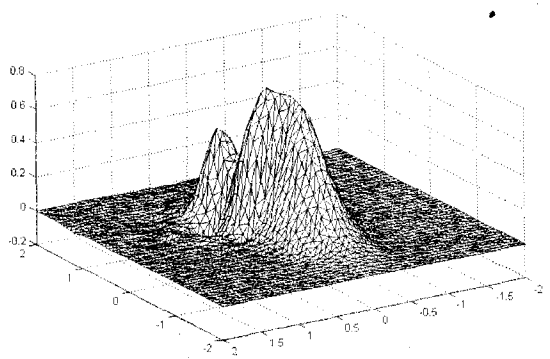
Приклад 16.3. На рис. 16.7 зображено відновлення мережею CMAC функції, заданої в прикладі 3.11, під час вибору прямокутних (рис. 16.7, а), трикутних (рис. 16.7, б)



а



б



в

Рис. 16.7. Відновлення мережею CMAC функції, заданої у прикладі 3.11

і косинусоїдних (рис. 16.7, *в*) функцій активації. Хоча, як випливає з рисунків, вибір функції активації будь-якого вигляду забезпечує відновлення функції.

Контрольні запитання

1. У чому полягає принцип роботи мережі?
2. Яка структура мережі CMAC?
3. У який спосіб здійснюється кодування інформації в мережі?
4. Яким чином відбувається вибір базисних функцій? На що впливає вигляд обраної БФ?
5. Що таке гешування інформації? Назвіть методи гешування й поясніть їхні відмінності.
6. Внаслідок чого можуть виникати геш-колізії?
7. Як відбувається навчання мережі?

17. НЕОКОГНІТРОН

Система зорового сприйняття людини є універсальним механізмом, здатним виділяти одні об'єкти на фоні інших і розпізнавати різні зашумлені, частково приховані або спотворені об'єкти незалежно від їхнього місця розташування, орієнтації, розмірів, кольору й т. д. Тому досить привабливою завжди була ідея технічної реалізації такої системи. Хоча більшість спроб у цьому напрямку були невдалими, деякі результати виявилися багатообіцяючими. Зокрема, розроблена *К. Фукушімою* спеціальна архітектура ШНМ, *неокогнітрон*, дозволяла розпізнавати обмежену кількість об'єктів, у тому числі спотворених, незалежно від їхнього розташування на загальному зображенні.

Неокогнітрон став розвитком *когнітрона* [61], що моделював процес здійснюваного в мозку людини розпізнавання зорових образів і являв собою багатошарову мережу прямого поширення, що навчається без учителя (використовувався принцип конкурентного навчання). Однак, як й інші ШНМ, що моделювали роботу системи зорового сприйняття, він був чутливий до спотворення інформації. Так, навчений когнітрон не міг розпізнати об'єкт, якщо той змінив своє місцерозташування або з будь-якої причини деформувався. Подальшим розвитком когнітрона, що усував недоліки останнього, виявився неокогнітрон [62–64].

Як ми вже зазначали, неокогнітрон був запропонований Фукушімою для розпізнавання зображень і використав принципи функціонування зорової системи ссавців, дослідження якої показало, що в ссавців є візуальний кортекс, який складається із групи нейронів головного мозку, що вибірково реагують на лінії й ребра певної орієнтації. Наступні ділянки клітин реагують на більш складні фігури, такі як кола, трикутники, квадрати і т. д. Клітини, що перебувають у вищому шарі, реагують на дуже складні фігури (обличчя, букви й ін.). Фукушіма спробував реалізувати таку обробку інформації в неокогнітроні. Пізніші версії неокогнітрона мають властивості керування вибірною увагою.

17.1. Структура неокогнітрона

Неокогнітрон має ієрархічну структуру, утворену різними *сходинками* (*stages*), кожна з яких, за винятком вхідної, що є вхідним шаром, складається у свою чергу із двох шарів (*layers*): *S*-шару й *C*-шару. Сигнали, що описують досліджуване зображення, надходять на вхідний шар і розповсюджуються через наступні шари до вихідного, причому якщо початкові шари розрізняють деякі прості елементи (фрагменти) образів, то наступні вже можуть розрізняти складніші елементи, а останній, вихідний, шар ідентифікує всі наведені на зображенні об'єкти. Пізніші модифікації неокогнітрона мали вже зворотні зв'язки, використовувані однак не для рекурсивних обчислень, здійснених, наприклад, у динамічних рекурсивних мережах (*див.* розділ 9).

Вхідний, *сенсорний*, шар неокогнітрона є прямокутним полем, що складається зі світлочутливих клітин. Розмір поля залежить від кількості використовуваних клітин і може коливатися від 8×8 до 128×128. Найчастіше використовуваними є поля 16×16 й 32×32.

Кожен наступний шар у свою чергу складається із груп нейронів, причому нейрони однієї групи мають однакові ваги й розпізнають ті самі частини образу. Ці нейрони утворюють так звані *S*-клітини *або S*-площини (*plane*) («*S*» означає *simple* — простий), розташовані у відповідних *S*-шарах у вигляді прямокутних полів, клітин.

S-клітини служать для виділення властивостей (фрагментів) образів, що надходять із попереднього шару. Зв'язки між вхідним і першим *S*-шаром організовані спеціальним способом, на якому ми зупинимося дещо пізніше.

Виходи нейронів *S*-шару з'єднані із входами нейронів *C*-шару, утвореного *C*-клітинами або *C*-площинами («*C*» служить для позначення *complex* — складних нейронів).

Зв'язки, що з'єднують *S*-клітини із *C*-клітинами однієї сходинки, є міцними й не змінюються в процесі навчання. Отже, неокогнітрон складається з модулів, кожний з яких утворений двома шарами: *S* і *C*. Для того, щоб розрізняти шари, вводять позначення *U* (вхідний шар), *S*1, *C*1, *S*2, *C*2 і т. д. до останнього, яким є шар *C*.

Зв'язки, які з'єднують нейрони *S*-шару з нейронами попереднього *C*-шару, є такими, що змінюються й можуть модифікуватися в процесі навчання. *S*-клітина активізується тільки тоді, коли знайдений певний образ (*feature*) у точно визначеному місці попереднього *C*-шару. Образ, на який реагує клітина, визначається в процесі навчання. Це відповідає певному образу (фрагменту складного образу) вхідного шару.

На рис. 17.1 наведено структуру неокогнітрона, що складається із вхідного шару й трьох модулів, утворених парами *S*- і *C*-шарів.

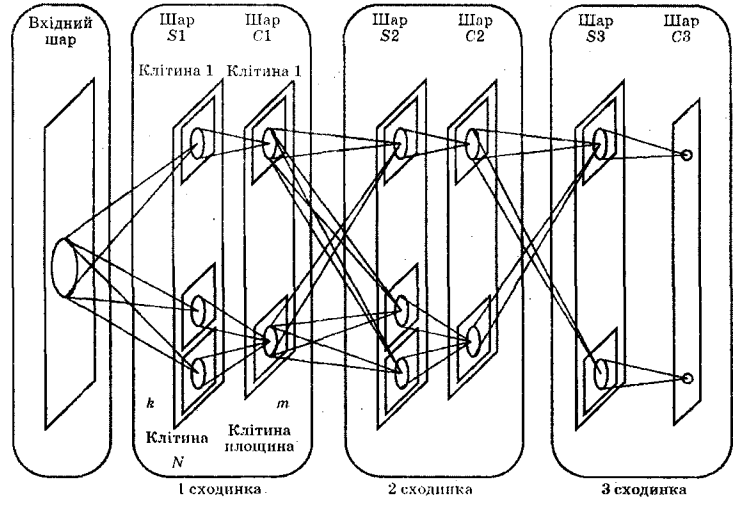


Рис. 17.1. Структура неокогнітрона

Слід зазначити, що кількість шарів неокогнітрона залежить від складності розв'язуваного завдання й мережа з більшою кількістю шарів здатна розпізнавати складніші зображення. Таке чергування шарів, при якому в шарі *S* виділяються основні ознаки (фрагменти) зображення, а в шарі *C* відбувається корекція спотворень зображень, виявилося найбільш ефективним під час синтезу систем розпізнавання образів, інваріантних до деформацій зображень.

На рис. 17.2 показано перші три шари неокогнітрона.

S-клітини шару *S*1 призначені для розпізнавання визначених фрагментів зображення. Тому в процесі навчання кожна клітина цього шару вчиться реагувати на конкретний фрагмент. Такими фрагментами можуть бути розташовані, наприклад, під різними кутами відрізки прямих, кути й т. д. (рис. 17.3) [34]. Отже, кількість клітин шару *S*1 не має бути меншою кількості фрагментів зображення, яке розпізнається.

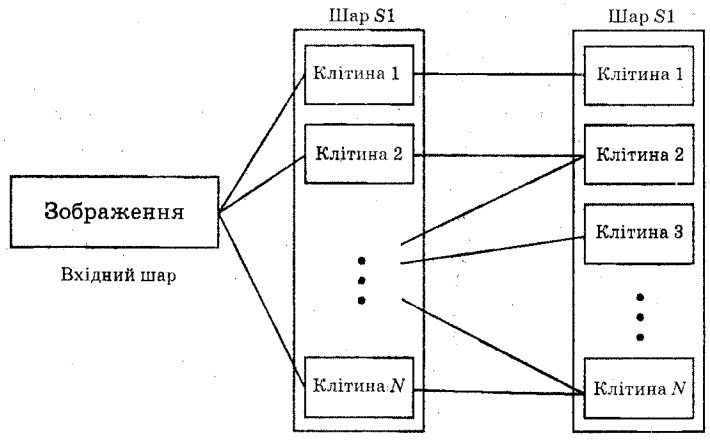


Рис. 17.2. Перші три шари неокогнітрона

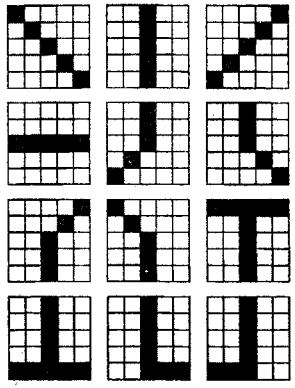


Рис. 17.3. Приклади фрагментів,
які розпізнаються нейронами шару $S1$

Усі клітини S-шару реагують на ті самі фрагменти зображень. Тому незалежно від ступеня зашумленості наведеного зображення щонайменше одна клітина цього шару буде активована.

На кожну клітину шару $C1$ надходять сигнали від декількох клітин $S1$, що викликають відповідну реакцію даної клітини. Клітини шару $C1$ реагують на ті самі фрагменти зображення, що й клітини $S1$.

Кожна група клітин утворює двовимірний шар нейронів, кожний з яких реагує на той самий образ або його фрагмент, що надходить із попереднього шару. Нейрони однієї групи різняться лише розташуванням своїх рецептивних полів, тобто частиною попереднього шару, з яким вони пов'язані (рис. 17.4).

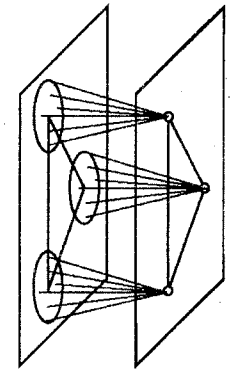


Рис. 17.4. Нейрони однієї групи

Відповідні ваги всіх нейронів однієї групи є однаковими й запам'ятовуються один раз.

17.2. Процес розпізнавання образу

Як ми вже зазначали, неокогнітрон, навчившись, може розпізнавати зображення не тільки неспотворені, але й деформовані під впливом різних завад.

17.2.1. Розпізнавання неспотвореного образу

Фрагменти зображення, що надходять із вхідного, сенсорного шару, надходять на нейрони шару $S1$, вихідні сигнали яких подаються на входи клітин шару $C1$. C-клітина активується, якщо активною є одна із S-клітин, пов'язаних з даною C-клітиною. Як довів *P. Гехт-Нільсен* [115], сила зв'язку між S- і C-клітинами залежить від положення S-клітин усередині рецептивного поля C-клітин і для клітин, що перебувають усередині рецептивного поля, вона більше, ніж для клітин, розташованих поза ним.

Кожна наступна сходинка мережі за допомогою S-клітин розпізнає фрагменти образів, переданих з попередньої сходинки, і збільшує за допомогою C-клітин область, що впізнається. Остання операція є нечіткою. На рис. 17.5 наведено процес розпізнавання букви «А». У шарі $S1$ розпізнаються фрагменти зображення, наприклад, у верхній клітині цього шару — фрагмент «∧» (верхній фрагмент букви «А»). Інші фрагменти розпізнаються іншими клітинами цього шару. Комбінації різних фрагментів розпізнаються в наступних S шарах, причому в пізніших — більш складні комбінації, і так далі до вихідного C-шару, що розпізнає повністю букву «А».

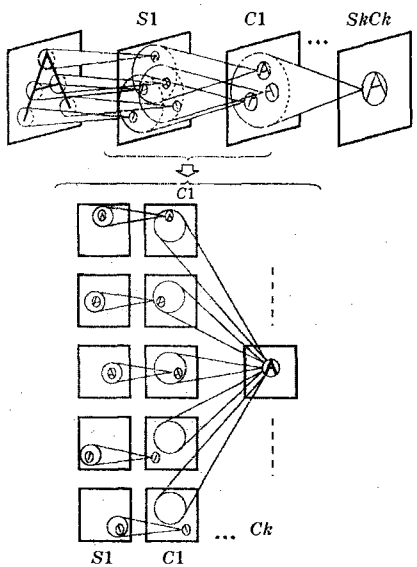


Рис. 17.5. Процес розпізнавання спотвореного образу

17.2.2. Розпізнавання спотвореного образу

C-нейрон забезпечує інваріантість мережі до зміни розташування образу. Кожен C-нейрон отримує сигнал від S-клітин, що містять інформацію про один і той самий фрагмент образу в різних позиціях. C-нейрон активується, якщо хоча б один S-нейрон активований. У свою чергу активація одного окремого C-нейрона викликає активацію ряду S-нейронів наступного шару (рис. 17.6).

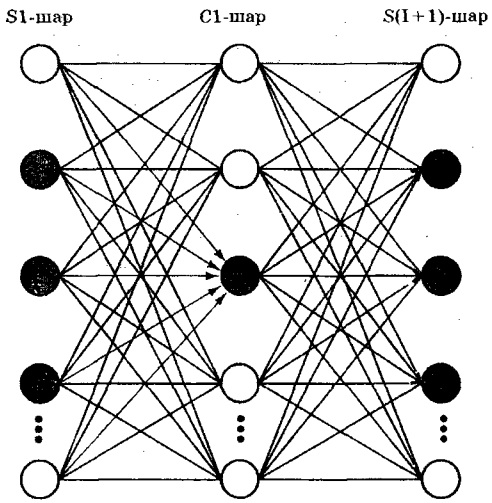


Рис. 17.6. Послідовна активація нейронів S-шарів

Завдяки цьому вихід C-нейрона є нечіткою версією (*blurred*) виходів S-нейронів. У такий спосіб можуть розпізнаватися ідентичні образи, що навчають, які зміщені один відносно одного або спотворені. Подібну ситуацію зображено на рис. 17.7.

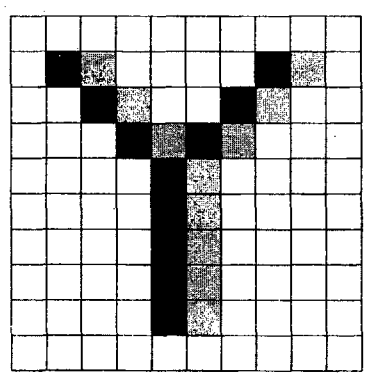


Рис. 17.7. Буква «Y» та її зміщене зображення

Обидві букви «Y», зміщені на 1 позицію, під час застосування звичайних мір подібності (близькості), наприклад, скалярного

добутку або відстані Хеммінга, не є ідентичними. Більш того, вони є ортогональними один одному — скалярний добуток дорівнює нулю. Завдяки нечіткій операції, здійснюваній *C*-нейронами, обидві букви можуть бути розпізнані як ідентичні.

Зазначимо, що нечітка операція спочатку відбувається після розпізнавання окремих фрагментів образу, тому що інакше точну структуру неможливо буде відтворити.

Принцип розпізнавання спотвореного образу демонструє рис. 17.8.

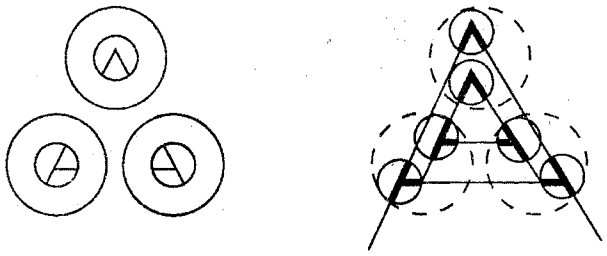


Рис. 17.8. Принцип розпізнавання спотвореного образу

Без *C*-клітин *S*-клітини могли б після навчання розпізнати три локальні образи букви «А», які розташовані в точно зазначених позиціях. Після проходження через *C*-клітини кожна область, усередині якої розпізнався образ, збільшується (штриховане коло). Завдяки цьому *S*-клітина може розпізнати букву «А», якщо вона легко деформована, збільшена або зміщена. Зазначимо, що нечітка операція *C*-клітини збільшує тільки радіус області, всередині якої розташований фрагмент образу, не змінюючи величини самого фрагмента.

17.3. Структура *S*-нейрона

S-клітини складаються з нейронів Фукушіми (*див.* розділ 2) і функціонують у такий спосіб. Зважені з вагами w_{ij} сигнали з попередньої сходинки x_i надходять на вхід *S*-клітини й підсумовуються $e = \sum_i x_i w_{ij}$.

Усі $w_{ij} > 0$, тобто дані зв'язки є *прискорювальними* або *збуджувальними* (*ексгібіторними*). Однак є один *гальмуючий* (*інгібіторний*) зв'язок зі спеціальним гальмуючим нейроном, *V*-нейроном, що також використовує позитивну вагу w_{vj}. Сигнал, що надходить по ній і є добутоком вихідного сигналу V гальмуючого нейрона і його вагового коефіцієнта $w_{ij} > 0$ з деяким додатним числом h_j

$$h_j = \frac{r}{1+r} V w_{vj}, \tag{17.1}$$

де $r > 0$ — константа, що задає силу гальмуючого *V*-нейрона, подається назустріч збуджувальному сигналу $e = \sum_i x_i w_{ij}$ (рис. 17.9).

Активність даного нейрона залежить від співвідношення величин збуджувальних і гальмуючих сигналів, тобто

$$z_j = \frac{1+e_j}{1+h_j} - 1. \tag{17.2}$$

Вихідний сигнал *S*-нейрона y_j є лінійним перетворенням сигналу $z_j > 0$

$$y_j = r f(z_j), \tag{17.3}$$

де

$$f(z_j) = \begin{cases} z_j \text{ при } z_j \geq 0; \\ 0 \text{ при } z < 0. \end{cases}$$

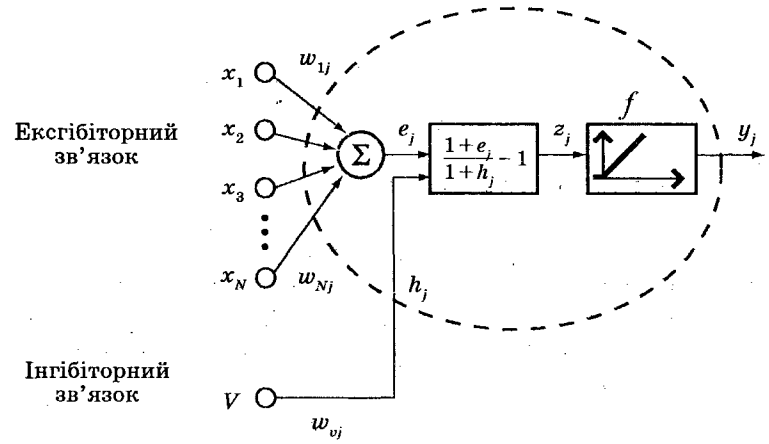


Рис. 17.9. Структура *S*-нейрона

Ексгібіторні зв'язки *S*-нейрона із *C*-нейронами попередньої сходинки можуть змінюватися в процесі навчання. Зв'язки *V*-нейрона

з попередніми C-нейронами є також збуджувальними, але вони задаються заздалегідь і згодом не змінюються. Кожна S-клітина пов'язана з асоційованою з нею V-клітиною, причому ваги, що є додатними й такими, що перестроюються у процесі навчання, діють інгібіторно.

Активація V-клітини визначається у такий спосіб:

$$V = \sqrt{\sum_i x_i^2 w_{ij}} \,. \qquad (17.4)$$

Взаємодію нейронів і відповідних зв'язків наведено на рис. 17.10. Тут суцільні лінії відповідають варіювальним екстібіторним, пунктирні — неварiювальним екстібіторним і пунктирні — варіювальним інгібіторним зв'язкам.

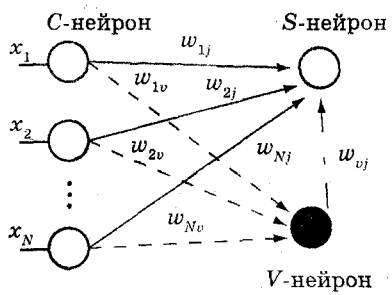


Рис. 17.10. Взаємодія нейронів

17.4. Навчання неокогнітрона

Розрізняють навчання неокогнітрона без учителя (*unsupervised*) та з учителем (*supervised*). За будь-якого виду навчання настроюються тільки ваги шарів S. Ваги C-шарів вибираються заздалегідь і надалі не змінюються. Процес навчання відбувається послідовно, починаючи з навчання шару $S1$ і завершуючи навчанням останнього S-шару.

17.4.1. Навчання без учителя

Даний метод навчання є найбільш повільним. Ідея його полягає в заохоченні найбільш активних нейронів. При такому навчанні

підсилюються зв'язки нейронів тільки тієї групи, що найсильніше реагує на вхідний сигнал. Це посилення пропорційне величині вихідного сигналу нейрона, з яким дана група пов'язана. Хоча одночасно підсилюється й гальмуючий зв'язок відповідного V-нейрона, це посилення незначне. Отже, даний вид навчання є конкурентним (*competition learning*). Посилення зв'язку як реакція на стимул наведено на рис. 17.11.

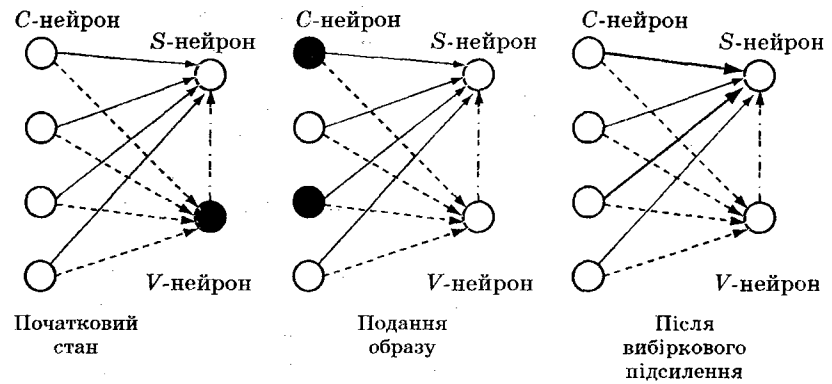


Рис. 17.11. Посилення зв'язків під час навчання без учителя

Перед початком навчання вагам S-шару привласнюють малі різні додатні значення й задаються незмінні в подальшому ваги C-шару. Потім із вхідного шару на шар $S1$ у будь-якій послідовності подаються фрагменти образів, що навчають, і вивчається реакція нейронів кожної клітини. Нейрон кожної клітини, що найсильніше реагує на деякий фрагмент, вибирається представником відповідної клітини. Потім із всіх клітин шару $S1$ вибирається клітина, реакція якої на певний образ є найсильнішою, і відбувається подальше навчання її нейрона-представника з метою вироблення ще більш сильної його реакції на даний фрагмент образу. Після навчання ваги всіх інших нейронів даної клітини дорівнюють вагам нейрона-представника.

Потім подається новий фрагмент образу й вибирається новий представник іншої клітини, що навчається на чітке реагування з появою даного фрагмента. Значення його ваги привласнюється всім нейронам даної клітини й т. д. Процес навчання повторюють для всіх фрагментів образу, навчаючи кожному з них нову клітину.

Зміна відповідних ваг здійснюються за правилом

$$\Delta w_{ij} = \alpha w_{ij} y_i;$$

$$\Delta w_{vj} = \alpha V = \alpha \sqrt{\sum_i x_i^2 w_{iv}}, \qquad (17.5)$$

де α — параметр навчання.

Після того, як усі клітини шару $S1$ будуть навчені відповідним фрагментам, відбувається навчання шару $S2$, однак використання спеціальних (екстібіторних та інгібіторних) зв'язків між шарами дозволяє навчати клітини другого шару більш складним фрагментам.

Після посилення екстібіторних й інгібіторних зв'язків S-нейрон міститиме фрагмент образу, переданий з попереднього шару. Якщо подано відповідний фрагмент, то екстібіторні зв'язки підсилюються, і S-клітина активується, якщо ж ні, то інгібіторні зв'язки V-нейрона виявляються сильніше, і стан S-клітини не змінюється. Таким чином, V-нейрони відіграють важливу роль бар'єра при розпізнаванні образів. Оскільки клітина реагує на певний фрагмент пропонованого образу, вона має бути нечутливою до інших його фрагментів.

Чим далі розташований S-шар від вхідного, тим складніші комбінації фрагментів образів він буде здатний розпізнати.

Процес навчання завершується, коли всім образам будуть навчені всі S-шари неокогнітрона.

17.4.2. Навчання з учителем

У процесі розв'язання багатьох практичних задач образи, яких розпізнають, є заздалегідь відомими. Така ситуація виникає, наприклад, при розпізнаванні рукописного тексту. У цьому випадку кожна сходинка неокогнітрона може бути навчена окремо. Навчання кожної окремої сходинки неокогнітрона Фукушіма назвав навчанням із учителем (*supervised*). Однак тут є відмінність від звичайного навчання з учителем, використовуваного в інших ШНМ, коли мережі подаються навчальні пари вхідних і вихідних образів, на підставі яких алгоритм навчання має визначити активацію прихованих шарів. У неокогнітроні всім прихованим шарам також подають бажану активацію, вірніше, бажані вихідні сигнали. Навчання відбувається так само пошарово, як і під час навчання без учителя.

На рис. 17.12–17.14 наведено образи чотиришарового неокогнітрона, що навчають, призначеного для розпізнавання цифр 0, 1, ..., 9 [115]...

З рисунків видно, що кожному наступному шару подається більш складний фрагмент образу, а останньому — зображення цифри повністю.

Так, на рис. 17.12 зображено 38 груп фрагментів зображень, що служать як навчальні для шару $S2$ неокогнітрона. У кожному рядку наведено групи образів для S-клітин. Наведене на рисунку об'єднання рядків свідчить про те, що дані групи S-клітин з'єднані з відповідними групами C-клітин, яких, як видно з рисунка, отримуємо 19.

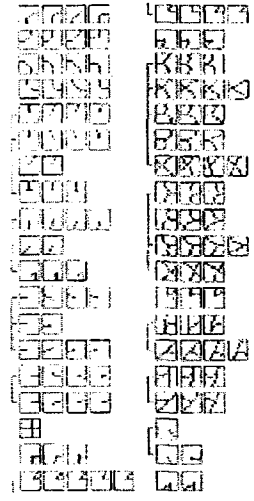


Рис. 17.12. Фрагменти образів для шару $S2$, що навчають

На рис. 17.13 наведено 35 груп фрагментів для шару $S3$ цього ж неокогнітрона. Хрестиками на рисунках позначено нейрони, які будуть активовані при розпізнаванні відповідних фрагментів.

Нарешті, на рис. 17.14 наведено повні образи, що навчають, для шару $S4$ останньої сходинки неокогнітрона. Оскільки розпізнаються 10 цифр, вихідний $C4$-шар складатиметься з 10 нейронів, кожний з яких реагуватиме на відповідну цифру. В останньому шарі врахована можливість різного написання цифри «4».

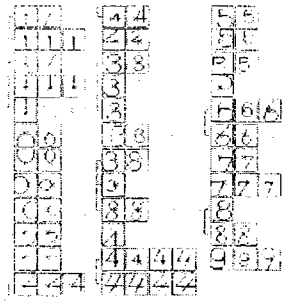


Рис. 17.13. Навчальні фрагменти для шару $S3$

Нарешті, на рис. 17.14 наведено повні образи, що навчають, для шару $S4$ останньої сходинки неокогнітрона. Оскільки розпізнаються 10 цифр, вихідний $C4$-шар складатиметься з 10 нейронів, кожний з яких реагуватиме на відповідну цифру. В останньому шарі врахована можливість різного написання цифри «4».

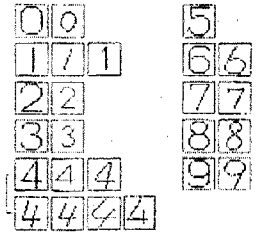


Рис. 17.14. Образи, що навчають, для шару $S4$

17.5. Неокогнітрон з механізмом вибірної уваги

Зазвичай у полі зору людини перебуває багато різних об'єктів, вона ж концентрує свою увагу на одному або декількох з них, за якимись причинами виділеним вибірково.

Такий механізм вибірної уваги може бути реалізований і в одній із версій неокогнітрона. Даний варіант неокогнітрона крім ієрархічної структури, описаної вище, містить ще й локальні зворотні зв'язки наступних сходинок з попередніми, за допомогою яких заглушуються нейрони, що найслабше реагують на відповідний вхідний образ. При цьому на наступні шари передаються сигнали тільки від найбільш активних нейронів. Р. Гехт-Нільсен описує спосіб роботи такого *top-down* зворотного зв'язку, як *pixel channel selector*. Даний механізм дозволяє не тільки автоматично розпізнавати багато об'єктів разом з їхнім розташуванням на зображенні, але й шляхом фільтрації завад відновлювати чіткість зображення.

Контрольні запитання

1. Якою є структура неокогнітрона?
2. Яку функцію виконують S-нейрони? C-нейрони?
3. Яку структуру має S-нейрон?
4. У який спосіб відбувається розпізнавання неспотвореного образу?
5. Яким чином відбувається розпізнавання спотвореного образу?
6. Яку роль у розпізнаванні відіграє непарна операція?
7. Поясніть процес навчання неокогнітрона без учителя.
8. У яких випадках і в який спосіб здійснюється процес навчання неокогнітрона з учителем?
9. Що являє собою механізм вибірної уваги, використовуваний у неокогнітроні?

18. КАСКАДНО-КОРЕЛЯЦІЙНІ МЕРЕЖІ

Каскадно-кореляційні мережі (*ККМ, Cascade Correlation Network*) були введені *С. Фальманом і К. Леб'єром* [116] і є представниками нового класу ШНМ — ШНМ прямого поширення зі змінюваною (зростаючою) архітектурою. Як й інші ШНМ, ККМ містять вхідні, приховані й вихідний шари, причому приховані шари з'являються лише в процесі навчання мережі.

Використовувані в мережі нейрони можуть мати лінійні або нелінійні активаційні функції або ж використати їхню комбінацію. Як алгоритм навчання може використовуватися будь-яке правило (дельта-правило, ВР тощо).

18.1. Архітектура мережі

Початкова архітектура мережі відповідає мережі прямого поширення, що містить тільки два шари — вхідний і вихідний (рис. 18.1, *а*). У процесі навчання в мережі з'являються приховані шари.

18.2. Навчання ККМ

Навчання мережі базується на двох основних принципах:

— покрокове інкрементне введення нейронів у приховані шари;

— навчання знову введених нейронів при збереженні ваг нейронів, які уже перебувають у мережі.

Зміну архітектури ККМ у процесі навчання мережі наведено на рис. 18.1.

На самому початку процесу навчання, як ми вже зазначали, мережа складається тільки з двох шарів: вхідного й вихідного (рис. 18.1, *а*). Навчання нейронів цих шарів відбувається до досягнення

мінімального значення помилки навчання. Якщо це значення буде припустимим, процес навчання завершується. В іншому випадку в прихований шар вводиться нейрон, на вхід якого подаються вихідні сигнали всіх нейронів вхідного й прихованого (якщо такий уже є) шарів. Знову введений нейрон не зв'язується поки з нейронами вихідного шару.

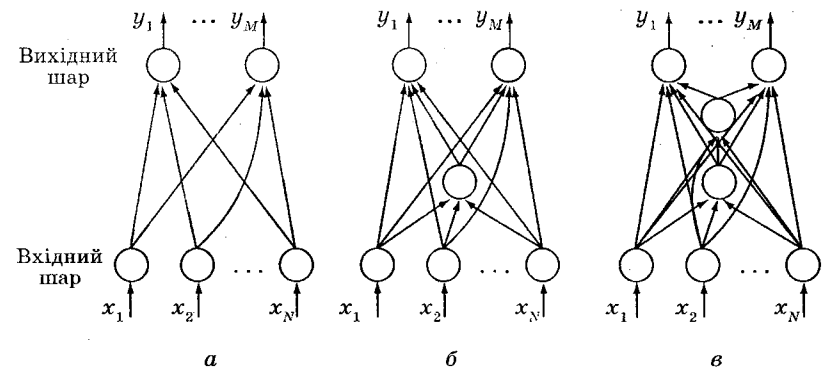


Рис. 18.1. Зміна архітектури ККМ у процесі навчання:
а — початкова архітектура; *б* — введення першого нейрона;
в — введення другого нейрона

Мета навчання полягає в максимізації кореляції між введеним, припустимо, *i*-м, нейроном і вихідною залишковою помилкою e_k

$$s_i = \sum_k \left| \sum_p (z_{pi} - \overline{z}_i)(e_{pk} - \overline{e}_k) \right|, \qquad (18.1)$$

де z_{pi} — вихід знову введеного *i*-го нейрона для *p*-образу; e_{pk} — залишкова помилка вихідного шару, що обчислюється як різниця необхідних і реально обчислених вихідних сигналів; $\overline{z}_i$, $\overline{e}_k$ — відповідно середні значення z_{pi} й e_{pk}; підсумовування здійснюється за всіма виходами (k) і всіма образами (p).

Оскільки в результаті навчання мережі мають бути визначені такі значення ваг w_{ij}, які максимізують (18.1), визначають частинні похідні

$$\frac{\partial s_i}{\partial w_{ij}} = \sum_k \sum_p \sigma_k f_a'(x_{pi}) z_{pj} (e_{pk} - \overline{e}_k), \qquad (18.2)$$

які й використовують в алгоритмі навчання знову введених нейронів.

Тут σ_k — значення кореляції між j-м вихідним нейроном і помилкою на виході k-го нейрона для p-го образу; $f'_a(\cdot)$ — перша похідна використовуваної функції; x_{pi} і z_{pi} — i-та компонента вхідного сигналу мережі і вихід j-го нейрона вхідного шару для p-го образу. Настройка ваг з використанням (18.2) аналогічна дельта-правилу.

Залежно від знака кореляції вихідного сигналу введеного нейрона із сигналом помилки утворюється прискорювальний або гальмуючий зв'язок нейрона з вихідним шаром. Після досягнення максимуму кореляції значення ваг вхідних сигналів введеного нейрона фіксуються і його виходи з'єднуються з усіма нейронами вихідного шару. Відповідні вагові коефіцієнти обчислюються за допомогою обраного алгоритму навчання (рис. 18.1, *б*).

Якщо необхідне значення залишкової помилки вихідного шару не досягнуто, у мережу вводиться наступний нейрон, для якого спочатку визначаються ваги зв'язку як з нейронами вхідного шару, так і вага зв'язку з раніше введеним нейроном прихованого шару. Отже, знову введений нейрон утворює вже другий прихований шар мережі, а його навчання відбувається за аналогією з навчанням раніше введеного нейрона, тобто спочатку визначаються ваги зв'язку з нейронами вхідного шару й раніше введеним нейроном прихованого шару, що забезпечують максимум (18.1), а після фіксації значень цих ваг обчислюються ваги зв'язку з нейронами вихідного шару (рис. 18.1, *в*).

Процес навчання повторюється доти, поки для всіх пар образів не буде досягнута необхідна точність.

Алгоритм навчання мережі може бути записаний у такий спосіб:

1. Вибирається вихідна конфігурація мережі, що являє собою повнозв'язну мережу, що складається з нейронів вхідного й вихідного шарів, початковим значенням ваг яких присвоюються випадкові числа.

2. На входи мережі подаються навчальні пари й за допомогою обраного алгоритму здійснюється настроювання ваг мережі до досягнення мінімального значення вихідної помилки. Якщо це значення є припустимим, процес навчання завершується, якщо ж ні — переходять до кроку 3.

3. Вводиться нейрон прихованого шару, вхідними сигналами якого є вихідні сигнали всіх нейронів вхідного шару. Відповідним ваговим коефіцієнтам присвоюються малі випадкові значення.

4. Обчислюються й фіксуються значення вагових коефіцієнтів введеного нейрона, що максимізують кореляцію (18.1) для всієї множини навчальних пар.

5. Виходи нейрона прихованого шару з'єднуються із входами всіх нейронів вихідного шару й визначаються значення коефіцієнтів, що мінімізують вихідну помилку для всіх пар образів. Якщо величина помилки є припустимою, процес навчання завершується, якщо ні — утворюється новий прихований шар шляхом введення нового нейрона, входи якого з'єднуються з виходами всіх нейронів вхідного шару й раніше введених нейронів вихідного шару (вихідних шарів).

6. Кроки 4 й 5 повторюються до досягнення необхідної точності (припустимої помилки).

Слід зазначити, що описаний алгоритм припускає й інші реалізації. Зокрема, максимізацію кореляції (18.1) можна здійснювати, вводячи не один новий нейрон прихованого шару, а декілька, використовуючи як початкові значення вагових коефіцієнтів різні випадкові значення. А оскільки ці нейрони між собою не пов'язані і їхнє навчання здійснюється на одних образах, можлива одночасна (паралельна) корекція їхніх вагових коефіцієнтів, що значно прискорює процес навчання.

Крім того, замість критерію максимуму кореляції для настроювання ваг можна використати критерій мінімуму середньоквадратичної помилки.

18.3. Основні переваги ККМ

Мережі цього типу мають цілу низку переваг порівняно з іншими ШНМ. Серед основних переваг ККМ слід зазначити такі:

— відсутня необхідність апріорного вибору архітектури мережі; починаючи з найпростішої, що складається тільки із вхідного й вихідного шарів шляхом поступового ускладнення, мережа сама визначає свою архітектуру, що відповідає обраному критерію;

— внаслідок того, що нейрони одного шару не пов'язані, конкуренції між ними немає й процес навчання завершується швидше, ніж в інших мережах;

— навчання при поданні нових образів може здійснюватися без перенавчання всієї мережі шляхом корекції лише вагових коефіцієнтів зв'язків нейронів вихідного й прихованого шарів;

— процес навчання мережі значно простіший щодо обчислення, оскільки під час навчання корегуються ваги тільки одного шару й сигнали в мережі прямують тільки в одному напрямку;

— ця мережа дуже зручна для застосування в обчислювальних системах, які використовують паралельні обчислення.

Контрольні запитання

1. Що являє собою архітектура мережі?
2. У який спосіб здійснюється навчання ККМ?
3. Які переваги має ККМ порівняно з іншими ШНМ?

19. МЕРЕЖА З ЕЛЕМЕНТАМИ ЗАТРИМКИ СИГНАЛУ

ШНМ із елементами затримки сигналу (*Time Delay Neural Networks, TDNN*) є одним із варіантів багатошарового персептрона, призначеного для кращої обробки зміщених у часі образів, які подаються на вхідний шар, згасаючи в часі. Типовим прикладом такої ситуації є задача класифікації мовного сигналу. Тому ШНМ цього типу знайшли застосування у задачі розпізнавання мови.

При розпізнаванні фонем використовується мережа, вхідний шар якої має певні чітко задані розміри. Кількість нейронів цього шару визначає величину часового вікна для цих фонем, довжина яких, однак, може бути різною.

Якщо мережі подаються фонеми різної довжини, то величина вхідного шару повинна бути обрана такою, щоб мережа змогла відтворити найдовшу фонему. Для більшості звуків це означає, що тільки частина даних, пропонованих мережі, дійсно містить мовну інформацію. Решта — це мовні паузи, шум мікрофона й інші завади. При цьому корисна мовна інформація подається мережі часто згасаючою в часі, тобто після паузи вона може з'являтися впродовж свого існування в різних піддіапазонах вхідного шару. Крім того, тривалість окремих фонем може значно коливатися, що також призводить до появи інформації в різних піддіапазонах вхідного шару.

Якщо мережі подається фонема певної тривалості, а наступна за нею фонема внаслідок затяжної початкової мовної паузи затримується, тобто зміщується назад, то мережа має однаково добре обробляти обидві ці фонеми. Тому має сенс зробити ШНМ *робастною* до зовнішнього зміщення поданих образів.

Саме мережа даного типу, будучи робастною, здатна успішно розв'язувати такі задачі [117–122].

19.1. Структура мережі

Ця мережа є багатошаровою мережею прямого поширення, складеною з нейронів, що мають структуру, наведену на рис. 19.1.

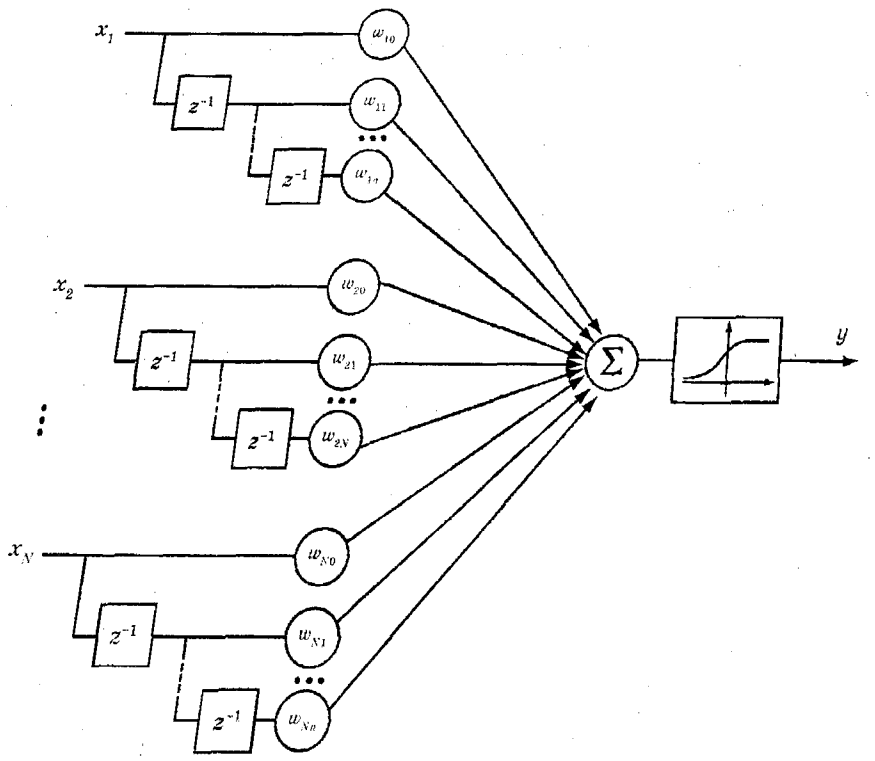


Рис. 19.1. Структура нейрона

На вхід нейрона надходить вхідний вектор $x = (x_1, x_2, ..., x_N)^T$. *У самому нейроні створюються його часові копії за допомогою елементів (ліній) затримки* z^{-1}, ..., z^{-n} і відбувається підсумовування *зваженого сигналу і його копій.*

Нейрон, на вхід якого надходить N-вимірний сигнал і який використовує затримку на n тактів, містить N (n + 1) ваг. Компоненти вхідного сигналу $x_1, x_2, ..., x_N$ *є часовою послідовністю образів мовного сигналу).*

У багатошарових мережах можуть використовуватися елементи затримки між прихованими шарами так, як це зображено на рис. 19.2 на прикладі двох шарів для випадку n = 2.

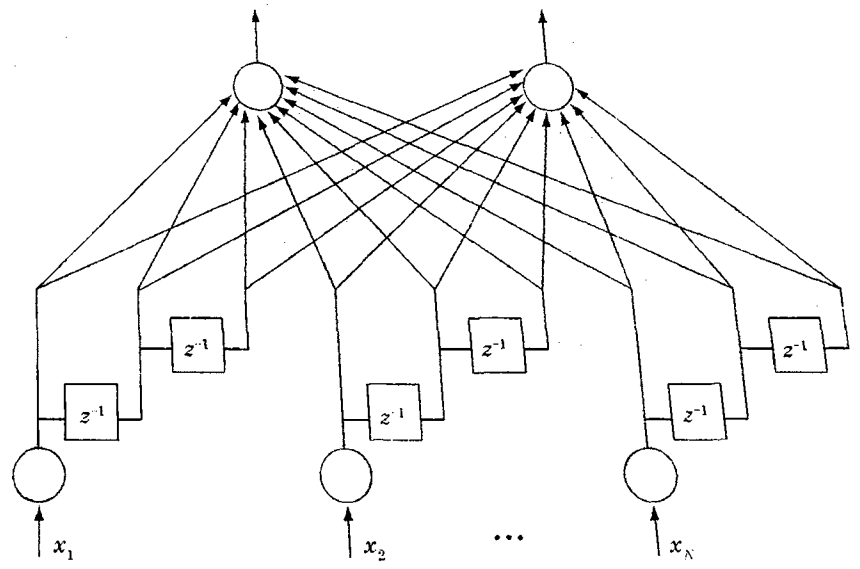


Рис.19.2. Приклад фрагмента мережі з елементами затримки у прихованих шарах

На нейрони другого шару подаються не тільки самі образи, подані вектором $x = (x_1, ..., x_N)^T$, але й їх копії. Цей принцип використовується для всіх шарів багатошарової мережі, завдяки чому досягається робастність мережі до часових спотворювань (подовження або скорочування) вхідних образів. Слід, однак, зазначити, що мережа при цьому ускладнюється, наприклад, кількість вагових коефіцієнтів стає в n разів більше, ніж у багатошарового персептрона.

Звичайно, можна було б з'єднати кожен нейрон першого шару з усіма нейронами вхідного шару. Проте це призвело б до різкого збільшення числа елементів вагових матриць, які необхідно настроювати в процесі навчання.

У мережі цього типу, приклад структури якої наведено на рис. 19.3, нейрони прихованих шарів пов'язані з обмеженою кількістю нейронів попереднього шару, які утворюють їх так звані *рецептивні поля.*

Розмір рецептивного поля r деякого прихованого шару визначає кількість нейронів (вікно) наступного прихованого шару. Так для мережі, зображеної на рис. 19.3, вхідний шар якої утворює матрицю розміру $N \times I$ (I — часове вікно), кількість нейронів у стовпці матриці першого прихованого шару дорівнює

$$L \le I - r^{(1)} + 1 = I - 1,$$

другого прихованого шару —

$$Q \le L - r^{(2)} + 1 \le I - 3 \text{ і т. д.}$$

Тут $r^{(i)}$ — розмір рецептивного поля i-го прихованого шару.

Оскільки шари мережі складаються з матриць нейронів, то місцерозташування нейрона в матриці визначається двома індексами. Тому позначимо вагу зв'язку (n, i)-нейрона ($n = \overline{1,N}$, $i = \overline{1,L}$) вхідного шару з (j, l)-нейроном ($j = \overline{1,J}$, $l = \overline{1,L}$) першого прихованого шару як $w_{n,i,j,l}$. Крім того, позначимо $w_{n,i+r^{(1)},j,l}$ — ваги зв'язку (j, l)-нейрона прихованого шару з нейронами вхідного шару, що перебувають у його рецептивному полі розміру $r^{(1)}$. Для інших же ваг $w_{n,i,j,l}$ виконується умова $w_{n,i,j,l} = w_{n,i+q,j,l+q}$, де $q = \overline{1,r-1}$ [117].

Якби в мережі не було рецептивних полів, то вхідний сигнал (j, l)-нейрона прихованого шару відображав би зважену суму вихідних сигналів всіх нейронів попереднього шару, тобто з використанням позначень, прийнятих у розділі 3, можна було б записати

$$z_{j,l} = \sum_{n=1}^{N} \sum_{i=1}^{I} w_{n,i,j,l} f_{n,i}, \qquad (19.1)$$

де $f_{n,i}$ — активаційна функція (n, i)-нейрона.

За наявності рецептивних полів розміру r формула (19.1) змінюється у такий спосіб:

$$z_{j,l} = \sum_{n=1}^{N} \sum_{i=1}^{I-r+1} w_{n,i,j,l} f_{n,i}. \qquad (19.2)$$

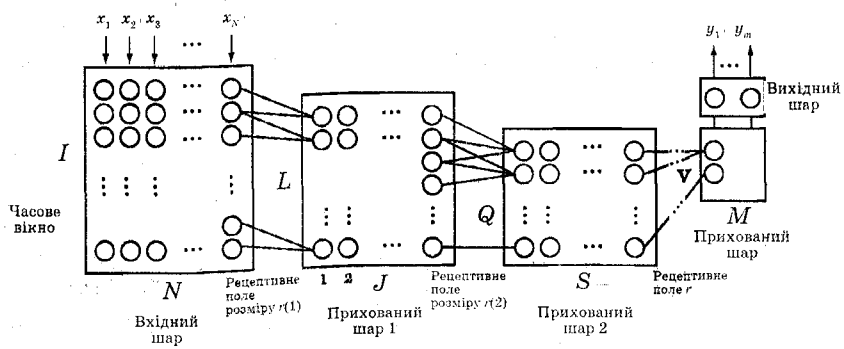


Рис. 19.3. Структура мережі

Як активаційні функції для нейронів прихованих шарів звичайно використовується сигмоїдальна (логістична), а для нейронів вихідного шару — лінійна функція. Однак, як показали дослідження [122], кращих результатів можна досягти, якщо замість простого підсумовування своїх вхідних сигналів нейрони вихідного шару обчислюватимуть величини (*див.* рис. 19.3)

$$y_m = \sum_{v=1}^{V} f_{m,v}^2, \; m = \overline{1,M}. \qquad (19.3)$$

19.2. Навчання мережі

Навчання мережі здійснюється за допомогою алгоритму зворотного поширення помилки, причому якщо в ранніх варіантах цих мереж використовувалося навчання в режимі *offline*, коли настроювання вагових коефіцієнтів виконувалося на основі одноразового подання мережі всіх образів, що навчають, то в більш пізніх уже реалізовувався режим *online*, при якому відбувається корекція ваг після подання кожного образу, що навчає. Оскільки обидва методи виявилися нестійкими, у роботі [120] запропоновано використати їхню комбінацію, яка полягає в тому, що настроювання *offline* здійснюється на основі трьох образів, що навчають, їх кількість поступово підвищується до P, а корекція ваг при переході від однієї групи образів — у режимі *online*.

19.2.1. Алгоритм зворотного поширення помилки

Відповідно до даного алгоритму необхідна зміна ваг при поданні мережі p-го образу обчислюється за формулою, аналогічною (3.49),

$$\Delta_p w_{n,i,j,l} = -\gamma \frac{\partial e_p^2}{\partial w_{n,i,j,l}}, \quad p = \overline{1,P}, \qquad (19.4)$$

де $e_p = \sum_{m=1}^{M} (y_{p,m,l}^{*} - y_{p,m,l})$ — сумарна вихідна помилка мережі при поданні p-го образу; γ — коефіцієнт, що впливає на швидкість навчання.

За аналогією зі звичайним алгоритмом зворотного поширення можна записати

$$\frac{\partial e_p^2}{\partial w_{n,i,j,l}} = \frac{\partial e_p^2}{\partial f_{p,jl}} \frac{\partial f_{p,jl}}{\partial z_{p,jl}} \frac{\partial z_{p,jl}}{\partial w_{n,i,j,l}} = \frac{\partial e_p^2}{\partial f_{p,jl}} f'_{p,jl} f_{p,ni}, \qquad (19.5)$$

де $f'_{p,jl}$ — похідна активаційної функції (j, l)-нейрона (індекс p відповідає p-му образу).

Підстановка (19.5) у (19.4) дає

$$\Delta_p w_{n,i,j,l} = -\gamma \frac{\partial e_p^2}{\partial f_{p,jl}} f'_{p,jl} f_{p,ni}. \qquad (19.6)$$

Як й у звичайному алгоритмі зворотного поширення, корекція ваг зв'язку нейрона прихованого шару (n, j) з нейроном (s, q) $(s = \overline{1,S};\ q = \overline{1,Q})$ залежить від того, де розташований нейрон (s, q): у наступному прихованому чи у вихідному шарі. Якщо нейрон (s, q) перебуває у прихованому шарі, то наявність у всіх нейронів цього шару рецептивних полів розміру $r^{(2)}$ призводить до того, що, як видно на рис. 19.3, нейрон (j, l) попереднього прихованого шару буде пов'язаний з тими нейронами (s, q) даного шару, для яких $l - r^{(2)} + 1 \le q \le l$. Внаслідок цього для будь-якого прихованого нейрона з рецептивним полем розміру r отримуємо

$$\frac{\partial \varepsilon_p^2}{\partial f_{p,jl}} = \sum_{s=1}^{S} \sum_{q=L-r+1}^{L} \left(\frac{\partial \varepsilon_p^2}{\partial z_{p,sq}} \frac{\partial z_{p,sq}}{\partial p_{p,jl}} \right) = \sum_{s=1}^{S} \sum_{q=L-r+1}^{L} \delta_{p,jl} w_{j,l,s,q}, \qquad (19.7)$$

де $\delta_{p,jl} = \dfrac{\partial \varepsilon_p^2}{\partial z_{p,sq}}$.

Якщо ж нейрон (s, q) знаходиться у вихідному шарі (у цьому випадку, як легко переконатися з рис. 19.3, кількість нейронів вихідного шару дорівнюватиме J), то з урахуванням виду його активаційної функції (19.3) можна записати

$$e_p^2 = \sum_{j=1}^{J} (y_{p,j}^* - y_{p,j})^2 = \sum_{j=1}^{J} (y_{p,j}^* - \sum_{l=1}^{L} f_{p,jl}^2)^2, \qquad (19.8)$$

звідки

$$\frac{\partial e_p^2}{\partial f_{p,jl}} = 2 f_{p,jl} (\sum_{l=1}^{L} f_{p,jl} - y_{p,j}^*). \qquad (19.9)$$

Отже, в остаточному вигляді алгоритм корекції ваг може бути записаний у такий спосіб:

$$\Delta_p w_{n,i,j,l} = -\gamma f'_{p,jl} f_{p,ni} \frac{\partial e_p^2}{\partial f_{p,jl}}, \qquad (19.10)$$

де

$$\frac{\partial e_p^2}{\partial f_{p,jl}} = \begin{cases} \sum_{s=1}^{S} \sum_{q=L-r+1}^{L} \delta_{p,jl} w_{j,l,s,q}, & \text{якщо нейрон } l \text{ знаходиться} \\ & \text{у прихованому шарі;} \\ 2 f_{p,jl} (\sum_{l=1}^{L} f_{p,jl} - y_{p,j}^*), & \text{якщо нейрон } l \text{ знаходиться} \\ & \text{у вихідному шарі.} \end{cases} \qquad (19.11)$$

Згадувана вище умова рівності «паралельних» ваг, розташованих в одному стовпці нейронів (рис. 19.3) $w_{n,i,j,l} = w_{n,i+q,j,l+q}$ $(q = \overline{1, r-1})$, досягається шляхом додавання до всіх значень вагових коефіцієнтів $w_{n,i,j,l}$ просумованих необхідних корекцій ваг одного стовпця $\Delta w_{n,i,j} = \sum_{l=1}^{L} \Delta w_{n,i,j,l}$.

19.2.2. Прискорення процесу навчання

Як зазначалося вище, подання на нейрони другого шару не тільки самих образів, але і їхніх копій призводить до ускладнення мережі. Якщо для простого розпізнавання окремих фонем достатньо мережі невеликої розмірності, для якої таке ускладнення виявляється несуттєвим, то для мережі великої розмірності це стає відчутним, тому що використання для її навчання алгоритму зворотного поширення помилки призводить до збільшення тривалості (затягування) процесу навчання.

Тому питання прискорення процесу навчання є дуже важливим. Оскільки алгоритм зворотного поширення реалізує градієнтний метод мінімізації деякого заздалегідь обраного функціонала й містить, як випливає з (19.10), ряд призначуваних за якимись міркуваннями параметрів, існує кілька можливостей скорочення процесу навчання.

Перша можливість — відповідний вибір функціонала, що мінімізується. Можна замість квадратичного e_p^2 взяти, наприклад, функціонал вигляду

$$e_p = \sum_{m=1}^{M} \ln(1 - (y_{p,m}^* - y_{p,m})^2), \qquad (19.12)$$

мінімізація якого, як показують дослідження, призводить до задовільних результатів. Однак слід зважати на те, що під час застосування (19.12) формули (19.10), (19.11) зміняться.

Друга, найбільш очевидна, можливість прискорення збіжності алгоритму (19.4) полягає у виборі оптимального значення коефіцієнта навчання γ. Як уже неодноразово раніше зазначалося, вибір малих значень γ, обумовлюючи незначну зміну $\Delta w_{n,i,j,l}$, призводить до затягування процесу навчання й застрягання алгоритму в локальному мінімумі; великі ж значення γ, викликаючи значну зміну $\Delta w_{n,i,j,l}$, можуть призвести до проскакування мінімуму й осциляції процесу настроювання. У роботі [120] запропоновано використати для кожного нейрона (j, l) свій коефіцієнт γ_{jl}, що залежить від помилок усіх нейронів, які відносяться до рецептивного поля цього нейрона, і визначений у такий спосіб:

$$\gamma_{jl} = \frac{\gamma}{1 + \frac{\gamma}{\omega}\sqrt{\sum_{n=1}^{N}\sum_{i=1}^{I+r-1}\left(\frac{\partial e_p}{\partial w_{n,i,j,l}}\right)^2}}, \qquad (19.13)$$

де $\gamma > 0$, $\omega > 0$ — деякі задані параметри (наприклад, у зазначеній роботі задавалося $\omega = 1$).

Нарешті, *третя* можливість полягає у виборі відповідних активаційних функцій нейронів. Хоча звичайно вони вибираються логістичними, на практиці найчастіше застосовують або їхні деякі модифікації, або модифікації використовуваних в алгоритмах похідних цих функцій. Зокрема, замість використання логістичних функцій, що прямують до нуля при збільшенні аргументу до $-\infty$ (це означає, що не буде зміни ваг при такому збільшенні вхідних сигналів), застосовують їхнє зміщення, наприклад $\hat{f}(x) = f(x) - 0{,}5$. При цьому, природно, необхідно привести й значення вхідних сигналів до інтервалу $[-0{,}5; 0{,}5]$.

Більш ефективною є модифікація активаційної функції вигляду

$$\hat{f}(x) = f(x) + \alpha x \qquad (19.14)$$

з $\alpha > 0$, для якої похідна $\hat{f}'(x) = f'(x) + \alpha$ завжди відмінна від нуля, що забезпечує постійну корекцію параметрів. Звичайно вибирають $\alpha \in [0{,}01; 0{,}1]$.

Застосування розглянутих методів прискорення процесу навчання або кожного окремо, або їхньої комбінації дають істотний виграш у часі.

19.3. Ієрархічні мережі з часовою затримкою сигналу

Структура мережі із затримкою сигналу істотно ускладнюється, якщо ця мережа призначається для розпізнавання мови, а не окремих фонем. У цьому випадку використовують мережу з ієрархічною структурою, при якій приховані шари розбиваються на окремі модулі, що навчаються незалежно один від одного.

Приклад такої мережі, призначеної для розпізнавання фонем, наведено на рис. 19.4.

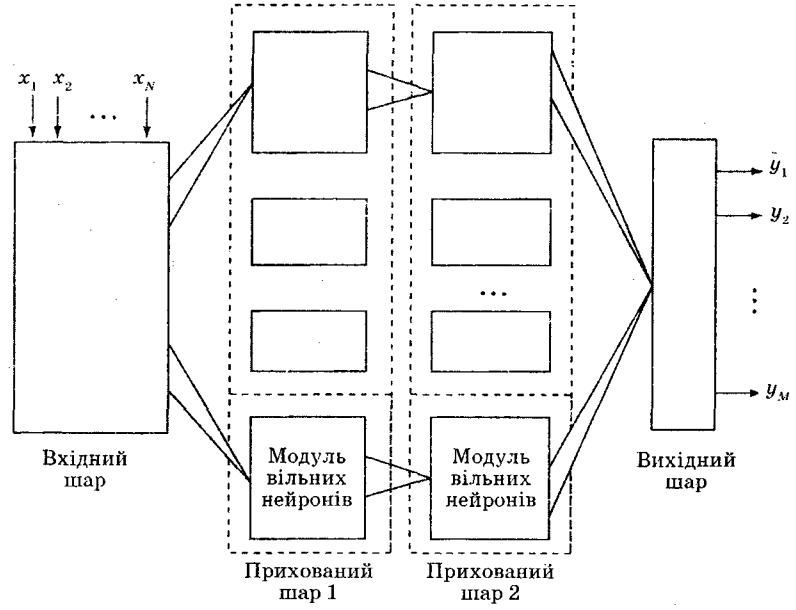


Рис. 19.4. Ієрархічна мережа з елементами часової затримки сигналу

Результати численних досліджень свідчать про те, що для ефективного розпізнавання фонем достатньо мати мережу із двома прихованими шарами.

Зображена на рисунку мережа також містить два прихованих шари, що містять крім основних модулів, кожний з яких настроюється на певний образ, модулі вільних нейронів. Навчання мережі

відбувається в три етапи. На *першому* етапі відповідним образам (фонемам) незалежно один від одного навчаються тільки модулі прихованих шарів. На *другому* — з використанням усіх образів, що навчають, настроюються тільки вагові коефіцієнти зв'язків другого прихованого шару з вихідним. На *третьому* — мережі подаються образи, що навчають, і корегуються всі її вагові коефіцієнти.

Введення додаткових груп вільних нейронів, що відіграють сполучувальну роль (*connectionist glue*) між окремими групами нейронів прихованих шарів, має на меті поліпшення розпізнавальних властивостей мережі. Так, введення вільних нейронів у перший прихований шар мережі, що містить два прихованих шари й призначена для розпізнавання приголосних *b-d-g-p-t-k* (одна група нейронів навчалася приголосним *b-d-g*, а друга — *p-t-k*), дозволило підвищити відсоток правильного розпізнавання з 96,1 до 98,6 [119].

Більш складними є структури мереж, призначені для розпізнавання окремих слів і мови (*Multi-State TDNN*). Вони мають ієрархічну структуру й містять крім двох звичайних прихованих шарів ще один прихований, що здійснює динамічне перетворення образів (*dynamical time warping*) і складається з нейронів із сигмоїдальними або навіть лінійними функціями активації [121]. Нейрони цього шару утворюють два класи: перший відповідає стійкому стану (*stable state units*), тобто стійкому звучанню слова, другий — перехідному стану (*transition state units*), тобто коартикуляції різних фонем. При цьому дуже важливою є проблема вибору оптимальної структури мережі, зокрема проблема визначення кількості необхідних нейронів стану.

Навчання цих мереж здійснюється за допомогою розглянутого вище алгоритму зворотного поширення.

Контрольні запитання

1. Якою є структура мережі з елементами затримки сигналу?
2. Що являє собою рецептивне поле мережі?
3. У який спосіб визначається необхідна кількість нейронів рецептивних полів?
4. Яким чином здійснюється навчання мережі?
5. Які існують підходи до прискорення процесу навчання мережі?
6. Чим відрізняється навчання в режимі *offline* від навчання в режимі *online*?
7. Що являє собою ієрархічна мережа з часовою затримкою сигналу?
8. Назвіть сфери найбільш ефективного застосування мереж цього типу.

20. МЕРЕЖІ НА ОСНОВІ ТЕОРІЇ АДАПТИВНОГО РЕЗОНАНСУ

Мозок людини має властивість запам'ятовувати нову інформацію без руйнування старої, що вже перебуває в пам'яті. Бажано, щоб таку ж властивість мала й ШНМ. У цьому випадку ШНМ, по-перше, могла б адаптивно навчатися при надходженні нової інформації, зберігаючи стабільність, яка гарантує, що вже наявна корисна інформація не зруйнується, і, по-друге, була б досить пластичною для визначення того, чи є нова інформація суттєвою або несуттєвою. Це створює дилему, названу *С. Гроссберґом дилемою стабільності – пластичності* й розв'язувану запропонованою ним мережею.

Теорія адаптивного резонансу (*ART*) розроблена для моделювання самоорганізувальної ШНМ розпізнавання образів, що має високий ступінь паралелізму архітектури й заснована на біологічних й поведінкових даних. Двошарова архітектура мережі, запропонована *Гроссберґом* і *Карпентером* [123] така, що забезпечується можливість її роботи у двох режимах: *навчання*, коли внутрішні параметри мережі можуть змінюватися, і *розпізнавання*, коли при незмінних параметрах мережі відбувається класифікація вхідних образів. Таким чином, ART-мережа є векторним класифікатором, що відносить вхідний вектор до одного з раніше запам'ятованих образів. Якщо вхідний вектор подібний у значенні даного критерію до одного з раніше запам'ятованих, то останній змінюватиметься (навчатиметься), щоб стати більш подібним на вектор, що знову надійшов. Якщо ж новий вектор виявиться недостатньо подібним на запам'ятовуючий, то останній не змінюватиметься, а мережа створить новий образ, ідентичний вектору, що знову надійшов. У такий спосіб вирішується дилема стабільності – пластичності.

Основні властивості мереж ART відображені в ряді теорем, доведених у [123]. Найбільш важливими з них є такі:

1. *Після стабілізації процесу навчання подання одного з векторів, що навчають* (*або вектора з істотними характеристиками*

образу), *активуватиме необхідний нейрон шару розпізнавання без пошуку*. Ця характеристика «прямого доступу» визначає швидкий доступ до попередньо вивчених образів.

2. *Процес пошуку є стійким*. Після визначення нейрона-переможця в мережі не буде збуджень інших нейронів у результаті зміни векторів виходу шару порівняння; тільки сигнал скидання може викликати такі зміни.

3. *Процес навчання є стійким*. Навчання не викликатиме перемикання з одного активованого нейрона шару розпізнавання на інший.

4. *Процес навчання кінцевий*. Будь-яка послідовність довільних вхідних векторів формуватиме стабільний набір ваг після кінцевої кількості навчальних серій; послідовності, що повторюються, навчальних векторів не приводитимуть до циклічної зміни ваг.

Залежно від виду вхідних змінних і способу їх обробки розрізняють:

- ART-1, що використовує двійкові вхідні вектори;
- ART-2, ART-2a, що використовує як двійкові, так і неперервні вектори входів;
- ART-3, використовувана для моделювання біологічних процесів;
- ART MAP, що являє собою комбінацію двох ART-мереж;
- Fuzzy-ART — це ART-1, що використовує нечітку логіку.

20.1. МЕРЕЖА ART-1

20.1.1. Архітектура й призначення основних модулів

Спрощену архітектуру мережі ART-1 наведено на рис. 20.1. Мережа складається з таких компонентів:

- шар порівняння F_1 (*comparison layer*);
- шар розпізнавання F_2 (*recognition layer*);
- приймачі 1 і 2 (*gain*);
- модулі вагових матриць; неперервних ваг (*bottom-up-matrix*); двійкових ваг (*top-down-matrix*);
- модуль скидання (*reset*).

Характерною рисою мережі ART-1 є те, що безупинно змінюваний вхідний вектор $\boldsymbol{x}$ передається в прямому й зворотному напрямках (резонує) між шарами мережі.

Свою назву ART-мережа отримала внаслідок того, що навчання в ній відбувається тільки під час виникнення резонансу між F_1 й F_2 (адаптивне).

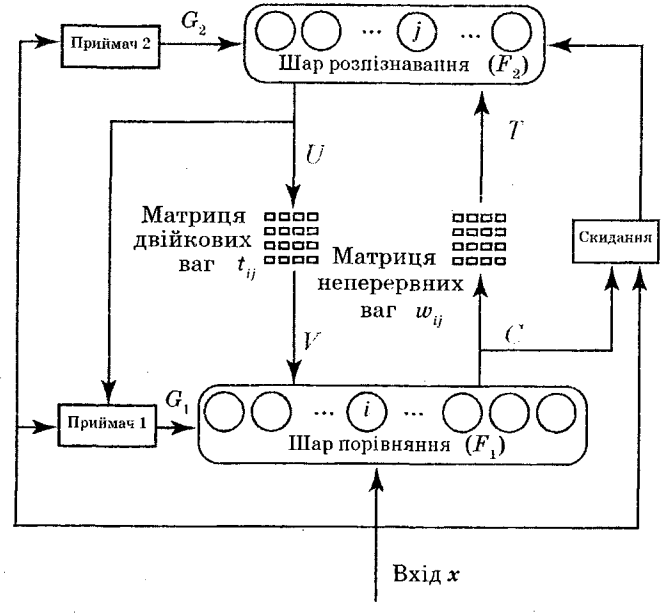


Рис. 20.1. Спрощена мережа ART-1

Послідовність перетворення двійкового вхідного сигналу $\boldsymbol{x}$, що надходить на шар порівняння, у сигнал $\boldsymbol{u}$, який повертається на цей же шар із шару розпізнавання, може бути подана у такий спосіб:

$$X \to C \to T \to U \to V. \qquad (20.1)$$

Шар порівняння (F_1) порівнює вихідний сигнал шару розпізнавання із вхідним сигналом x, що надійшов. Крім того, він спочатку пропускає незмінним сигнал $\boldsymbol{x}$, формуючи з нього вихідний сигнал $\boldsymbol{C}$, що подається на матрицю неперервних вагових коефіцієнтів w_{ij}, і скидання.

На кожен нейрон даного шару надходить три двійкових входи: G_1 — вихід приймача 1, x_i — i-та компонента (i-й біт) вектора входів $\boldsymbol{x}$, V — зважена сума виходів шару розпізнавання. Спочатку сигнал G_1 установлюється в одиницю, забезпечуючи тим самим один

із необхідних для активації нейронів даного шару, а вектор V приймається нульовим. Під час процесу навчання компоненти вектора S формуються за правилом *«два із трьох»*:

$$G_i = \begin{cases} 1, & \text{якщо } x_i v_i \vee x_i G_1 \vee v_i G_1 = 1; \\ 0, & \text{в іншому випадку.} \end{cases} \qquad (20.2)$$

Отже, щоб i-та компонента вектора c мала одиничне значення (вихід i-го нейрона цього шару дорівнював одиниці) необхідно, щоб мінімум два із трьох компонентів G_1, x_i, V_i дорівнювали одиниці.

Приклад 20.1.

Нехай $x = 101101100111$

$V = 101011010101.$

Тоді при $G_1 = 0 \quad C = 101001000101$

$G_1 = 1 \quad C = 101111110111.$

Динаміка (активність) i-го нейрона цього шару описується таким диференціальним рівнянням:

$$\gamma \frac{dC_i}{dt} = -C_i + (1 - \alpha_1 C_1) \cdot I_i^+ - (\beta_1 + \delta_1 C_i) \cdot I_i^-, \qquad (20.3)$$

де $I_i^+ = x_i + v_i$, $I_i^- = \sum_k f(u_k)$ — відповідно узагальнені підсилюючі й гальмуючі сигнали, що надходять на на i-й нейрон: γ, α_1, β_1, δ_1 — деякі невід'ємні параметри.

Шар розпізнавання (F_2) класифікує вхідні вектори. Кожен нейрон цього шару має свій вектор вагових коефіцієнтів з компонентами, що є дійсними числами. Цей вектор і порівнюється з вектором вхідного сигналу. Отже, кожен нейрон обчислює згортку своїх власних ваг і вхідного вектора c, тобто формує сигнал $T_k = \sum_i w_{ik} C_i$.

Однак активується тільки той нейрон (сигнал на його виході дорівнює одиниці), вагові коефіцієнти якого найбільше відповідають вхідному сигналу, що здійснюється відповідно до правила

$$T_j = \max_k \{T_k\}; \qquad (20.4)$$

$$u_i = f(T_j) = \begin{cases} 1, & \text{якщо } i = j; \\ 0 & \text{в іншому випадку.} \end{cases} \qquad (20.5)$$

Слід зазначити, що нейрони цього шару з'єднані в латерально-гальмівну мережу, коли вихід кожного нейрона подається на входи всіх інших нейронів з негативними вагами, а на свій власний — з позитивною вагою. Це забезпечує з одного боку посилення нейрона, що має найбільше значення виходу, а з іншого — гальмування інших нейронів у шарі.

Динаміка j-го нейрона цього шару описується рівнянням, аналогічним (20.3):

$$\gamma \frac{du_j}{dt} = -u_j + (1 - \alpha_2 u_j) I_j^+ - (\beta_2 + \delta_2 u_j) I_j^-, \qquad (20.6)$$

де $I_j^+ = g(u_j) + T_j$ — сигнал власного додатного зворотного зв'язку j-го нейрона; $I_j^- = \sum_{k \neq j} g(u_k)$ — сума сигналів від'ємних зворотних зв'язків, що надходять зі всіх інших нейронів шару F_2; γ, α_2, β_2, δ_2 — деякі невід'ємні параметри, обирані таким чином, щоб активувався тільки один нейрон, на вхід якого надходить найбільший сигнал T_j (j-й нейрон є переможцем).

Приймач 2 реалізує логічну функцію «або» булевих компонент вхідного вектора x, тобто $G_2 = x_1 \vee x_2 \vee \ldots \vee x_n$.

Приймач 1 (*Attentional gain control*, *AGC*) реалізує механізм зіставлення вхідного сигналу із сигналом, що надходить із шару розпізнавання, формуючи сигнал G_1 за правилом $G_1 = x_1 \overline{U}$. Таблиця істинності сигналу G_1 має вигляд:

Таблиця 20.1

Компонента		G_1
x_1	U	
0	0	0
0	1	0
1	0	1
1	1	0

Модуль вагових матриць ART-1 містить:

— *матрицю неперервних ваг* w_{ij} (*bottom-up-matrix, BUM*), використовувану для визначення подібності вхідного вектора наявному образу у *фазі розпізнавання*;

— *матрицю двійкових ваг* t_{ji} (*top-down-matrix*), що застосовується для визначення коректності віднесення подання вхідного вектора наявному образу у *фазі порівняння*.

Внаслідок того, що вагові коефіцієнти, що складають ці матриці, досить повільно змінюються (корегуються) при поданні мережі різних вхідних образів, ці модулі називають також *довгостроковою* пам'яттю (*LTM — Long Term Memory*) ART.

Модуль скидання (*STM — Reset, Short Term Memory*) виробляє сигнал скидання активованого нейрона в шарі розпізнавання в тому випадку, якщо вхідний вектор $\boldsymbol{x}$, що надійшов, значно відрізняється (щодо обраного критерію подібності) від наявного $\boldsymbol{c}$. Це здійснюється шляхом підрахунку й порівняння кількості одиниць у векторах $\boldsymbol{x}$ та $\boldsymbol{c}$. Цей модуль називають також *короткочасною пам'яттю* (*STM*) ART.

20.1.2. Функціонування мережі

У процесі функціонування мережі розрізнюють такі фази: ініціалізація, розпізнавання, порівняння, пошук, навчання.

Ініціалізація. Перед початком процесу навчання мережі встановлюється значення елементів обох вагових матриць, а також параметр подібності $\boldsymbol{p}$.

Якщо позначити нейрони шару порівняння й шару розпізнавання відповідно n_i $(i=\overline{1,N})$ и n_j $(j=\overline{1,M})$, згідно з [123] значенням w_{ij} елементів матриці неперервних ваг (*BUM*) вибирають такими, що задовольняють умові

$$0 < w_{ij}(0) < \frac{\lambda}{\lambda - 1 + N} \text{ для всіх } i, j, \qquad (20.7)$$

де $\lambda \in (1, 2]$. Вибір більших значень λ призводить до того, що мережа може розподілити всі нейрони розпізнавального шару одному вхідному вектору $\boldsymbol{x}$.

Початкові значення матриці двійкових ваг (*TDM*) вибирають такими, що задовольняють умові

$$\frac{\beta_1 - 1}{d} < t_{ji}(0) \le 1, \qquad (20.8)$$

де β_1 — стала, визначена з (20.3); $d > 0$ — стала, на яку помножуються виходи активованих нейронів шару F_2.

У [124] показано, що вибір занадто малих значень $t_{ji}(0)$ призводить до відсутності як відповідності в шарі порівняння, так і навчання. Звичайно приймається $t_{ji}(0) = 1$.

Параметр подібності $\rho \in (0, 1]$ установлюється залежно від необхідного ступеня відповідності вхідного сигналу наявному образу. При $\rho \to 1$ мережа віднесе до одного класу практично нерозрізнені образи, при $\rho \to 0$ — образи, що мають малу подібність.

Розпізнавання. Оскільки в початковий момент часу вихідний сигнал $U(V)$ шару розпізнавання відсутній, а сигнал $G_2 = 1$ забезпечує активацію одного з нейронів шару порівняння, поява вхідного сигналу $\boldsymbol{x}$ ініціалізує фазу розпізнавання. На виході шару порівняння з'являється вектор $\boldsymbol{c}$, компоненти якого, визначені згідно з (20.2) (у цьому випадку $c_i = G_2 x_i$), ідентичні компонентам вектора $\boldsymbol{x}$.

Як зазначалося вище, розпізнавання реалізується обчисленням згортки ваг кожного нейрона шару розпізнавання із сигналом N і визначенням найбільш активного нейрона, *нейрона-переможця* відповідно до (20.4), (20.5). Вихід нейрона-переможця дорівнюватиме одиниці, а виходи всіх інших нейронів будуть нульовими. З (20.3) випливає, що тільки j-та компонента вектора $\boldsymbol{u}$, перетворена далі двійковою ваговою матрицею t_{ji}, буде відмінна від нуля.

Порівняння. На цій фазі з'являється одиничний вихідний сигнал шару розпізнавання U, що установлює β_1 у нуль (*див.* табл. 20.1) і перетворюється за допомогою матриці двійкових ваг t_{ji} у сигнал V, прямуючи на вхід шару порівняння.

Причому, якщо найбільша кореляція виникає між вектором $\boldsymbol{c}$ й j-м образом, що перебуває на шарі розпізнавання, коли $u_j = 1$, для i-го нейрона шару порівняння, отримуємо

$$v_i = u_j t_{ji} = t_{ji}. \qquad (20.9)$$

Відповідно до правила (20.2) будуть активовані тільки ті нейрони, у яких відповідні компоненти векторів $\boldsymbol{x}$ й $\boldsymbol{v}$ дорівнюють одиниці, тобто $x_i = v_i = 1$. Тому активація нейронів у цьому

випадку є нічим іншим, як покомпонентним порівнянням векторів x й v. Розходження значень деяких компонентів x й v призводить до того, що відповідні компоненти вектора c будуть нульовими.

Приклад 20.2.

Нехай $U = 010000$

$$w_{ij} = \begin{vmatrix} 011011 \\ 111101 \\ 110110 \\ 100111 \\ 111111 \\ 010100 \end{vmatrix},$$

тоді $V = V_2 = 111101$.

Блок скидання, порівнюючи x й c, виробляє сигнал скидання, якщо подібність цих векторів нижче заданого порога. Для обчислення подібності використовують співвідношення

$$S = \frac{|t_i \wedge x|}{x}. \qquad (20.10)$$

Тут $|\cdot|$ означає довжину відповідного вектора. А оскільки ці вектори є двійковими, то їх довжина дорівнює кількості даних у них одиниць. S змінюється від 0 (повна невідповідність) до 1 (повна відповідність).

Приклад 20.3.

Нехай $t_j = (101101)$ $x = (100111)^T$.

Тоді

$|t_j \wedge x| = 1 + 0 + 0 + 1 + 0 + 1 = 3;$

$|x| = 1 + 0 + 0 + 1 + 1 + 1 = 4;$

$S = \frac{3}{4}.$

Пошук. Якщо подібність S нейрона-переможця перевищує заданий параметр подібності ρ (*див.* фазу ініціалізації), пошук не потрібний. Якщо ж ця подібність нейрона, що переміг, нижче необхідного, то можливо, що будь-який інший нейрон шару розпізнавання забезпечуватиме більшу подібність із поданим сигналом, незважаючи на те, що згортка вектора його вагових коефіцієнтів із вхідним фактором має менше значення. У цьому випадку запам'ятовані образи можуть бути переглянуті для пошуку образу, найбільш відповідному поданому вхідному вектору.

Для ініціалізації пошуку сигнал скидання гальмує активований нейрон у шарі розпізнавання, сигнал G_1 установлюється в одиницю, й інший нейрон шару розпізнавання виходить переможцем. Його запам'ятований образ також перевіряється на подібність, і процес повторюється доти, поки не буде або знайдений нейрон-переможець шару розпізнавання з подібністю, що перевищує рівень (успішний пошук), або переглянуті й загальмовані всі нейрони шару (невдалий пошук). В останньому випадку жоден наявний образ не відповідає вхідному сигналу, тому вводиться новий незв'язаний нейрон (образ), що згодом буде навчений.

Таким чином, невдалий пошук автоматично завершуватиметься на незв'язаному нейроні, тому що його ваги t_{ji} не змінювалися й дорівнюють початковому значенню, тобто одиниці. Тому відповідно до (20.2) компоненти вектора c будуть ідентичні компонентам нового вектора x.

Навчання в LTM виникає тоді, коли виявляється відповідність між образом, що перебуває в пам'яті, і вхідним сигналом, який знову надійшов, або коли вхідний сигнал, що надійшов, утворить новий образ.

На цій фазі елементи обох матриць w_{ij} й t_{ji} змінюються таким чином, щоб подібні фактори активували відповідні нейрони. Навчання відбувається без учителя.

Рівняння, що описує процес навчання, задовольняє закону Вебера й *правилу асоціативного гальмування* [34], згідно з яким вагові коефіцієнти, що відповідають вектору c з більшою кількістю одиниць, будуть менше вагових коефіцієнтів, які відповідають вектору c з меншою кількістю одиниць. Ця властивість самомасштабування уможливлює поділ двох векторів у випадку, коли один вектор є піднабором іншого, тобто коли набір одиничних компонентів одного вектора складає підмножину одиничних компонентів іншого.

Розрізняють два види навчання [124]: *повільне* й *швидке*. При *повільному* навчанні вхідний вектор $\boldsymbol{x}$ подається короткочасно, так, що ваги мережі не встигають досягти своїх асимптотичних значень за одне подання $\boldsymbol{x}$. При цьому значення ваг визначатимуться швидше статистичними характеристиками векторів $\boldsymbol{x}$, ніж характеристиками будь-якого одного вектора $\boldsymbol{x}$.

При *швидкому* навчанні вхідний вектор подається на тривалий час, що дозволяє вагам досягти асимптотичних значень.

Процес навчання мережі описується відповідними диференціальними рівняннями.

Для матриці ваг $[w_j]$ (*bottom-up-pfade*) це рівняння має вигляд

$$\frac{dw_{ij}}{dt} = \gamma f(T_j)\left[(1-w_{ij})\lambda h(x_i) - w_{ij}\sum_{k\neq i} h(x_k)\right], \qquad (20.11)$$

де $f(T_j) = n_j$ — вихід нейрона n_j, $h(x_i)$ — вихід нейрона n_i; γ, λ — деякі сталі.

Рівняння корекції ваг $[t_{ji}]$ є аналогічним (20.11), однак дещо простіше внаслідок того, що в ньому сталі, еквівалентні γ і λ у (20.11), приймаються рівними одиниці:

$$\frac{dt_{ji}}{dt} = f(T_j)\left[-t_{ji} + h(x_i)\right]. \qquad (20.12)$$

Під час аналізу приведених рівнянь необхідно врахувати таке. Навчання елементів матриці w_{ij} відбувається тоді, коли активовані нейрони n_i й n_j, тобто при $f(T_j) = 1$ й $h(x_i) = 1$. З іншого боку, w_{ij} прямує до нуля, якщо нейрон n_i не активований, а нейрон n_j активований. Нарешті, навчання не відбувається, якщо нейрон n_j не активований. Все це дозволяє записати правило настроювання (20.11) у такий спосіб:

$$\frac{dw_{ij}}{dt} = \begin{cases} \gamma\left[(1-w_{ij})\lambda - w_{ij}(|\boldsymbol{c}|-1)\right], & \text{якщо активовані } n_i \text{ й } n_j; \\ -\gamma|\boldsymbol{c}|w_{ij}, & \text{якщо нейрон } n_i \text{ не активований,} \\ & \text{а } n_j \text{ — активований;} \\ 0, & \text{якщо нейрон } n_i \text{ не активований.} \end{cases} \qquad (20.13)$$

Аналогічно для елементів матриці t_{ji}, отримуємо, що навчання відбувається, якщо одночасно активовані нейрони n_i й n_j, тобто при

$f(T_j) = 1$ й $h(x_i) = 1$, причому ваги експоненціально прямують до одиниці (нейрон n_j намагається вивчити поданий з F_1 образ). Якщо ж нейрон n_i не активований, а нейрон n_j активований, t_{ji} експоненціально прямує до нуля, і нарешті, при неактивованому нейроні n_j навчання не відбувається. Таким чином, правило навчання (20.12) набуває вигляду

$$\frac{dt_{ij}}{dt} = \begin{cases} -t_{ij} + 1, & \text{якщо активовані } n_i \text{ й } n_j; \\ -t_{ij}, & \text{якщо нейрон } n_i \text{ не активований,} \\ & \text{а нейрон } n_j \text{ активований;} \\ 0, & \text{якщо нейрон } n_j \text{ не активований.} \end{cases} \qquad (20.14)$$

Можна сказати, що при швидкому навчанні елементи обох матриць прагнуть до асимптотичних значень.

$$w_{ij} = \frac{\lambda c_i}{\lambda - 1 + \sum_k^n c_k}; \qquad (20.15)$$

$$t_{ji}(t+1) = c_i \wedge w_{ji}(t) \text{ для всіх } i. \qquad (20.16)$$

Тут c_i — i-та компонента вихідного вектора шару порівняння; j — номер нейрона-переможця шару розпізнавання; t_{ji} — вага зв'язку між нейроном-переможцем j шару розпізнавання й нейроном i шару порівняння.

При цьому відповідний ваговий вектор матриці двійкових ваг настроюється так, що він виробляє вектор $\boldsymbol{c}$.

20.1.3. Функціонування ART-1

Порядок роботи мережі наведено на рис. 20.2 [124]. На вхід мережі (шар порівняння F_2) подається вхідний сигнал $\boldsymbol{x}$, образ якого мережа намагається знайти на шарі розпізнавання. Із сигналу $\boldsymbol{x}$ шар порівняння формує STM-образ (*Short Term Memory*) $\boldsymbol{c}$. Одночасно блокується сигнал скидання. Вектор $\boldsymbol{c}$ за допомогою матриці неперервних ваг трансформується у вектор T, що подається у шар порівняння й *активує* деякий образ Y (заштрихований образ у шарі порівняння на рис 20.2, *а*). На виході із розпізнавального шару виробляється вектор $\boldsymbol{u}$ (рис. 20.2, *б*), перетворений за допомогою матриці двійкових ваг у вектор $\boldsymbol{v}$, що подається на шар порівняння. Якщо подібність даного образу до поданого нижче

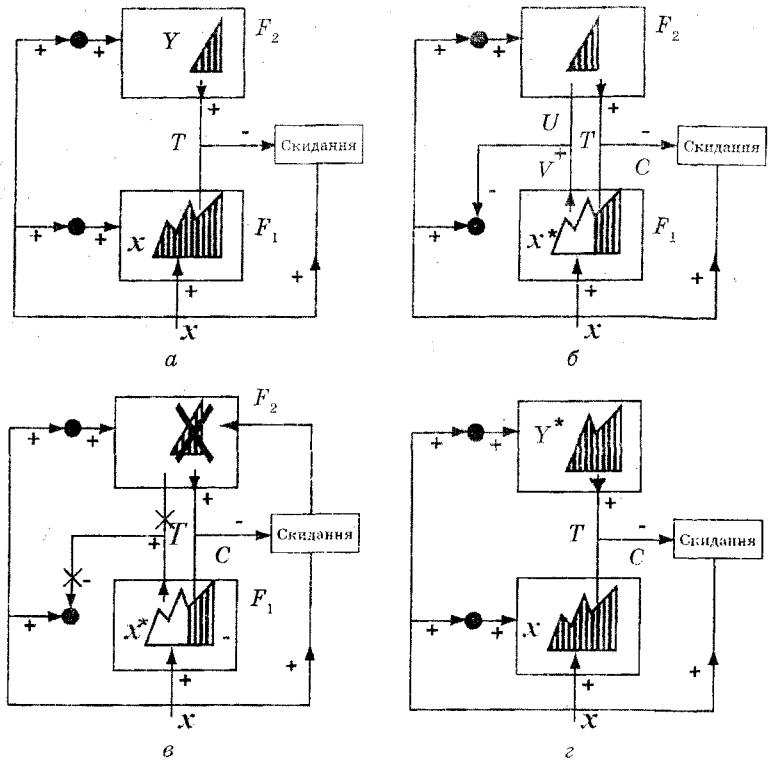


Рис. 20.2. Робота мережі ART-1

необхідного, з'являється сигнал скидання (рис. 20.2, *в*), за допомогою якого виключається образ Y, що виникає (гальмується активований нейрон шару розпізнавання). Потім здійснюється пошук іншого, більш відповідного даному вхідному сигналу $\boldsymbol{x}$ образу Y^*, що зберігається в F_2. Як ми вже зазначали, процес пошуку завершується, коли буде отриманий образ Y, що має необхідну подібність із сигналом $\boldsymbol{x}$, або коли будуть переглянуті всі образи, що зберігаються в шарі розпізнавання. В останньому випадку, якщо жоден з образів не задовольняє необхідний параметр подібності, у розпізнавальному шарі буде сформований новий образ, який відповідає сигналу $\boldsymbol{x}$, що подається.

Приклад 20.4. Нехай на вхід мережі ART-1, для якої заданий параметр подібності $\rho = 0{,}6$, надходять вектори розмірності $N = 5$ $\boldsymbol{x}_1 = (0\,0\,0\,0\,1)^T$, $\boldsymbol{x}_2 = (0\,0\,1\,1\,0)^T$, $\boldsymbol{x}_3 = (0\,1\,1\,1\,1)^T$, $\boldsymbol{x}_4 = (1\,1\,1\,1\,1)^T$. Нехай $\lambda = 1{,}5$.

Задамо відповідно до (20.7), (20.8) такі початкові значення параметрів ваг:

$$w_{ij}(0) = 0{,}2;\ t_{ji}(0) = 1,\ i = \overline{1,5};\ j = \overline{1,4}.$$

При надходженні на вхід мережі вектора $\boldsymbol{x}_1$ на її виході маємо

$$T_1 = \sum_{i=1}^{5} w_{1i}(0)x_{1i} = 0{,}2\cdot 0 + 0{,}2\cdot 0 + 0{,}2\cdot 0 + 0{,}2\cdot 0 + 0{,}2\cdot 1 = 0{,}2.$$

Обчислюємо параметр подібності для даного сигналу за формулою (20.10):

$$S_1 = \frac{1\cdot 0 + 1\cdot 0 + 1\cdot 0 + 1\cdot 0 + 1\cdot 1}{0+0+0+0+1} = 1.$$

Оскільки $S > \rho$, вхідний сигнал належить класу, визначеному нейроном T_1. Дійсно, оскільки $\boldsymbol{x}_1$ є першим сигналом, що надійшов, а в мережі ще не було запам'ятованих сигналів, то він й утворить перший збережений у пам'яті образ.
Зміна ваг відбувається за правилами (20.15) і (20.16):

$$w_{1i}(1) = \frac{1{,}5\cdot 0}{0{,}5+0+0+0+0+1} = 0,\ i = \overline{1,4};$$

$$w_{15}(1) = \frac{1{,}5\cdot 1}{0{,}5+0+0+0+0+1} = 1;$$

$$t_{1i}(1) = 0,\ i = \overline{1,4};$$

$$t_{15}(1) = 1.$$

Потім на вхід мережі подається сигнал $\boldsymbol{x}_2 = (0\,0\,1\,1\,0)^T$ й обчислюється реакція на нього нейрона T_1

$$T_1 = \sum_{i=1}^{5} w_{1i}(1)x_{2i} = 0.$$

Визначаючи для даного нейрона S, маємо $S = 0$. Внаслідок того, що $S < \rho$, даний вхідний сигнал не відноситься до класу, визначеного нейроном T_1. Тому вводимо новий нейрон T_2, для якого отримуємо

$$T_2 = \sum_{i=1}^{5} w_{2i}(0)x_{1i} = 0{,}2\cdot 0 + 0{,}2\cdot 0 + 0{,}2\cdot 1 + 0{,}2\cdot 1 + 0{,}2\cdot 0 = 0{,}4.$$

Оскільки $T_2 > T_1$, то відповідно до (18.4) вибираємо нейрон T_2. Для нього $S_2 = 1$, а елементи вагових матриць дорівнюватимуть

$$w_{2i}(1) = \begin{cases} 0 & \text{для } i = 1,2,5; \\ 0{,}6 & \text{для } i = 3,4; \end{cases}$$

$$t_{2i}(1) = \begin{cases} 0 & \text{для } i = 1,2,5; \\ 1 & \text{для } i = 3,4. \end{cases}$$

Отже, введено новий клас, визначений нейроном T_2.
При надходженні вхідного сигналу $\boldsymbol{x}_3 = (0\ 1\ 1\ 1\ 1)^T$ на виходах нейронів T_1 й T_2 маємо відповідно

$$T_1' = \sum_{i=1}^{5} w_{1i}(1) x_{3i} = 1;$$

$$T_2' = \sum_{i=1}^{5} w_{2i}(1) x_{3i} = 1{,}2.$$

Оскільки $T_2' > T_1'$, вибираємо нейрон T_2 (дійсно, хеммінгова відстань між $\boldsymbol{x}_2$ й $\boldsymbol{x}_3$ дорівнює 2, а між $\boldsymbol{x}_3$ й $\boldsymbol{x}_1$ — 3, тобто сигнал $\boldsymbol{x}_3$ ближче розташований до $\boldsymbol{x}_2$, а не до $\boldsymbol{x}_1$). Обчислюємо для цього нейрона параметр подібності S:

$$S' = \frac{2 \cdot (1 \cdot 1)}{4} = 0{,}5.$$

Через те, що $S' = 0{,}5 < \rho$, сигнал $\boldsymbol{x}_3$ не можна віднести до того ж класу, що й $\boldsymbol{x}_2$. Тому вводимо новий нейрон, T_3, ваги якого визначаються у такий спосіб:

$$w_{3i}(1) = \begin{cases} 0 & \text{для } i = 1; \\ 0{,}33 & \text{для } i = \overline{2,5}; \end{cases}$$

$$t_{3i}(1) = \begin{cases} 0 & \text{для } i = 1; \\ 1 & \text{для } i = \overline{2,5}. \end{cases}$$

При надходженні на вхід мережі сигналу $\boldsymbol{x}_4$ вихідні сигнали нейронів T_1, T_2, T_3 матимуть такі значення:

$$T_1'' = 1;\ T_2'' = 1{,}2;\ T_3'' = 1{,}32.$$

Тому відповідно до (20.4 вибираємо нейрон T_3. Обчислюючи параметр подібності, отримуємо

$$S'' = \frac{4 \cdot 1}{5} = 0{,}8 > \rho.$$

Отже, мережа віднесе сигнал $\boldsymbol{x}_4$ до того ж класу, що й $\boldsymbol{x}_3$.

20.2. Мережа ART-2

ART-2 є узагальненням мережі ART-1, здатним обробляти як двійкові, так і неперервні сигнали. Архітектура мережі ART-2 аналогічна архітектурі мережі ART-1 (рис. 20.1), однак нейрони шару порівняння мережі ART-2 мають більш складну структуру і є невеликою мережею, що складається із шести нейронів і служить буферним пристроєм між вектором вхідних сигналів $\boldsymbol{x}$ і векторами сигналів $\boldsymbol{c}$.

20.2.1. Динаміка мережі ART-2

Спрощену архітектуру мережі ART-2 з розкритою структурою одного нейрона шару порівняння F_1 наведено на рис. 20.3.

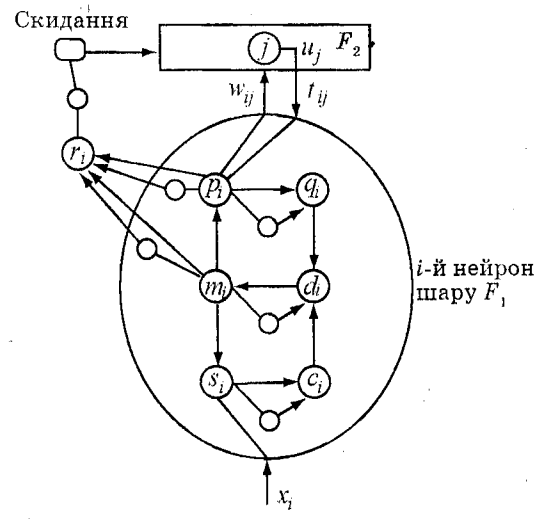


Рис. 20.3. Спрощена архітектура мережі ART-2

На рисунку наведено послідовність перетворення i-ї компоненти вхідного сигналу x_i в i-му нейроні шару F_1, що надходить на j-й нейрон шару F_2

$$x_i \to s_i \to c_i \to d_i \to m_i \to p_i \to T_j.$$

Тут $T_j = \sum_{i=1}^{N} p_j w_{ij}$ — сума всіх зважених ваг вхідних сигналів нейрона n_j, що надходять із шару F_1.

Вихідний сигнал нейрона n_j шару $F_2(n_j)$ подається на вхід нейрона n_j шару F_1 після таких перетворень:

$$n_j \to p_i \to q_i \to d_i \to m_i.$$

Динаміка i-го нейрона шару F_1 може бути описана рівнянням, аналогічним (20.3), (20.6)

$$\varepsilon \frac{do_i}{dt} = -Ao_i + (1 - Bo_i)I_i^+ - (C + Do_i)I_i^-, \qquad (20.17)$$

де $o_i (i = \overline{1,N})$ — вихідний сигнал нейрона n_i; I_i^+, I_i^- — відповідно підсилюючі і гальмуючі сигнали, що надходять на n_i; ε — мала додатна константа.

Якщо в (20.17) прийняти $B = C = 0$ й $\varepsilon \to 1$, опис n_i значно спроститься й набуде вигляду

$$o_i = \frac{I_i^+}{A + DI_i^-}. \qquad (20.18)$$

У цьому випадку рівняння, що описують динаміку нейрона n_i, який має вигляд, зображений на рис. 20.3, можуть бути записані у такий спосіб:

$$S_i = x_i + am_i; \qquad (20.19)$$

$$c_i = \frac{S_i}{e + \|S\|}; \qquad (20.20)$$

$$d_i = f(c_i) + bf(q_i); \qquad (20.21)$$

$$m_i = \frac{d_i}{e + \|d\|}; \qquad (20.22)$$

$$q_i = \frac{p_i}{e + \|p\|}; \qquad (20.23)$$

$$p_i = m_i + \sum_j g(n_j) t_{ij}. \qquad (20.24)$$

Тут $a \in (0,1)$, $b > 0$, $e > 0$, $\|\cdot\|$ — норма вектора $(\cdot)$, $f(x)$ — монотонно диференційована функція

$$f(x) = \begin{cases} \dfrac{2\theta x^2}{x^2 + \theta^2} & \text{при } 0 \le x \le \theta; \\ x & \text{в іншому випадку.} \end{cases} \qquad (20.25)$$

Параметр e в (20.20), (20.22), (20.23) відіграє регуляризуючу роль, коли норми відповідних векторів малі. На практиці нерідко застосовують спрощення, вибираючи $e = 0$ й $\|S\| = \|d\| = \|p\| = 1$.

Функція $g(n_j)$ у (20.24) призначена для визначення нейрона-переможця n_j шару F_2

$$g(n_j) = \begin{cases} k, & \text{якщо } T_j = \max\{T_k\}; \\ 0 & \text{в іншому випадку.} \end{cases} \qquad (20.26)$$

Тут n_j — вихідний сигнал нейрона n_j шару F_2; $k \in (0, 1)$.

Внаслідок (20.26) рівняння (20.24) набуває вигляду

$$p_i = \begin{cases} m_i, & \text{якщо шар } F_2 \text{ не активований;} \\ m_i + kt_{ji}, & \text{якщо нейрон } n_j \text{ шару } F_2 \text{ активований.} \end{cases} \qquad (20.27)$$

Для визначення відповідності вхідного сигналу, що надійшов, деякому збереженому в пам'яті образу й організації сигналу скидання формується вектор $r = (r_1, r_2, \ldots, r_n)^T$ з компонентами

$$r_i = \frac{m_i + lp_i}{\|m\| + \|lp\|}, \qquad (20.28)$$

де $0 < l < 1$.

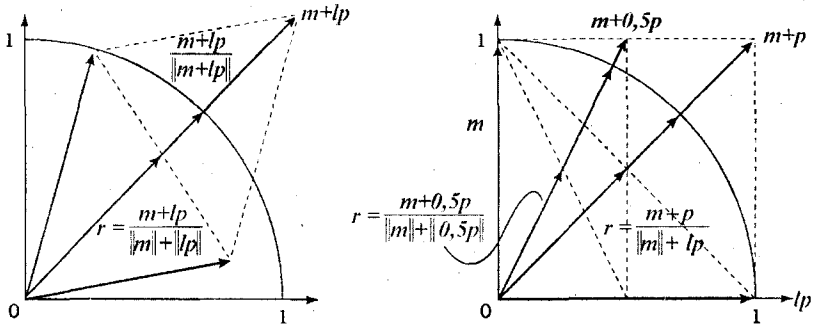
Сигнал скидання виникає, якщо поданий вхідний сигнал не відповідає в необхідному степені (для заданого параметра відповідності $0 < \rho < 1$) збереженому, тобто з виконанням умови

$$\frac{\rho}{e + \|r\|} > 1, \qquad (20.29)$$

де $e > 0$ — параметр, що виконує ту ж регулязуючу роль, що й в (20.20), (20.22), (20.23).

Геометричну ілюстрацію співвідношення (20.28) для двовимірного випадку наведено на рис. 20.4. Зображений тут випадок додатних координат вектора, що визначає образ, не є істотним

обмеженням, тому що вектор, який має від'ємні координати, може бути перетворений у вектор з додатними координатами. Для простоти прийнято, що $c = 0$.



а *б*

Рис. 20.4. Геометрична ілюстрація співвідношення (20.28)

Вектор r по суті визначає кут між векторами m і p. Загальний випадок взаємного розташування векторів m й p зображено на рис. 20.4, *а*. Найбільша відмінність цих векторів виявиться, якщо кут між ними становить 90°. Довжина вектора помилки r при цьому залежатиме від величини l. Як випливає з рис. 20.4, *б*, найбільша помилка буде при $\|m\| = \|lp\| = 1$, тобто при виборі $l = 1$. У цьому випадку

$$\|r\| = \frac{\|m+p\|}{\|m\|+\|p\|} = \frac{\sqrt{2}}{2}. \qquad (20.30)$$

Величина $\|r\|$ служить підставою для вибору параметра відповідності ρ. З урахуванням (20.29) і (20.30) маємо

$$\frac{\sqrt{2}}{2} \le \rho \le 1. \qquad (20.31)$$

20.2.2. Навчання мережі ART-2

Хоча архітектура даної мережі складніша, ніж ART-1, правила навчання залишаються практично такими ж, як в ART-1, тобто аналогічними (20.11), (20.12)

$$\frac{dw_{ij}}{dt} = g(n_j)\left[p_i - w_{ij}\right]; \qquad (20.32)$$

$$\frac{dt_{ji}}{dt} = g(n_j)\left[p_i - t_{ji}\right]. \qquad (20.33)$$

Оскільки вихідні сигнали $g(n_j)$ всіх нейронів n_j $(i \ne j)$ шару F_2, крім нейрона-переможця n_j, дорівнюватимуть нулю, їхні вагові коефіцієнти не зміняться. Ваги ж нейрона-переможця n_j змінюються відповідно до правил (20.32), (20.33), які з урахуванням (20.26) набувають вигляду

$$\frac{dw_{ij}}{dt} = k(1-k)\left[\frac{m_i}{1-k} - w_{ij}\right]; \qquad (20.34)$$

$$\frac{dt_{ji}}{dt} = k(1-k)\left[\frac{m_i}{1-k} - t_{ji}\right]. \qquad (20.35)$$

де $k \in (0, 1)$.

У сталому стані, коли

$$\frac{dw_{ij}}{dt} = \frac{dt_{ji}}{dt} = 0,$$

значення ваг, як випливає з (20.34), (20.35), дорівнюватиме

$$w_{ij} = t_{ji} = \frac{m_i}{1-k}. \qquad (20.36)$$

Як початкові значення w_{ij} й t_{ji} звичайно приймають

$$t_{ji} = 0,\ w_{ij} \le \left[(1-k)N^{\frac{1}{2}}\right]^{-1},$$

де N — розмірність вхідного сигналу.

20.3. Мережа ART-2a

Як й у мережі ART-2, тут використовується навчання без учителя, при якому подані образи, представлені неперервними векторами, мережа розподіляє по класах самостійно. Вхідні вектори, що надходять, попередньо нормалізуються у вхідному шарі $\hat{x}_i = x_i \|x\|^{-1}$ й на наступні шари подається сигнал

$$\hat{x}_i = \begin{cases} x_i, & \text{якщо } x_i > \theta; \\ 0 & \text{в іншому випадку, } i = \overline{1,N}, \end{cases} \qquad (20.37)$$

де θ — зміщення, вибране з умови

$$0 < \theta \le N^{-\frac{1}{2}}. \qquad (20.38)$$

Тут N — розмірність вектора вхідного сигналу.

За аналогією із сигналом T_j (20.4) визначається вхідний сигнал нейрона-переможця шару розпізнавання як

$$T_j = \max_i \{T_i\},$$

де T_j — вхідний сигнал нейрона n_j цього шару, що обчислюється як

$$T_j = \begin{cases} \alpha \sum_i \hat{x}_i, & \text{якщо нейрон } n_j \text{ є вільним;} \\ \hat{\boldsymbol{x}}^T \boldsymbol{w}_i, & \text{якщо нейрон } n_j \text{ є зв'язаним,} \end{cases} \qquad (20.39)$$

де $\alpha \le N^{-\frac{1}{2}}$; $\boldsymbol{w}_j$ — нормований вектор, обчислений у процесі навчання.

Перед початком процесу навчання всі нейрони шару розпізнавання є вільними, а в процесі навчання визначаються зв'язані нейрони. Якщо під час навчання одночасно були активовані кілька нейронів, вибір нейрона-переможця здійснюється або випадковим чином, або переможцем оголошується нейрон, що має найменший індекс. У зв'язку з тим, що вектори $\hat{\boldsymbol{x}}$ й $\boldsymbol{w}_j$ мають одиничну довжину, вхідний сигнал T_j нейрона n_j шару розпізнавання (20.39) є косинусом кута між цими векторами.

Після визначення нейрона-переможця відбувається корекція його вектора $\boldsymbol{w}_j$ відповідно до алгоритму

$$w_j(k+1) = \begin{cases} \hat{x}(k), & \text{якщо нейрон } n_j \text{ є вільним;} \\ \hat{w}_j(k+1), & \text{якщо нейрон є зв'язаним,} \end{cases} \qquad (20.40)$$

де

$$\hat{\boldsymbol{w}}_j(k+1) = \frac{\gamma\hat{\varphi} + (1-\gamma)w_j(k)}{\left\|\gamma\hat{\varphi} + (1-\gamma)w_j(k)\right\|}; \qquad (20.41)$$

$$\hat{\varphi} = (\varphi_1, \varphi_2, \ldots, \varphi_N)^T; \quad \varphi_i = \begin{cases} \hat{x}_i, & \text{якщо } w_{ji}(k) > 0; \\ 0 & \text{в іншому випадку.} \end{cases}$$

Тут $\gamma \in (0, 1)$ — коефіцієнт навчання.

Сигнал скидання виникає, якщо $T_j < \rho$, де $\rho \in [0, 1]$ — параметр відповідності.

Даний алгоритм навчання значно простіший за алгоритм (20.32), однак, як свідчать результати досліджень, отримувані при цьому результати практично збігаються з результатами, забезпечуваними мережею ART-2.

Основний недолік ART-2a для багатьох реалізацій — втрата всієї інформації, що зберігається у довжині вектора вхідного образу, оскільки всі образи приведені до одиничної евклідової довжини. Інакше кажучи, ART-2a не може знайти відмінності між двома вхідними образами A_1 й A_2, якщо $A_1 = c \cdot A_2$, де $c > 0$. Використання комплементарного кодування дозволяє всю інформацію, що перебуває в довжині некодованого вектора A, розмістити в напрямку результуючого вектора $I = (A, A_C)$. Даний метод використовується для розширення можливості розпізнавання мережами ART-2a і реалізується в мережі ART-2a-C.

Інший варіант зберегти інформацію, що перебуває в довжині вектора, під час обробки вхідних образів реалізований у мережі ART-2a-E й полягає у заміні показника відстані ART-2a евклідовою мірою подібності й виключенні процесу нормалізації довжини вхідних образів на етапі попередньої обробки й адаптації.

Приклад 20.5. Розглянемо приклади класифікації двовимірних даних різними варіантами мереж ART-2a. Образи подавалися у випадковому порядку, а навчання мережі припинялося після того, як кожен образ був поданий мінімальну, заздалегідь задану, кількість разів. Для забезпечення порівнянності результатів моделювання на різних типах мереж або з різними мережевими параметрами відбувалося з тим самим псевдовипадковим набором.

Тестовий набір складається з 1000 двовимірних образів, що являють собою рівномірно розподілені точки в одиничному квадраті. Величини вхідного образу взято з інтервалу [0,1; 1,0], що задовольняє обмеження на вхідні величини будь-якого описаного типу мережі ART. У просторовому розподілі точок немає очевидного розходження класів. Тому результати класифікації в різних експериментах демонструють тільки різну геометричну інтерпретацію вхідного простору образів і вид відповідних прототипів.

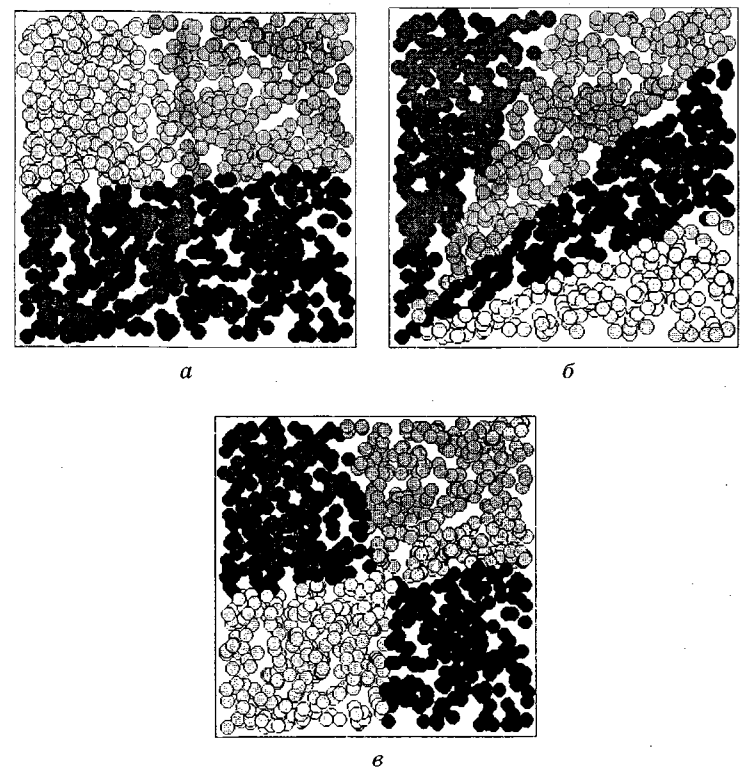
Прототипи мереж ART-2a безупинно модифікуються при поданні чергового вхідного образу. Тому вони ніколи не досягають стану рівноваги. Навчання мережі в цьому випадку можна припиняти, коли при черговому поданні образів, що навчають, вони потрапляють у ті самі класи. У більшості випадків такий стан установлюється після декількох циклів навчання, незалежно від природи й розміру вхідних образів. Із двовимірними вхідними образами, використаними в даних дослідженнях, така

стабільність завжди виникала при коефіцієнті навчання η рівному 0,1, і мінімум десяти подавань кожного образу. Геометрична інтерпретація прототипів ART-2a є усередненим вектором усіх образів, внесених в один клас. Значення коефіцієнта η впливає на кількість подавань, необхідних для переміщення прототипу в середню точку. Доцільно вибирати η по можливості більше, але так, щоб мережа залишалася стабільною й при цьому вимагалася мінімальною кількість подавань образів, що навчають. Режим швидкого навчання при η = 1 не може бути використаний стосовно мереж типу ART-2a, оскільки клас не буде стабільним, тому що прототип завжди приймає значення останнього вхідного образу, віднесеного до класу, що представляє цей прототип. На рис. 20.5 наведено результати класифікації за допомогою мережі ART-2a й модифікованих алгоритмів ART-2a-C й ART-2a-E. При класифікації набору образів чутливість вибиралася так, щоб кількість отримуваних класів була три або чотири. Навчання припинялося після того, як кожен образ був поданий мінімум десять разів. Хоча мережа ART-2a-C зберігає чотиривимірні вектори прототипів, їхня інтерпретація в просторі образу залишається двовимірною комплементарно закодованою точкою.

Із рис. 20.5 видно, що простір образу розділяється на класи різними типами мереж по-різному. Оскільки ART-2a нормалізують усі прототипи й незакодовані вхідні образи до одиничної евклідової довжини (3.69), то образ $I = (0{,}4;\ 0{,}5)$ сприймається мережею так само, як образ $I = (0{,}8;\ 1{,}0)$. Як наслідок, класи розділяють двовимірний простір образу вздовж радіальних ліній (рис. 20.5, *а*). Чутливість ρ визначає максимальний кут між вхідним вектором і вектором прототипу. Якщо $\rho = 0{,}94$, то $\beta = 20^\circ$.

Мережі ART-2a-C й ART-2a-E розділили простір образів на класи аналогічно, хоча ART-2a-C порівнює кут між комплементарно закодованим вхідним образом і прототипом, а ART-2a-E вимірює евклідову відстань між вхідним образом і прототипом. Це доводить, що комплементарне кодування зберігає інформацію про довжину незакодованого образу в напрямку закодованого. У випадку двовимірних вхідних образів функція вибору/зіставлення ART-2a-E визначає навколо будь-якого прототипу коло з радіусом $r = (1 - \rho)\sqrt{m} = (1 - \rho)\sqrt{2}$.

На рис. 20.5, *б* прийнято радіус $r = 0{,}45$ при $\rho = 0{,}68$. Вхідний образ вноситься в той клас, у якого прототип знаходиться на мінімальній евклідовій відстані від цього образу, або образ, що лежить в області відповідного кола. В іншому випадку вхідний образ лежить за межами кожного із уже наявних кластерних кіл, що спричиняє створення нового класу з новим прототипом.



а *б* *в*

Рис. 20.5. Класифікація двовимірних образів за допомогою мереж:
а — ART-2A, $\rho = 0{,}94$; *б* — ART-2A-C, $\rho = 0{,}86$;
в — ART-2A-E, $\rho = 0{,}68$

20.4. Мережа ART MAP

Дана мережа (ART MAP) є комбінацією двох мереж ART, якими можуть бути ART-1 або ART-2. Схематичне подання мережі ART MAP наведено на рис. 20.6.

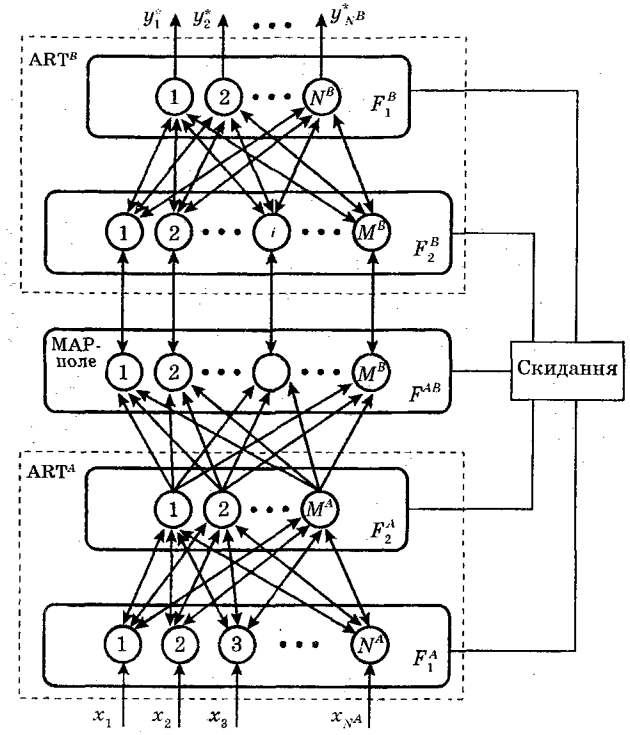


Рис. 20.6. Архітектура мережі ART MAP

Як видно з рисунка, мережа складається із двох мереж ART, ART^A, ART^B, з'єднаних з MAP-полем (асоціативною пам'яттю). Як і будь-яка мережа ART, кожна з мереж ART^A, ART^B має шар розпізнавання F_2^A й F_2^B і шар порівняння F_1^A и F_1^B. Вектор вхідних сигналів, що характеризує p-й образ $A_p = (a_{1p}, a_{2p}, \ldots, a_{N^A})^T$ розмірності $N^A \times 1$ надходить на ART^A, на вхід же мережі ART^B подається асоційований з A_p вихідний сигнал $B_p = (b_{1p}, b_{2p}, \ldots, b_{N^B})^T$. Шари F_1 й F_2 обох мереж зв'язані між собою, як правило, за допомогою ваг w_{ij}^a й t_{ji}^a, w_{ij}^b й t_{ji}^b. Зв'язки ж шарів F_2^A й F_2^B з MAP-полем організовані по-різному. Якщо всі нейрони шару F_2^A зв'язані з усіма нейронами MAP-поля й сигнали розповсюджується тільки в одному напрямку, від нейронів шару F_2^A до нейронів MAP-поля, то кожен нейрон шару F_2^B зв'язаний тільки з одним нейроном MAP-поля, і сигнали можуть передаватися в обох напрямках. Тому кількість нейронів шару F_2^B и MAP-поля збігається.

Класифікація пропонованих образів здійснюється у MAP-полі. Навчання мережі з учителем і застосування спеціальних методів кодування вхідних векторів підвищують ефективність розв'язання задачі класифікації.

20.5. Інші мережі ART

Мережа ART-2a є більш швидким варіантом ART-2. Як свідчить порівняльний аналіз, процес класифікації здійснюється обома мережами практично ідентично, однак швидкість процесу навчання мережі ART-2a вища. Тому багато авторів рекомендують застосовувати ART-2a для розв'язання складних задач.

20.5.1. Мережа ART-3

Архітектура цієї мережі значно більшою мірою відповідає моделям біологічних механізмів. Використання спеціального механізму порівняння векторних сигналів, що надходять на вхід мережі й зберігаються в ній, дозволяє досліджувати процеси синаптичної передачі й обробки інформації, пасивність рецепторних молекул на постсинаптичній стороні й т. д. Більш докладно ART-3 описані в [126].

20.5.2. FUZZY-ART

Цей тип мережі представляє мережу ART-1, що використовує нечітку логіку. Застосування операторів нечіткої логіки забезпечує більшу гнучкість ART-1. Дані оператори застосовуються як у фазі розпізнавання при формуванні класів, так й у фазі навчання. Крім того, за допомогою цих операторів формується критерій скидання.

Ці мережі мають можливості багатошарового персептрона.

Контрольні запитання

1. Що таке дилема стабільності — пластичності?
2. Назвіть основні властивості мережі ART. Які існують різновиди цих мереж?
3. Наведіть архітектуру ART-1. Яке призначення основних модулів?
4. Поясніть принципи функціонування мережі ART-2. У чому відмінність ART-2 від ART-1?
5. Що являє собою модифікація мережі ART-2a?
6. Наведіть архітектуру мережі ART MAP. Які мережі є основою для її побудови?

21. МЕТОДИ СПРОЩЕННЯ СТРУКТУРИ МЕРЕЖ

При практичному застосуванні ШНМ зовсім не останню роль відіграє їхня складність. Тому під час побудови мережі прагнуть досягти її максимально простої архітектури.

Існує два підходи до вирішення даної проблеми: зменшення кількості елементів вагової матриці мережі (*weight pruning*) і зменшення кількості використовуваних нейронів (*unit pruning*). У першому випадку прирівнюються нулю або елементи матриці ваг, що мають мінімальні значення, або ті елементи, наявність яких впливає на зменшення помилки мережі. Найбільш відомими представниками цих методів є *OBD* (*Optimal Brain Damage*) [128] й *OBS* (*Optimal Brain Surgeon*) [129, 130].

Методи вилучення нейронів поширені менше, найбільш відомим з них є так звана *скелетизація* (*skeletonization*) [131], застосовувана для зменшення кількості нейронів як вхідного, так і прихованих шарів мережі.

21.1. Методи вилучення ваг

21.1.1. Вилучення ваг, що мають найменші значення

Ідея даного методу (*weight decay*) має багато спільного з ідеєю регуляризації [132], коли у звичайний функціонал, що мінімізується, вводиться додатковий штрафний член

$$I_d = \frac{1}{2}\sum_p\sum_j (y^*_{pj} - y_{pj})^2 + \frac{\lambda}{2}\sum_i\sum_j w_{ij}^2, \qquad (21.1)$$

де $\lambda > 0$ — коефіцієнт штрафу; $p = \overline{1,P}$ — кількість поданих образів навчання.

Мінімізація функціонала (21.1) призводить до такого градієнтного алгоритму корекції ваг:

$$\Delta_p w_{ij}(k+1) = -\gamma(y^*_{pj}(k) - y_{pi}(k))y'_{pj}(k) + \lambda w_{pij}(k), \qquad (21.2)$$

де $y'_{pj}(k) = \dfrac{\partial y_{pj}(k)}{\partial w_{ij}(k)}$.

При цьому вага, значення якої перебуває в заданому ε-околі нуля, віддаляється.

Приклад 21.1. При моделюванні функції, заданої в прикладі 3.11, за допомогою двошарового персептрона 2-10-6-1 отримано такі значення елементів вагових матриць W і векторів зміщень θ (у наведених нижче даних підкреслено значення ваг, що мають найменше значення):

$$W_1 = \begin{vmatrix} 1{,}3541 & 0{,}89078 \\ 1{,}2782 & 1{,}323 \\ 3{,}7394 & 2{,}0062 \\ -0{,}1918 & 2{,}1641 \\ 1{,}6283 & -2{,}6483 \\ 6{,}0488 & 3{,}6176 \\ -0{,}8538 & 3{,}6176 \\ -1{,}7991 & 4{,}3197 \\ 2{,}7639 & -1{,}7269 \\ 0{,}84795 & 1{,}8147 \end{vmatrix},$$

$$W_2 = \begin{vmatrix} -3{,}3322 & 1{,}4698 & 2{,}1891 & -4{,}0497 & 1{,}3107 & -0{,}6311 & 0{,}2324 & -1{,}6441 & -0{,}0669 & -0{,}2518 \\ -2{,}4683 & -1{,}261 & 0{,}301 & 4{,}3771 & -3{,}5357 & 0{,}2291 & -0{,}1907 & 2{,}9924 & 3{,}5416 & 2{,}1398 \\ -0{,}5298 & -1{,}4915 & 2{,}9326 & -0{,}4842 & 2{,}0004 & -1{,}049 & 0{,}9067 & -1{,}9921 & 1{,}8798 & 2{,}6475 \\ 2{,}8347 & -1{,}5191 & -1{,}1836 & 4{,}4641 & -0{,}9081 & 0{,}7034 & 0{,}0479 & 1{,}4633 & -0{,}9296 & 0{,}2314 \\ -0{,}841 & -0{,}2568 & 1{,}6651 & -0{,}7962 & -1{,}1131 & 0{,}9279 & 0{,}5369 & 1{,}9487 & -0{,}5671 & 0{,}7010 \\ -2{,}9044 & 1{,}2693 & 0{,}0406 & 1{,}8313 & 0{,}2007 & -0{,}0294 & 0{,}5615 & -2{,}7324 & -0{,}0507 & 2{,}163 \end{vmatrix},$$

$$W_3 = \begin{vmatrix} 0{,}9474 & -0{,}1958 & 0{,}0028 & 1{,}1495 & -0{,}0068 & 4{,}0316 \end{vmatrix},$$

$$\theta_1 = \begin{vmatrix} -1{,}5563 \\ 2{,}1759 \\ -3{,}368 \\ 1{,}6145 \\ 0{,}1352 \\ -1{,}6329 \\ -1{,}2381 \\ -3{,}4141 \\ 0{,}4876 \\ -1{,}1356 \end{vmatrix}, \quad \theta_2 = \begin{vmatrix} 0{,}8242 \\ -0{,}5099 \\ -2{,}0658 \\ -1{,}2879 \\ -0{,}4490 \\ -7{,}9501 \end{vmatrix}, \quad \theta_3 = \begin{vmatrix} 4{,}0332 \end{vmatrix}.$$

Помилка апроксимації після 100 тактів навчання дорівнюва-

Вилучення ваг, що мають мінімальні значення, привело до отримання персептрона 2-10-4-1 з такими параметрами:

$$W_1 = \begin{vmatrix} 4{,}2116 & 0{.}303 \\ -1{,}2276 & 1{,}5171 \\ 0{,}0421 & 2{,}3269 \\ 0{,}0724 & 2{,}3744 \\ 2{,}4255 & 0{,}7909 \\ 2{,}3950 & 1{,}7729 \\ -1{,}1591 & 1{,}9489 \\ -8{,}5543 & 5{,}6831 \\ 2{,}3625 & -0{,}4705 \\ 1{,}5707 & 3{,}1528 \end{vmatrix},$$

$$W_2 = \begin{vmatrix} -0{,}9502 & -3{,}7149 & -1{,}9241 & -2{,}2787 & -2{,}1802 & 2{,}3239 & -2{,}8475 & -0{,}4753 & -0{,}4878 & 0{,}6121 \\ -1{,}0526 & -3{,}1155 & -1{,}6270 & -2{,}3335 & 5{,}3819 & -4{,}8737 & -2{,}388 & -0{,}3898 & -0{,}2191 & 0{,}1930 \\ -2{,}8379 & -4{,}7226 & -10{,}7118 & 4{,}3610 & -4{,}1643 & 3{,}3090 & 4{,}0071 & 0{,}0640 & 1{,}4579 & 3{,}5305 \\ -1{,}1147 & -1{,}1339 & -2{,}3014 & 2{,}0987 & -1{,}8886 & 1{,}3035 & -2{,}4273 & -0{,}1546 & 0{,}1378 & 1{,}3964 \end{vmatrix},$$

$$W_3 = \begin{vmatrix} -0{,}3130 & 0{,}3128 & 2{,}006 & 0{,}6359 \end{vmatrix},$$

$$\theta_1 = \begin{vmatrix} -4{,}4744 \\ 2{,}6391 \\ -1{,}6981 \\ 1{,}8654 \\ -0{,}9917 \\ -0{,}7259 \\ -2{,}8348 \\ -2{,}0497 \\ 2{,}2832 \\ -2{,}1935 \end{vmatrix}, \quad \theta_2 = \begin{vmatrix} 1{,}3456 \\ 1{,}1837 \\ -8{,}3667 \\ -6{,}2985 \end{vmatrix}, \quad \theta_3 = \begin{vmatrix} 2{,}6414 \end{vmatrix}.$$

Помилка апроксимації при цьому склала $2{,}69 \cdot 10^{-5}$.

Спрощення структури персептрона призвело до скорочення часу його навчання.

21.1.2. Метод OBD

Цей метод дозволяє на основі аналітичного упередження впливу зміни вектора вагових параметрів на використовуваний функціонал помилки так скорегувати вектор ваг, щоб при обнулінні деяких з них значення функціонала збільшувалося незначною мірою. Хоча OBD застосовують і за наявності зв'язаних ваг (*shared weights*), розглянемо найпростіший випадок, коли ваги w_i незалежні.

Обидва методи, і OBD, і OBS, використовують розкладання в ряд Тейлора функціонала помилки

$$\Delta I = \left(\frac{\partial I}{\partial \boldsymbol{w}}\right)^T \Delta \boldsymbol{w} + \frac{1}{2}(\Delta \boldsymbol{w})^T H \Delta \boldsymbol{w} + O(\| \Delta \boldsymbol{w} \|^3) \qquad (21.3)$$

або

$$\Delta I = \sum_i g_i \Delta w_i + \frac{1}{2}\sum_i h_{ii} \Delta w_i^2 + \frac{1}{2}\sum_{i \neq j} h_{ij} \Delta w_i \Delta w_j + O(\| \Delta \boldsymbol{w} \|^3), \qquad (21.4)$$

де w_i — компоненти вектора $\boldsymbol{w}$; $g_i = \dfrac{\partial I}{\partial \boldsymbol{w}_i}$; $H = [h_{ij}] = \left[\dfrac{\partial^2 I}{\partial w_i \partial w_j}\right]$ — матриця Гессе.

Задача полягає в знаходженні параметрів, які найменше впливають на збільшення функціонала I. Оскільки гессіан зростає квадратично зі збільшенням кількості ваг, у зв'язку з чим у мережах великої розмірності обчислювальні витрати стають істотними, OBD використовує три спрощення:

1) гессіан приблизно подають діагональною матрицею, нехтуючи недіагональними елементами $h_{ij} = (\partial^2 I / \partial w_i \partial w_j)$ $i \neq j$;

2) передбачається, що метод застосовується після навчання мережі, тобто після досягнення локального мінімуму I, коли $g_i = \partial I / \partial w_i = 0$;

3) функціонал помилки в околі мінімуму є квадратичним, тобто всіма членами розкладання вищих порядків можна знехтувати.

За умов виконання всіх цих пропозицій (21.3) набуває вигляду:

$$\Delta I = \frac{1}{2}\sum_i h_{ii} \Delta w_i^2. \qquad (21.5)$$

Позначивши входи, виходи й квадратичну помилку мережі при поданні p-го образу відповідно

$$y_{pj} = f(z_{pj});$$

$$z_{pj} = \sum_i y_{pi} w_{ij};$$

$$I_p = \sum_j (y_{pj}^* - y_{pj})^2,$$

отримаємо

$$h_{ij,ij} = \frac{\partial^2 I_p}{\partial w_{ij} \partial w_{ij}} = \frac{\partial^2 I_p}{\partial z_j^2} y_i^2. \qquad (21.6)$$

Використовувані в алгоритмі для корекції ваг внутрішніх шарів мережі другі похідні мають вигляд

$$\frac{\partial^2 I_p}{\partial z_j^2} = f^2(z_{pj}) \sum_k w_{jk}^2 \frac{\partial^2 I_p}{\partial z_k^2} - f''(z_{pj}) \frac{\partial I_p}{\partial z_k}. \qquad (21.7)$$

Для вихідних нейронів друга похідна функціонала помилки обчислюється в такий спосіб:

$$\frac{\partial^2 I_p}{\partial z_j^2}=2f'(z_{pj})^2-2(y^*_{pj}-y_{pj})f''(z_{pj}). \qquad (21.8)$$

Отже, компоненти діагональної апроксимації матриці Гессе обчислюються аналогічно градієнтам функціонала помилки в алгоритмі зворотного поширення. Наявність операції віднімання у (21.8) може призвести до нестійкості алгоритму. Тому в деяких випадках другим членом у (21.8) нехтують (це відповідає апроксимації Левенберга — Маркуардта), що гарантує позитивність оцінок коефіцієнтів матриці Гессе.

Остаточно алгоритм OBD може бути поданий у такий спосіб:

1. Вибирається «прийнятна» архітектура мережі.
2. Відбувається навчання мережі (до досягнення локального мінімуму).
3. Обчислюються другі похідні h_{kk} для кожної ваги k.
4. Обчислюється значущість кожної ваги k (*saliencies*) за формулою:

$$S_k = h_{kk}\frac{w_k^2}{2}.$$

5. Упорядковуються всі ваги щодо S_k і вага, що має мінімальне значення S_k, прирівнюється нулю (як варіант може відбуватися його зменшення).
6. Обчислюється помилка мережі. Якщо вона задовольняє заданому значенню — зупинник, інакше — перехід до п. 2.

В описаному в роботі [128] експерименті щодо застосування повнозв'язної мережі для розпізнавання рукописних цифр, що містила 10^5 зв'язків, 2578 вільних вагових параметрів (*intensives weight sharing*), 9300 даних, що навчають, й 3350 тестових даних вдалося зменшити кількість параметрів до 800 без істотного збільшення помилки.

Однак у багатьох випадках таке спрощення матриці Гессе не дозволяє отримати структуру мережі мінімальної складності, а в деяких випадках, наприклад, при вирішенні проблеми «що виключає АБО», взагалі можуть бути вилучені інші ваги.

21.1.3. Метод OBS

Цей метод є узагальненням OBD і використовує обчислення повної матриці Гессе [129]. З огляду на те, що OBS-метод є більш складним у порівнянні з OBD, він дозволяє вилучати більшу кількість ваг.

Як і в методі OBD, у цьому методі здійснюється апроксимація функціонала (21.3), однак на відміну від OBD при цьому використовуються дві останні спрощені умови.

Вилучення ваг w_q методом OBS задається рівнянням $\Delta w_q + w_q = 0$, при якому зміна ваги на Δw_q призводить до його обнуління, тобто

$$e_q^T \Delta \boldsymbol{w} + w_q = 0, \qquad (21.9)$$

де $\boldsymbol{e}_q$ — q-й одиничний вектор у просторі ваг; $e_q^T \Delta \boldsymbol{w} = \Delta w_q$.

Задача полягає в пошуку мінімуму квадратичного члена (21.3)

$$\min_q \left\{ \min_{\Delta w} \left(\frac{1}{2} (\Delta \boldsymbol{w})^T H \Delta \boldsymbol{w} \right) \right\} \qquad (21.10)$$

при виконанні умови (21.9).

Перший мінімум у (21.10), $\min_q\{\cdot\}$, служить для того, щоб у випадку появи декількох ваг, що призводять до однакової зміни функціонала, вибрати той, який має найменший індекс.

Для розв'язання (21.10) складається функція Лагранжа

$$L = \frac{1}{2}(\Delta \boldsymbol{w})^T H \Delta \boldsymbol{w} + \lambda \left(e_q^T \Delta \boldsymbol{w} + w_q \right), \qquad (21.11)$$

де λ — множник Лагранжа, що підлягає визначенню.

Мінімізація (21.11) дає [130]

$$\Delta \boldsymbol{w} = -\frac{w_q}{\left[H^{-1}\right]_{qq}} H^{-1} \boldsymbol{e}_q; \qquad (21.12)$$

$$L_q = \frac{1}{2}\frac{w_q^2}{\left[H^{-1}\right]_{qq}}. \qquad (21.13)$$

В обох рівняннях використовується обернена матриця Гессе. Зазначимо, що ні H, ні H^{-1} не є діагональними.

Вираз (21.12) — рівняння корекції ваг, а (21.13) визначає значення L_q (*saliency*), що характеризує зростання помилки при вилученні ваги w_q.

Алгоритм OBS може бути поданий у такий спосіб:

Пункти 1 й 2 збігаються із п. 1 і п. 2 алгоритму OBD.

3. Обчислюється обернена матриця Гессе H^{-1}.
4. Обчислюється значущість кожної ваги q за формулою (21.13).

Якщо $L_q << I$, дана вага вилучається й здійснюється перехід до п. 5, інакше вага залишається.

5. Знайдене q використовується в (21.12) для зміни всіх ваг і здійснюється перехід до п. 3.

Слід зазначити, що жодна вага не може бути вилучена, якщо після її вилучення функціонал помилки істотно зростає.

Цей метод забезпечує найкращі результати, якщо функціонал помилки в точці мінімуму є квадратичним.

Труднощі реалізації OBS-алгоритму полягають у необхідності обчислення оберненої матриці Гессе H^{-1}, що під час розв'язання практичних задач містить тисячі або навіть мільйони елементів. Тому для спрощення обчислень запропоновано алгоритми, що використовують подання матриці Гессе у вигляді рекурсивно обумовленої коваріаційної матриці деяких градієнтів.

OBS вирішує багато задач більш успішно, ніж зворотне поширення помилки. У *Nettalk*-мережі завдяки використанню OBS вдалося первинні 5546 ваг зменшити до 2438, а після додаткового навчання й подальшого спрощення — до 1560 ваг [2, 6].

21.2. Методи вилучення нейронів

У процесі вилучення нейронів так чи інакше оцінюються наслідки, викликані їхнім вилученням і відображені на значенні величини помилкової реакції мережі.

21.2.1. Вилучення нейронів з урахуванням їх важливості

Метод *skeletonization*, запропонований у роботі [131], призначений для вилучення нейронів як вхідного, так і прихованих шарів і заснований на використанні у функціоналі помилки додаткового члена — *показника важливості нейрона* ρ, визначеного як різниця між помилкою всієї мережі й помилкою мережі, з якої цей нейрон вилучений.

Якщо важливість кожного нейрона мережі, що складається з n нейронів, визначати в такий спосіб (при цьому для одного нейрона необхідно P образів), то знадобилося б $O(nP)$ подавань образів, що навчають. Тому для скорочення кількості обчислень при визначенні ρ використовується наближена формула, для виведення якої вводять параметр *впливу* (*attention strength*) α_i $(i=\overline{1,n})$ кожного нейрона

$$y_j = f\left(\sum_i w_{ij}\alpha_i y_i\right). \qquad (21.14)$$

Якщо $\alpha_i = 0$, то нейрон не чинить жодного впливу на мережу, $\alpha_i = 1$ відповідає звичайному нейрону (рис. 21.1).

Подавши значущість нейронів як

$$\rho_i = I_{\alpha_i=0} - I_{\alpha_i=1}, \qquad (21.15)$$

можна апроксимувати коефіцієнт ρ_i похідної $\frac{\partial I}{\partial \alpha_i}$ у точці $\alpha_i = 1$

$$\lim_{\gamma\to 1}\frac{I_{\alpha_i=\gamma} - I_{\alpha_i=1}}{\gamma - 1} = \frac{\partial I}{\partial \alpha_i}\Big|_{\alpha_i=1}. \qquad (21.16)$$

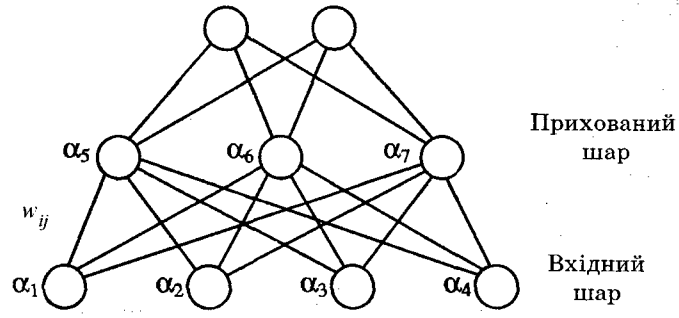


Рис. 21.1. Мережа з коефіцієнтами впливу

Якщо припустити, що похідна в точці $\alpha_i = 1$ дає добре наближення лівої частини рівняння й у випадку $\gamma = 0$, то

$$\frac{I_{\alpha_i=0} - I_{\alpha_i=1}}{-1} \approx \frac{\partial I}{\partial \alpha_i}\Big|_{\alpha_i=1} \quad \text{або} \quad -\rho_i \approx \frac{\partial I}{\partial \alpha_i}\Big|_{\alpha_i=1}. \qquad (21.17)$$

Отже, замість ρ_i використовують її оцінку $\hat{\rho}_i$

$$\hat{\rho}_i = -\frac{\partial I}{\partial \alpha_i}\Big|_{\alpha_i=1}. \qquad (21.18)$$

Ця похідна може бути обчислена методом, аналогічним методу зворотного поширення помилки.

На практиці похідна $\frac{\partial I}{\partial \alpha_i}$ швидко зменшується з часом, так що навіть середнє значення $\hat{\rho}_i$ зменшується експоненціально. В [131] запропоновано проводити корекцію $\hat{\rho}_i$ відповідно до формули

$$\hat{\rho}_i(k+1)=0{,}8\hat{\rho}_i(k)+0{,}2\frac{\partial I}{\partial \alpha_i}. \qquad (21.19)$$

Важливим у цьому методі є те, що на відміну від обчислення квадратичної помилки, використовуваної при настроюванні ваг, для визначення значущості застосовується лінійний функціонал

$$I_{linear}=\sum_p\sum_j\left|y^*_{pj}-y_{pj}\right|. \qquad (21.20)$$

Як показано в [131], звичайна квадратична функція від помилки забезпечує отримання поганих оцінок, коли значення реальних виходів близькі до необхідного. Тому в даній роботі для настроювання ваг використовувався квадратичний критерій, а для оцінювання значущості — лінійний.

Остаточно алгоритм *skeletonization* може бути поданий так:

1. Мережа навчається доти, поки виходи всіх нейронів не стануть достатньо близькі до необхідного.
2. Обчислюється значущість кожного нейрона.
3. Вилучається нейрон з найменшою значущістю.
4. Зупинник, якщо критерій зупинника задовольняється, в іншому випадку — перехід до п. 1.

21.2.2. Вилучення нейронів прихованого шару з використанням вартісної функції

Для вилучення нейронів прихованого шару *С. Генсон* і *Л. Пратт* використали підхід, аналогічний *weight decay* (21.1), тобто доповнювали критерій I вартісним показником C

$$I_C=I+C. \qquad (21.21)$$

Корекція ваг здійснюється з умови $\min_w I_c$.

Вид алгоритму корекції залежить від виду вартісної функції C. Так, якщо C — квадратична функція відносно w_{ij}, тобто $C=\sum_i\sum_j w_{ij}^2$, то алгоритм корекції ваг має вигляд

$$\Delta w_{ij}(k+1)=\gamma\left(-\frac{\partial I}{\partial w_{ij}}-2w_{ij}(k)\right), \qquad (21.22)$$

звідки

$$w_{ij}(k)=\gamma\sum_{m=1}^{k}\left[(1-2\gamma)^{k-m}\left(-\frac{\partial}{\partial w_{ij}}I(m)\right)\right]+(1-2\gamma)^k w_{ij}(0). \qquad (21.23)$$

Як вартісну функцію *Д. Румельхарт* запропонував таку:

$$C=\sum_i\sum_j\frac{w_{ij}^2}{(1+w_{ij}^2)}, \qquad (21.24)$$

використання в алгоритмі похідних якої

$$\frac{\partial C}{\partial w_{ij}}=\frac{2w_{ij}}{\left(1+w_{ij}^2\right)^2} \qquad (21.25)$$

дозволяє малі ваги швидко перетворити в нуль, а більші змінити незначною мірою. Перехід від урахування вартості ваги до урахування вартості нейрона здійснюється шляхом підсумовування значень ваг кожного прихованого нейрона

$$w_j=\sum_i\left|w_{ij}\right|. \qquad (21.26)$$

Поряд з видозміненою формою функції (21.24)

$$C=\sum_j\frac{w_j^2}{\left(1+w_j^2\right)} \qquad (21.27)$$

з похідною

$$\frac{\partial C}{\partial w_{ij}}=\frac{2w_j\,\mathrm{sgn}(w_{ij})}{\left(1+w_j^2\right)^2} \qquad (21.28)$$

застосовується також гіперболічна

$$C=\sum_j\frac{w_j}{1+\lambda w_j} \qquad (21.29)$$

з похідною

$$\frac{\partial C}{\partial w_{ij}}=\frac{\lambda\,\mathrm{sgn}(w_{ij})}{\left(1+w_j\right)^2} \qquad (21.30)$$

і спадна експоненціальна функція

$$C=\sum_j\left(1-e^{-\lambda w_j^2}\right) \tag{21.31}$$

з похідною

$$\frac{\partial C}{\partial w_{ij}}=\frac{\lambda\ \mathrm{sgn}(w_{ij})}{e^{\lambda w_j}}. \tag{21.32}$$

Хоча експоненціальна спадна функція в деяких задачах виявилася більш вдалою, її застосування, на жаль, іноді призводить до того, що процес навчання може не збігатися. Імовірно, це відбувається через використання градієнта функціонала як для зменшення помилки мережі, так і для вилучення прихованого нейрона.

21.2.3. Вилучення ваг з використанням вартісної функції

У роботі [134] запропоновано як штрафну функцію, аналогічну використовуваній у (21.1), (21.21), застосувати квадратичну вартісну функцію вихідних сигналів нейронів прихованого шару $C=\sum_j C(y_j)^2$, де $C(y_j)$ — монотонна функція виходу j-го нейрона.

Таким чином, функціонал, що мінімізується, набуває вигляду

$$I_C=\mu_I I+\mu_C C, \tag{21.33}$$

де μ_I,μ_C — вагові коефіцієнти окремих критеріїв.

Для зменшення вартості прихованих шарів, розташованих ближче до вихідного, обчислюють похідні

$$\frac{\partial C}{\partial w_{ij}}=\frac{\partial C}{\partial z_j}\frac{\partial z_j}{\partial w_{ij}}=\delta_{c,j}\frac{\partial z_j}{\partial w_{ij}}=\delta_{c,j}y_i; \tag{21.34}$$

$$\delta_{c,j}=\frac{\partial C\left(y_j^2\right)}{\partial z_j}=\frac{\partial C\left(y_j^2\right)}{\partial y_j^2}\frac{\partial y_j^2}{\partial y_j}\frac{\partial y_j}{\partial z_j}=2C'y_j f_j'(z_j), \tag{21.35}$$

де $C'=\frac{\partial C(y_j)^2}{\partial y_j}$; $f_j'(z_j)$ — похідна функції активації (логістична) входів мережі.

Для ваг усіх шарів, що перебувають після розглянутого, вартісна функція дорівнює нулю. Якщо аналізованому шару передує прихований, то для цього попереднього шару

$$\delta_{c,j}=f_j'(nz_j)\sum_k\delta_{c,k}w_{jk}, \tag{21.36}$$

що збігається зі стандартним алгоритмом зворотного поширення помилки, який відрізняється від нього тільки зворотно розповсюджуваним членом — сумарним сигналом помилки. Вираз для корекції ваг з урахуванням обернено розповсюджуваної помилки, і вартості набуває вигляду

$$\Delta w_{ij}=-\gamma\mu_I y_i\delta_{I,j}-\gamma\mu_C y_i\delta_{C,j}=-\gamma y_i\delta_{IC,j}, \tag{21.37}$$

де $\delta_{IC,j}$ — сумарний сигнал помилки, що враховує як помилку мережі, так і вартість вихідного сигналу j-го нейрона.

Для більш глибоко розташованих шарів можна записати відому формулу

$$\delta_{IC,j}=f_j'(z_j)\sum_m(\mu_I\delta_{I,m}+\mu_C\delta_{C,m})w_{im}=f_j'(z_j)\sum_m\delta_{IC,m}w_{jm}, \tag{21.38}$$

де $\delta_{I,m}$ — сигнал помилки наступного нейрона m;
$\delta_{C,m}$ — збільшення вартості;
$\delta_{IC,m}$ — сумарний сигнал помилки наступного нейрона k;
w_{jm} — вага зв'язку нейрона j з нейроном m.

Як й у випадку застосування критерію (21.21), ефективність (21.33) залежить від конкретного виду функцій $C(y_i)$.

У роботі [134] наведено результати експерименту з різними функціями вартості для прихованих нейронів, такими як $C_0(y^2)=y^2$, $C_1(y^2)=\log(1+y^2)$, $C_2(y^2)=y^2(1+y^2)^{-1}$, для яких

$$C_n'=\frac{\partial C_n(y^2)}{\partial y^2}=\frac{1}{\left(1+y^2\right)^n}.$$

При $n=0$ нейрони з високою й низькою вартістю штрафуються однаково, при $n=1$ нейрони з низькою вартістю штрафуються сильніше, при $n=2$ асимптотичні величини штрафу для нейронів з високою й середньою вартістю збігаються.

Даний алгоритм мінімізує одночасно квадрат помилки виходів прихованих нейронів і кількість цих нейронів, оскільки більшій кількості активних прихованих нейронів відповідає більш висока вартість.

Метод може бути модифікований шляхом його комбінування з *weight decay*-методом (21.1), наприклад, у такий спосіб:

$$\tilde{I}=\mu_I I+\mu_C C+\mu_w W. \tag{21.39}$$

Порівняльний аналіз методів свідчить, що найбільш трудомістким унаслідок необхідності обчислення й обернення повної матриці Гессе є OBS і тому він не зручний, а найчастіше не застосовуваний у мережах великої розмірності. Більш швидким у порівнянні з ним, однак менш швидким порівняно з простим вилученням найменших ваг є OBD. У всіх випадках тестування OBD вилучав менше зв'язків, ніж прості методи (практично завжди просте вилучення дає задовільні результати). Розглянуті методи вилучення нейронів (*skeletonization*), будучи найбільш швидкодіючими, як правило, видаляють менше ваг, але більше нейронів у порівнянні з методами OBS, OBD та ін.

Контрольні запитання

1. Які існують підходи до спрощення структури мереж?
2. Що являє собою метод спрощення структури мереж шляхом вилучення ваг, які мають найменше значення?
3. Які припущення лежать в основі застосування методу OBD?
4. Що є причиною нестійкості методу OBD?
5. Які методи апроксимації використовують для підвищення обчислювальної стійкості методу OBD?
6. Яка послідовність обчислень в алгоритмі OBD?
7. У чому полягає відмінність методів OBD й OBS?
8. У який спосіб обчислюється значущість нейрона в алгоритмі OBS?
9. Назвіть послідовність дій в алгоритмі OBS.
10. У чому полягає метод *skeletonization*?
11. Які вартісні функції використовуються для вилучення нейронів?
12. Яким чином здійснюється вилучення нейронів прихованого шару з використання вартісних функцій?
13. Які з методів вилучення нейронів є найбільш ефективними?

22. ГЕНЕТИЧНІ АЛГОРИТМИ

Генетичні алгоритми (ГА) запропоновано й досліджено *Дж. Голландом* і його студентами в 1975 році [135], однак їхнє практичне застосування почалося через майже два десятиліття. Початкова мета полягала в побудові методів оптимізації, що використовують механізми природного добору й генетики, зокрема, елементи випадковості й паралелізму, і супроводжуваних порівняно простими обчисленнями. При цьому передбачалося, що наявна деяка початкова популяція, що має деяку генетичну структуру, шляхом деяких генетичних операцій передачі спадковості (*селекції, розмноження* й *мутації*) утворювала вітки, деякі з яких (виживає найсильніший) у процесі боротьби за виживання зміцнювались і створювали генетичну структуру наступної популяції.

У 1859 році *Ч. Дарвін* створив учення, відповідно до якого факторами еволюції є *спадковість*, *мінливість* і *природний добір*. Дарвін довів, що нові види організмів походять тільки від інших, попередніх видів. Відтворення ж живих форм, засноване на відтворенні типу й особливостей обміну речовин, є наслідком реалізації в клітині властивої їй генетичної інформації. У живій клітині інформація подана генетичними структурами, у яких відбита історія розвитку живого, включаючи саму генетичну інформацію. Цими структурами служать полімери нуклеїнових кислот. Отже, клітина несе в собі запис генетичної інформації, що є підсумком еволюційного розвитку виду й основу всієї його майбутньої еволюції [136].

ГА, використовуючи основні ідеї природного (еволюційного) розвитку, проте не завжди наслідують поведінку еволюційного процесу (наприклад, прискорення життєвих процесів, що відбуваються, здійснюється ГА шляхом зміщеного відтворення, заснованого на активному «життєзабезпеченні» *(survival of the fittest)*).

Хоча механізми, що стимулюють процес розвитку, до кінця не зрозумілі, можливості еволюції вивчено досить повно. Спочатку розвиток відбувається в хромосомах, генетичні модулі яких кодують структуру й можливості живих організмів. Специфічні можливості живого організму визначені хромосомами попереднього покоління й впливають на розвиток тільки хромосом нащадка. Оскільки еволюція обмежена хромосомами, можливості живих організмів, як видно, установлюються при народженні й відрізняються від попереднього покоління тільки тому, що в процесі еволюції змінюються батьківські хромосоми.

22.1. Кодування генетичної інформації

Природний хромосомний матеріал є лінійною послідовністю різних комбінацій таких чотирьох нуклеїдів: аденін, тимін, гуанін, цитозін.

Відповідно до цього в ГА використовується запис інформації у вигляді послідовностей, що складаються з ланцюжків символів, поданих у дво-, три- або чотирибуквеному алфавіті. У зв'язку з тим, що поведінка рядків бітів є досить наочною й доступною для розуміння, найпоширенішим є двійкове кодування, тобто подання інформації, що міститься у хромосомах, у вигляді бітових рядків (нулів й одиниць). Однак істотним недоліком простого двійкового коду є те, що його *відстань Хеммінга* між поданнями двох прямуючих одна за одною десяткових цифр не є постійною. У цьому сенсі більш ефективним є *двійковий відбитий код Ґрея*. У табл. 22.1 наведено двійковий код, код Ґрея й відповідні відстані Хеммінга.

Таблиця 22.1

Десяткове число	Двійковий код	Відстань Хеммінга	Код Ґрея
1	2	3	4
0	0000		0000
1	0001	1	0001
2	0010	2	0011
3	0011	1	0010
4	0100	3	0110
5	0101	1	0111
6	0110	2	0101
7	0111	1	0100
8	1000	4	1100
9	1001	1	1101

Закінчення табл. 22.1

1	2	3	4
10	1010	2	1111
11	1011	1	1110
12	1100	3	1010
13	1101	1	1011
14	1110	2	1001
15	1111	1	1000
16	10000	5	11000

Як випливає з таблиці, у коді Ґрея відстань Хеммінга завжди дорівнює 1, тоді як у простому двійковому коді вона може дорівнювати 4 при чотирирозрядному кодуванні (так, при зміні десяткового числа 7 на 8 всі чотири біти двійкового коду змінюються). Більша відстань Хеммінга призводить до зниження стійкості ГА. Справа в тому, що зміна одиночного біта внаслідок мутації викликатиме меншу наступну зміну, якщо відстань Хеммінга є малою.

Приклад 22.1. Розглянемо три рядки, що знаходяться поруч, таблиці 22.1, наприклад, які кодують десяткові цифри 10, 11 й 12.

Припустимо, фрагменти хромосом, що кодують число 11, належать деякому оптимальному вектору. У результаті мутації відбулися деякі зміни. Якщо краща особина з поточної популяції містить фрагмент хромосоми, що кодує число 10, то для отримання правильного розв'язку достатньо замінити біти в молодшому розряді. Якщо ж краща особина містить фрагмент хромосоми, що кодує число 12, то для отримання розв'язку необхідно проінвертувати біти вже в трьох розрядах. Для коду Ґрея обидві ситуації аналогічні й полягають у корекції тільки одного біта.

Результати деяких досліджень свідчать про те, що застосування коду Ґрея істотно поліпшує процес пошуку екстремуму певних функцій і забезпечує отримання розв'язку у всіх випадках не гірше, ніж двійкове кодування.

22.2. Генетичні оператори

Природний добір — процес, за допомогою якого природа відбирає ті хромосоми, які кодують кращі (щодо деякого критерію) харак-

теристики з метою відтворення цих характеристик у потомстві. При цьому ГА, декодуючи обрані хромосоми, забезпечують отримання оптимального розв'язку. Процес природного добору супроводжується створенням нових хромосом шляхом відтворення, схрещування, мутації й виживання хромосом із кращими характеристиками. Під час вибору відповідних критеріїв виживання якість наступних хромосом поліпшується.

Таким чином, еволюція відбувається через процес відтворення, що полягає у схрещуванні, мутації й інверсії (рекомбінації), реалізованих при моделюванні генетичних структур відповідно *операторами схрещування* (*кросовером*), *мутації* й *інверсії*. У найпростіший спосіб дію операторів можна продемонструвати на прикладах хромосом, поданих у двійковому коді.

Кросовер (кросинговер)

Кожна соматична клітина людини містить 46 хромосом, кількість яких є постійною і зберігається у всіх наступних поколіннях. Існує регулюючий механізм, *мейоз*, що дозволяє зберігати сталість об'єму генетичної інформації з поколінь організмів і складається із двох клітинних розподілів при одному синтезі ДНК, що безпосередньо бере участь у шерензі генетичної інформації. Складний процес обміну хромосом блоками генів отримав назву *кросинговера*, або *кросовера* (від англійського *crossover*, *crossingover* — перехрестя). При математичному моделюванні під кросовером (кросинговером) розуміють оператор, що формує хромосому нащадка, використовуючи для цього фрагменти батьківських хромосом.

Батьківські хромосоми		Нові хромосоми	Вибір
Генотип 1	10010|110	10010010	1001010
Генотип 2	01001|010	01001110	

Кросовер змішує хромосоми обох батьків, внаслідок чого виходять дві нові хромосоми, кожна з яких містить деякі характеристики батьків. При цьому батьківські хромосоми розбиваються в деякій точці, й одна частина однієї батьківської хромосоми замінюється на іншу частину іншої батьківської хромосоми.

Якщо одна з нових хромосом отримує 70 % характеристик від першого з батьків й 30 % від іншого, то друга нова хромосома

отримує 30 % характеристик з першого батьків й 70 % другого. Саме завдяки операції, реалізованої кросовером, особини популяції обмінюються між собою генетичною інформацією, різко збільшуючи тим самим комбінаційну спадкоємну мінливість організмів. Обидві результуючі хромосоми входять у *POOL*-гени (там постійно перебувають усі алелі), де вони замінюють погані або добрі хромосоми чи відкидаються. Якщо хромосома відкидається, то унікальні можливості (добрі або погані) губляться назавжди. Тільки шляхом мутації така хромосома може бути створена знову.

Мутація

Мутація — процес, при якому довільно змінюється одиночна компонента (біт) хромосоми.

Хромосома до мутації	Після мутації
10010110	10000110
↑	
місце прикладення оператора мутації	

Це відбувається в рідкісних випадках. Залежно від того, у якій компоненті хромосоми відбулася зміна, визначається значення віддаленості нащадка від батька. Мутація викликає різку зміну характеристик хромосоми й впливає на всі наступні покоління хромосом, що містять цю видозмінену компоненту. Якщо мутація призводить до зниження якості хромосоми, то ця хромосома буде відкинута й виключена з *POOL*-генів.

Інверсія порушує порядок проходження фрагментів хромосом

Хромосоми до інверсії	Після інверсії
10010110	10110100
↑	
місце прикладення оператора інверсії	

Як ми вже зазначали, кожна популяція має *POOL*-гени, що складаються з великої кількості хромосом, згенерованих у процесі природного добору.

Нові хромосоми постійно відтворюються в процесі схрещування, описаному вище, і хромосоми із кращими характеристиками зберігаються, а з гіршими — відкидаються.

22.3. Цільова функція

У ШНМ цільовою функцією є деяка опукла функція помилки, що мінімізується в процесі навчання по всьому набору навчання. У ГА як цільова функція розглядається «*придатність*» або «*пристосованість*», яка є деякою величиною, що визначає якість хромосоми, тобто що дозволяє робити висновок про те, наскільки певна хромосома краще або гірше інших.

Пристосованість характеризується функцією пристосованості, що має звичайно позитивні значення й вибирається залежно від конкретної задачі. Максимальне значення цієї функції відповідає оптимальному розв'язку, отримуваному за допомогою ГА, й означає той факт, що дана хромосома має більшу ймовірність її вибору для відтворення наступного покоління. Функції пристосованості можуть мати велику розмірність, бути недиференційованими, багатоекстремальними й мати розриви.

22.4. Реалізація ГА

Для реалізації ГА необхідно:

а) закодувати оптимізувальні змінні, наприклад, подати їх у вигляді бітових рядків;

б) призначити відповідну функцію пристосованості.

Генерування початкових значень (вихідної популяції з M генотипів) бітових рядків звичайно здійснюється випадковим чином, однак ефективність роботи ГА зростає, якщо початкові значення хромосом будуть пов'язані з характером розв'язуваної задачі.

Процес відтворення полягає в копіюванні індивідуальних рядків відповідно до пріоритетів, установлюваних обраною функцією пристосованості. При цьому копіювання означає, що кандидати з більш високими значеннями функції пристосованості мають більшу ймовірність впливу на наступну популяцію. Імовірність вибору i-го бітового рядка в поточній популяції визначається формулою

$$p_i = \frac{f_i}{\sum_{i=1}^{M} f_i}, \qquad (22.1)$$

де f_i — значення функції пристосованості i-го індивіда в популяції, що містить M генотипів.

Ця ймовірність є основою для вибору індивіда, що в подальшому бере участь у відтворенні.

Середня пристосованість популяції обчислюється як

$$\bar{f}_i = \frac{\sum_{i=1}^{M} f_i}{M}. \qquad (22.2)$$

Якщо при заміні старого покоління новим розмір популяції залишається постійним, очікувана кількість копіювання i-го рядка складає

$$m_i = Mp_i. \qquad (22.3)$$

З (22.3) видно, що кращі хромосоми будуть відтворені з більшою ймовірністю. Існує два способи вибору хромосом для відтворення:

а) *пропорційний*, коли кількість відтворення гена пропорційна функції пристосованості (*див.* формулу (22.3));

б) *ранговий*, при якому кожен ген відтворюється тільки один раз.

Перший спосіб використовувався Голландом. Істотним недоліком пропорційного способу є те, що він внаслідок копіювання тільки кращих хромосом іноді призводить до перекручування *POOL*-генів. Ранговий же спосіб збігається повільно (для прискорення збіжності запропоновано його модифікації, що допускають множинне відтворення одиночного гена).

Вилучення найменш придатних хромосом також можна зробити двома способами:

а) повною заміною всієї сукупності після кожного циклу;

б) зі збереженням старих і нових елементів з найбільш високими значеннями функції пристосованості.

Перший спосіб здатний прискорити збіжність ГА за рахунок більшого різноманіття *POOL*-генів. При другому способі, коли зберігаються кращі бітові рядки, оптимальний розв'язок може бути не досягнутий внаслідок того, що були передчасно вилучені гени, які є попередниками більш високорозвинених генів.

Загальну блок-схему ГА наведено на рис. 22.1.

Після M тактів роботи алгоритмів нова популяція виявляється сформованою. Оскільки в процесі її формування використовувалися індивіди з більш високими значеннями функції пристосованості, то середня пристосованість нової популяції буде вищою (або не нижче) попередньої.

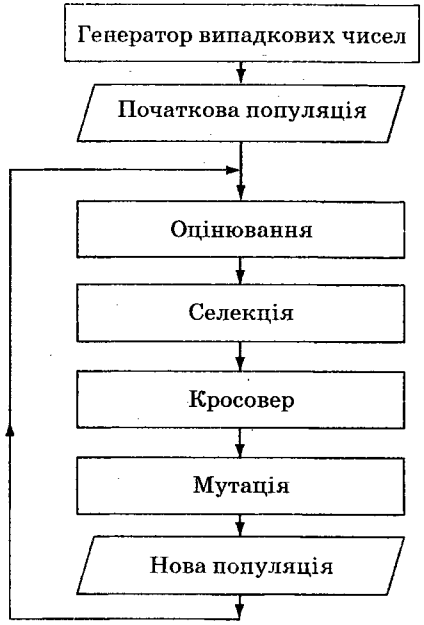


Рис. 22.1. Загальна блок-схема ГА

Це відповідає досягненню глобального екстремуму, коли бітові рядки будуть ідентичними, а вплив на них генетичних операторів не призведе до виходу із цього екстремуму. Так, кросовер, застосований до ідентичних бітових рядків, дасть такий самий бітовий рядок, а мутація й інверсія, вводячи деякі незначні збурювання в наступні популяції, не призведуть до отримання іншого розв'язку (рис. 22.2).

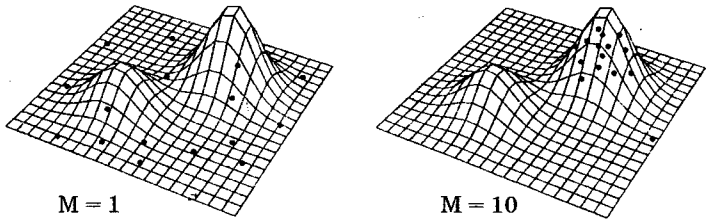


Рис. 22.2. Знаходження глобального екстремуму

22.5. Генетичні алгоритми й ШНМ

Хоча за словами одного із класиків ГА *К. Ді Янга* не слід розглядати ГА як алгоритми оптимізації, оскільки вони лише моделюють природні процеси, проте ці алгоритми розглядають саме як алгоритми оптимізації, які шукають розв'язок, використовуючи ідеї адаптації, тобто визначають елементи популяції, що доставляють екстремум обраному функціоналові (функції пристосованості).

У свою чергу, як ми вже бачили, алгоритми навчання ШНМ, підстроюючи синаптичні ваги, мінімізують кількість неправильних реакцій мережі, тобто також вирішують деяку оптимізаційну задачу. Як ГА, так й ШНМ, копіюючи процеси живої природи, намагаються забезпечити виживаність розглянутого об'єкта. Подібність розв'язуваних задач наводить на думку об'єднання цих підходів, наприклад шляхом конструювання ШНМ за допомогою ГА. Можливість знаходження за допомогою ГА глобального екстремуму робить ці алгоритми особливо привабливими.

Сьогодні ГА використовуються в ШНМ:

- для пошуку оптимальних синаптичних ваг (вагових коефіцієнтів) мереж прямого поширення невеликої розмірності;
- визначення оптимальної структури ШНМ для розв'язання конкретної задачі.

22.5.1. Визначення синаптичних ваг

Перевага ГА порівняно з іншими методами оптимізації, особливо з градієнтними, використовуваними для настроювання вагових коефіцієнтів, полягає в такому:

— ГА здатні досягати глобального екстремуму, не застрягаючи у локальних;

— використовуваний функціонал може бути недиференційованим;

— реалізація ГА можлива з використанням паралельних обчислень, що забезпечує можливість пошуку розв'язку одночасно у всьому просторі параметрів.

Використання ГА в ШНМ вимагає вирішення ряду питань, пов'язаних з розміром використовуваної популяції, вибором способу кодування параметрів мережі й необхідною точністю їх подання, вибором типу відтворення й критерію зупинника процесу. Зазвичай ГА кодують параметри ШНМ рядком (бітовим) або списком його реквізитів. Хоча для ефективної роботи ГА потрібна велика

кількість рядків (хромосом) або списків, що є великою кількістю наборів параметрів даної мережі, реалізація алгоритму ускладнень не викликає, оскільки використовує паралельні обчислення. Закодовані рядки поєднуються із застосуванням операторів схрещування (кросовера) і мутації для формування наступної популяції, що задовольняє обраній функції пристосованості. Функція пристосованості нерідко вибирається у вигляді промасштабованої інверсії мережевої помилки.

Використання ГА в ШНМ для визначення вагових коефіцієнтів забезпечує більш швидке влучення в окіл оптимального розв'язку, ніж пошук на основі алгоритму зворотного поширення помилки, однак у самому околі оптимуму властивості алгоритмів змінюються на протилежні. Це пояснюється тим, що збіжність ГА в точку оптимуму визначається оператором мутації.

У роботі [137] ГА використовувалися для керування алгоритмом зворотного поширення в мережах невеликої розмірності, що є модулями мережі більшої розмірності. Такий підхід забезпечив вищу швидкість навчання результуючої мережі порівняно з навчанням вихідної мережі.

Метод навчання, заснований на *моделі еволюції Ламарка* (*Lamarkian evolution*) і що є комбінацією нейромережевого й генетичного алгоритмів, містить періоди генетичної оптимізації між періодами навчання за алгоритмом зворотного поширення [138].

Метод навчання *Болдуїна* навпаки використовує алгоритм зворотного поширення для корекції значення функції пристосованості (*fitness*) хромосом. При цьому хромосоми, здатні навчатися за допомогою алгоритму зворотного поширення, вважаються кращими, тобто більш імовірними кандидатами для передачі своєї генетичної інформації наступній популяції.

До недоліків такого підходу відносять великий обсяг пам'яті, необхідний для підтримки життєздатної популяції, і складність його застосування до мереж великої розмірності.

22.5.2. Оптимізація топології ШНМ

Як ми вже зазначали вище, питання вибору топології ШНМ є надзвичайно важливим, оскільки від цього залежить і точність, і складність розв'язання поставленої задачі. Сьогодні не існує єдиної або хоча б достатньо ефективної методології проектування структури мереж, тому вибір топології мережі є емпіричним, заснованим на досвіді проектувальника.

Формулювання задачі пошуку оптимальної топології ШНМ для розв'язання конкретної прикладної проблеми як задачі дискретної оптимізації уможливлюють застосування для її розв'язання ГА. Це є тим більш доцільним через велику апріорну невизначеність щодо топології ШНМ, якій має відповідати більша й первинно різноманітна популяція, що відбиває відповідно більшу розмірність і різнорідність вихідного простору.

Використовуючи двійкове кодування хромосом (бітові рядки), *С. Гарп і Т. Самад* [139] розширили поняття хромосом і бітових рядків до поняття об'єкта деревоподібної структури, кожен ген якої характеризується адресою, кількістю нейронів, ваговими параметрами й т. д. Для керування ГА й набором з'єднаних нейронів використана схема «синьки» (*blueprint*), у якій мережеві характеристики визначені як структури даних. *Blueprint* визначає вид реальної мережі, а ШНМ навчається за алгоритмом, параметри якого уточнюються в *blueprint*. Навчання ШНМ згодом тестується для оцінки її стійкості при виключенні з неї деяких модулів або при зміні значень отриманих вагових коефіцієнтів. Потім визначається її пристосованість, що може бути задана довільною функцією, наприклад точністю й швидкістю навчання, кількістю вузлів, максимальною кількістю виходів, що виходять із вузла, і т. д.

Після цього формується така популяція мережі в *blueprint* з використанням генетичних операторів схрещування, мутації й інверсії, що полягає в створенні нового *blueprint* на основі двох батьківських *blueprint*. Отже, генетичний підхід тут використовується на двох рівнях: на мікрорівні — до бітових рядків, на макрорівні — до *blueprint*. Даний підхід дозволяє розроблювачу істотно розширити область можливих варіантів розв'язань.

Окрім зазначених робіт, є достатньо багато літератури, у якій розглядаються різні методи оптимізації топології мереж з використанням ГА. Багатообіцяючим є також застосування методів *генетичного програмування* [140–142].

Контрольні запитання

1. У який спосіб відбувається кодування інформації в ГА?
2. Назвіть генетичні оператори й поясніть їхнє призначення.
3. Які функції вибираються в якості цільових?
4. Що являє собою процес відтворення?
5. Для розв'язання яких задач ШНМ використовуються ГА?
6. Як здійснюється оптимізація топології ШНМ за допомогою ГА?

23. ПРИКЛАДИ ЗАСТОСУВАННЯ ШНМ

Як ми вже зазначали, ШНМ використовуються для розв'язання найрізноманітніших практичних задач. Тому описати приклади застосування мереж у всіх галузях не є можливим. Ми розглянемо лише кілька прикладів ефективного застосування різних ШНМ.

23.1. Синтезатор мови — NETtalk

В історичному огляді зазначалося, що дана система першою продемонструвала ефективність практичного застосування ШНМ [26]. Архітектуру *NETtalk* зображено на рис. 23.1. Система складається із двох модулів — багатошарового персептрона, на виході якого з'являються сигнали з 29 англійських букв, і промислового акустичного синтезатора, що перетворює вихідні сигнали персептрона у звукові.

У зв'язку з тим, що в англійській мові, як й у більшості інших, зв'язок між буквами й фонемами не є точним, а залежить від контексту, що впливає на артикуляцію тієї чи іншої букви, у системі використовується деяке вікно 3 + 1 + 3, що враховує вплив на розглянуту букву тексту трьох попередніх і трьох наступних букв, тобто кожна буква алфавіту подана 7 двійковими розрядами. Весь же текст подається у вхідній матриці, що складається з 7 × 29 бітових розрядів. Як видно з рис. 23.1, кожен стовпець цієї матриці відповідає певній букві, а весь текст — набору відповідних стовпців. Прихований і вихідний шар персептрона утворені відповідно 80 й 26 нейронами. Кожен нейрон вихідного шару видає відповідну фонему на акустичний синтезатор.

Навчання мережі полягає в поданні тексту, покроковому аналізі 3 + 1 + 3 вікна й формуванні правильних фонем. Після цього система набуває можливість утворювати послідовності фонем довільного тексту, тобто озвучувати будь-який текст.

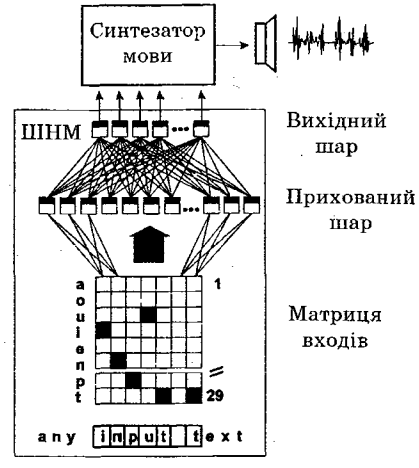


Рис. 23.1. Архітектура мережі *NETtalk*

Різновидом *NETtalk* є система *NETface*, що моделює зміну м'язів обличчя (міміку) під час розмови й призначена для комп'ютерної анімації [143].

23.2. Система автоматичного керування автомобілем

Сьогодні існує досить багато експериментальних систем автоматичного керування автомобілем [144, 145]. Одна з таких систем, *ALVINN* (*Autonomous Land Vehicle In a Neural Network*), побудована за допомогою ШНМ і призначена для керування вантажним автомобілем [145]. Автомобіль має численні датчики й відеокамери, лазерний скануючий прилад, автоматичну навігаційну систему, велику кількість мікроконтролерів.

Для орієнтації на місцевості використовується повнозв'язна тришарова мережа прямого поширення, що обробляє колірні сигнали, які надходять від відеокамер, вони утворюють поле 30 × 32 (або 45 × 48) пікселів. Прихований шар утворюють 9 нейронів, а вихідний — 45. Під час руху автомобіля по різних вулицях мережа використовує різні вагові перестроювані коефіцієнти. Укрупнену архітектуру цієї ШНМ наведено на рис. 23.2.

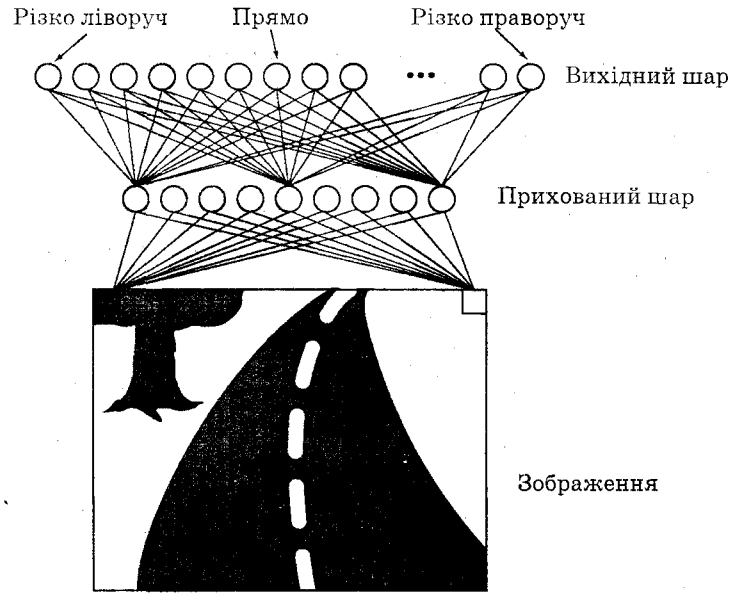


Рис. 23.2. Багатошарова ШНМ
для автоматичного керування автомобілем

Використовуване під час руху автомобіля зображення з метою підвищення контрастності дороги зазделегідь обробляється й передається на входи мережі. ШНМ розраховує шлях, і вихідні сигнали мережі є командами для необхідного повороту кермового колеса. У системі використовується 45 градацій повороту руля від «різко праворуч» до «різко ліворуч». Під час руху автомобіля зі швидкістю до 90 км/год і більше вихідні сигнали мережі обчислюються 15 разів за секунду, тобто керування автомобілем відбувається в реальному часі.

Для навчання мережі їй подаються оброблені зображення відповідних доріг, отриманих під час руху автомобіля, керованого водієм. Таким чином, множина навчальних пар складається з «миттєвого знімка» ділянки дороги (вектор вхідних сигналів) і відповідного розміщення кермового механізму (бажаний вихідний сигнал). Для кожної ділянки дороги формується зображення, отримане з різних точок, а результуюче зображення трансформується таким чином, щоб можна було б моделювати помилкові керуючі сигнали. Це дозволяє надалі уникнути помилкових розв'язків й у результаті навчання механізм керування має відповідати ідеальному руху автомобіля, керованого досвідченим водієм. Для розпізнавання різних перешкод на шляху руху (дерев, будов і т. д.) використовуються тривимірні зображення, отримані за допомогою лазерів. Крім того, здійснювалося тестування системи для різних типів доріг і різних погодних умов.

Автомобіль, забезпечений описаною системою, успішно проїхав в автоматичному режимі понад 150 км зі швидкістю 90 км/год і більше.

23.3. Розв'язання задачі про комівояжера

Задача про комівояжера полягає в знаходженні його найкоротшого маршруту, що проходить через задану кількість міст, причому в кожному місті комівояжер має побувати не більше одного разу. Зазначимо, що за наявності N міст існує порядку $N!$ різних маршрутів, серед яких необхідно визначити найкоротший. Як зазначалося у вступі, дана задача відноситься до класу NP-повних задач (NP-повнота означає, що не існує детермінованого алгоритму розв'язання задачі розмірності N за час $p(N)$, де $p(N)$ — поліном), оптимальне вирішення яких визначається шляхом повного перебору варіантів.

Під час розв'язання даної задачі за допомогою ШНМ необхідно враховувати таке. Будучи добрим апроксиматором, мережа може забезпечити не оптимальний (точний) розв'язок, а субоптимальний (прийнятний).

Розв'язання задачі про комівояжера за допомогою мережі Гопфілда засновано на наступному твердженні.

Твердження. Нехай є дві мережі Гопфілда з матрицями ваг $W^{(1)}$ та $W^{(2)}$ і порогами $\theta^{(1)}$ і $\theta^{(2)}$, енергія кожної з яких визначається виразом

$$E_K = -\frac{1}{2}\sum_i\sum_{j\neq i} w_{ij}y_iy_j + \sum_i \theta_i y_i \quad (k = 1,2). \qquad (23.1)$$

Тоді для дійсних чисел A та B величина

$$E = AE_1 + BE_2 \qquad (23.2)$$

є енергією мережі Гопфілда з матрицею ваг $W = AW^{(1)} + BW^{(2)}$ і порогами $A\theta^{(1)} + B\theta^{(2)}$.

Доведення цього твердження отримується підстановкою (23.2) в (23.1)

$$AE_1 + BE_2 = -\frac{1}{2}\sum_i\sum_{j\neq i}(Aw_{ij}^{(1)} + Bw_{ij}^{(2)})y_i y_j + \sum_i (A\theta_i^{(1)} + B\theta_i^{(2)})y_i.$$

Для розв'язання цієї задачі *Гопфілд* і *Танк* [25] описували шлях комівояжера за допомогою спеціальної матриці, що складається з нулів й одиниць, рядки якої зображували міста, а стовпці — послідовність їхнього відвідування. Таким чином, r-й маршрут подавався у вигляді матриці $M^{(r)}$ з елементами

$$m_{s,i}^{(r)} = \begin{cases} 1, & \text{якщо місто } s \text{ відвідується } i\text{-м;} \\ 0 & \text{в іншому випадку.} \end{cases} \qquad (23.3)$$

Наприклад, маршрут B→F→D→C→A→E відповідає матриці

Порядок відвідування / Місто	1	2	3	4	5	6
A	0	0	0	0	1	0
B	1	0	0	0	0	0
C	0	0	0	1	0	0
D	0	0	1	0	0	0
E	0	0	0	0	0	1
F	0	1	0	0	0	0

У такий спосіб можна побудувати множину $N \times N$ матриць, кожна з яких описує певний маршрут.

Позначимо відстань між містами s й q як $d_{s,q}$. Тоді довжина r-го маршруту може бути обчислена так:

$$E_1 = \frac{1}{2}\sum_{s=1}^{N}\sum_{q=1}^{N}\sum_{i=1}^{N} d_{s,q}\; m_{s,i}^{(r)}(m_{q,i+1}^{(r)} + m_{q,i-1}^{(r)}). \qquad (23.4)$$

Оскільки елементами матриці $M^{(r)}$ є нулі або одиниці, $m_{s,i}^{(r)}(m_{q,i+1}^{(r)} + m_{q,i-1}^{(r)}) = 1$, якщо місто s буде відвідане на r-му маршруті i-м, а місто q або буде попереднім місту s, або наступним за ним. Множник $\frac{1}{2}$ вводиться в (23.4) внаслідок того, що відстань між двома послідовно відвідуваними містами обчислюється в цьому виразі двічі.

Задача полягає у визначенні таких значень елементів матриці $M^{(r)}$, які забезпечували б мінімум (23.4). Вираз (23.4) може бути переписаний в такий спосіб:

$$E_1 = \frac{1}{2}\sum_s\sum_q\sum_i\sum_j d_{s,q}\;(\delta_{i+1,j} + \delta_{i-1,j})m_{s,i}^{(r)}\; m_{q,j}^{(r)}, \qquad (23.5)$$

де $\delta_{i,j}$ — символ Кронекера, тобто

$$\delta_{i,j} = \begin{cases} 1 & \text{при } i = j; \\ 0 & \text{при } i \neq j. \end{cases}$$

Порівнюючи (23.5) з (6.16), бачимо, що вираз (23.5) є енергією мережі Гопфілда, ваги якої рівні $d_{s,q}(\delta_{i+1,j} + \delta_{i-1,j})$, пороги $\theta_{s,I} = 0$, а сигналами є $m_{s,i}^{(r)}$ й $m_{q,j}^{(r)}$. Таким чином, при заданні довільної конфігурації мережа досягатиме деякого сталого стану, що відповідає локальному мінімуму функціонала (23.5). А оскільки всі елементи $m_{s,i}^{(r)} \in \{0,1\}$, $(s,i = 1,2,\ldots,N)$, то очевидно, що глобальний мінімум буде досягнутий при $m_{s,i}^{(r)} = 0$. Однак це означає, що комівояжер не повинен відвідати жодне місто. Щоб задача мала сенс, необхідно модифікувати функціонал (23.5), врахувавши наявні обмеження.

До таких обмежень належать:

— вимога відвідування кожного міста не більше одного разу

$$\sum_{s=1}^{N}\sum_{i=1}^{N}\sum_{j=1}^{N} m_{s,i}^{(r)} m_{s,j}^{(r)} = 0; \qquad (23.6)$$

— неможливість одночасного відвідування двох міст

$$\sum_{s=1}^{N}\sum_{i=1}^{N}\sum_{\substack{q=1 \\ q\neq s}}^{N} m_{s,i}^{(r)} m_{q,i}^{(r)} = 0; \qquad (23.7)$$

— вимога відвідування всіх N міст

$$\left(\sum_{s=1}^{N}\sum_{i=1}^{N} m_{s,i}^{(r)} - N\right)^2 = 0. \qquad (23.8)$$

За аналогією з (23.4) такі вимоги можуть бути подані у вигляді деяких енергетичних функцій відповідних мереж Гопфілда. Так, умова (23.6) може бути записана у вигляді:

$$E_2 = \frac{1}{2}\sum_{s=1}^{N}\sum_{q=1}^{N}\sum_{i=1}^{N}\sum_{j=1}^{N}\left[\delta_{s,q}(1-\delta_{i,j})\right] m_{s,i}^{(r)} m_{q,i}^{(r)} = 0. \qquad (23.9)$$

Дійсно, оскільки $m_{s,i}^{(r)} \in \{0,1\}$, $E_2 \geq 0$, виконання умови (23.6) еквівалентно мінімізації (23.9). Вираз же (23.9) є енергією мережі Гопфілда, заданої вагами $[-\delta_{s,q}(1-\delta_{i,j})]$, порогами $\theta_{s,i}=0$ і вхідними сигналами $m_{s,i}^{(r)}$ й $m_{q,j}^{(r)}$.

Аналогічно маємо для умови (23.7)

$$E_3 = \frac{1}{2}\sum_{s=1}^{N}\sum_{q=1}^{N}\sum_{i=1}^{N}\sum_{j=1}^{N}\delta_{i,j}(1-\delta_{s,q})\, m_{s,i}^{(r)} m_{q,j}^{(r)}. \qquad (23.10)$$

У цьому випадку ваги й пороги Гопфілда рівні відповідно $\left[-\delta_{i,j}\left(1-\delta_{s,q}\right)\right]$, $\theta_{s,i}=0$. Нарешті, використовуючи для зручності в (23.8) коефіцієнт $\frac{1}{2}$, отримуємо

$$E_4 = \frac{1}{2}\left(\sum_{s=1}^{N}\sum_{i=1}^{N} m_{s,i}^{(r)} - N\right)^2 =$$

$$= \frac{1}{2}\left[\left(\sum_{s=1}^{N}\sum_{i=1}^{N} m_{s,i}^{(r)}\right)\left(\sum_{q=1}^{N}\sum_{j=1}^{N} m_{q,j}^{(r)}\right) - 2N\left(\sum_{s=1}^{N}\sum_{i=1}^{N} m_{s,i}^{(r)}\right) + N^2\right] = \qquad (23.11)$$

$$= \frac{1}{2}\sum_{s=1}^{N}\sum_{q=1}^{N}\sum_{i=1}^{N}\sum_{j=1}^{N} m_{s,i}^{(r)} m_{q,j}^{(r)} - \sum_{s=1}^{N}\sum_{i=1}^{N} N m_{s,i}^{(r)} + \frac{N^2}{2}.$$

Досягнення мінімуму (23.11) еквівалентно виконанню умови (23.8). Оскільки при мінімізації цього виразу константою $\frac{N^2}{2}$ можна знехтувати, функціонал набуває вигляду

$$E_1 = \frac{1}{2}\sum_{s=1}^{N}\sum_{q=1}^{N}\sum_{i=1}^{N}\sum_{j=1}^{N} m_{s,i}^{(r)} m_{q,j}^{(r)} - \sum_{s=1}^{N}\sum_{i=1}^{N} N m_{s,i}^{(r)}, \qquad (23.12)$$

що відповідає енергетичному функціоналу мережі Гопфілда, ваги якої рівні -1, а пороги $\theta_{s,i} = -N$ $(s,i = 1,2,\ldots,N)$.

Нехай задані довільні дійсні числа A, B, C, D. Тоді, скориставшись (23.1), (23.2), запишемо повну енергію модифікованої мережі Гопфілда

$$E = AE_1 + BE_2 + CE_3 + DE_4, \qquad (23.13)$$

яка може бути використана для розв'язання поставленої задачі.

Мінімум функції (23.13) досягається, коли всі складові мають мінімальні значення.

Враховуючи обмеження на значення ваг для відповідних мереж, одержуємо

$$w_{si,qj} = -B\delta_{s,q}(1-\delta_{i,j}) - C\delta_{i,j}(1-\delta_{s,q}) - D - Ad_{s,q}(\delta_{i+1,j} + \delta_{i-1,j}). \qquad (23.14)$$

Оскільки отримуваний розв'язок залежить від констант A, B, C, D, виникає проблема їхнього вибору. При вдалому виборі констант можна отримати розв'язок, що мало відрізняється від оптимального. У роботі [25] використовувалися $A = B = C = 500$ й $D = 200$. При цьому для задачі з 10 містами мережа Гопфілда досягала стійкого стану в 16 випадках з 20, а у 8 випадках отримано розв'язки, що відрізняються від оптимального менш ніж на 3 %.

23.4. Ідентифікація нелінійних динамічних об'єктів

Необхідним етапом розв'язання задачі керування, прогнозування поведінки, моделювання нелінійних динамічних систем є одержання їх адекватних математичних моделей, що базується, як правило, на теоретичному й експериментальному аналізі властивостей цих систем. Теоретичний аналіз, заснований на вивченні фізичних і хімічних процесів, що відбуваються в системі, дозволяє отримати математичний опис у вигляді, наприклад, диференціальних рівнянь. При експериментальному аналізі на основі спостережень вхідних і вихідних сигналів системи отримують або її параметричну, або непараметричну модель. Найбільшого поширення набули параметричні моделі, що вимагають розв'язання задач структурної й параметричної ідентифікації й що використовують обмежену кількість параметрів. Незважаючи на величезну кількість робіт, різноманіття видів нелінійності не дозволяє створити єдину теорію ідентифікації нелінійних систем. Застосовуваний найчастіше класичний підхід заснований на апроксимації нелінійності, наприклад рядами Вольтерра, Гаммерштейна, Вінера, поліномами Колмогорова — Габора й ін., однак сфера застосування таких моделей обмежена. Крім того, додаткові труднощі одержання адекватного математичного опису обумовлює наявність у реальних сигналах завад, що вимагає попередньої фільтрації сигналів.

Розглянемо задачу ідентифікації нелінійного динамічного об'єкта, описуваного NARMAX-моделлю

$$\tilde{y}(k) = f[y(k-1), ..., y(k-m), u(k-1), ..., u(k-n), k] + \xi(k), \quad (23.15)$$

де $\tilde{y}(i)$, $u(i)$ — вихідний і вхідний сигнали об'єкта в момент часу i відповідно; m, n — порядки запізнювання за вихідним й вхідним каналами відповідно; $f(\cdot)$ — невідома нелінійна функція; ξ — завада вимірювання вихідного сигналу.

Введемо вектор узагальненого вхідного сигналу розмірності $(n+m)\times 1$ так, що:

$$x(k) = [\tilde{y}(k-1), \tilde{y}(k-2), ..., \tilde{y}(k-m), u(k-1), u(k-2), ..., u(k-n)]. \quad (23.16)$$

Тоді рівняння (23.15) може бути записане у такий спосіб:

$$\tilde{y}(k) = f[x(k), k] + \xi(k). \quad (23.17)$$

Задача ідентифікації полягає в оцінюванні функції $f(\cdot)$ за вимірюваннями вхідних $u(k)$ і вихідних $\tilde{y}(k)$ змінних.

Схему ідентифікації нелінійного об'єкта (23.15) на основі ШНМ наведено на рис. 23.3.

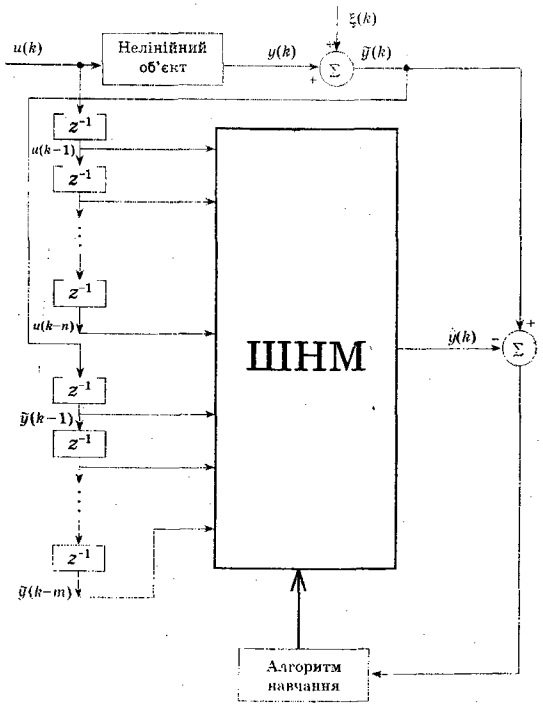


Рис. 23.3. Схема ідентифікації

Найбільшого застосування для розв'язання задачі ідентифікації нелінійних динамічних об'єктів (23.15) сьогодні отримали багатошаровий персептрон (БП) (рис. 23.4) і радіально-базисні мережі (РБМ) (рис. 23.5), що використовують апроксимацію нелінійного оператора $f(\cdot)$ нейронною мережею.

Рівняння БП має вигляд

$$\hat{y}(k) = f^q\left[\left(w^q\right)^T f^{q-1}\left[\left(w^{q-1}\right)^T f^{q-2}\left[...f^1\left[\left(w^1\right)^T\right]x(k)\right]\right]\right], \quad (23.18)$$

де q — кількість шарів у мережі; w^i — вектор вагових параметрів нейронів i-го шару мережі; $f^i[\cdot]$ — функція активації i-го шару.

Модель у вигляді РБМ, що використовує для апроксимації $f(\cdot)$ деякі базисні функції $\Phi_i(x)$, може бути подана у такий спосіб:

$$\hat{y}(k) = \sum_{i=0}^{N} c_i \Phi_i(x), \quad (23.19)$$

де c_i — вагові коефіцієнти, що підлягають визначенню.

Якщо як базисну функцію $\Phi_i(x)$ вибрати

$$\Phi_i(x) = \exp\left\{-\frac{\|x-\mu_i\|^2}{\sigma_i^2}\right\}, \quad (23.20)$$

де μ_i, σ_i — центри й радіуси базисних функцій відповідно; $\|\cdot\|$ — евклідова норма, то визначенню підлягає вектор

$$w(k) = (c_0(k), c_1(k), \mu_1^T(k), \sigma_1(k), ..., c_N(k), \mu_N^T(k), \sigma_N(k))^T,$$

який містить усі невідомі параметри мережі (звичайно приймають $\Phi_0(x) = 1$).

Завдання ідентифікації полягає в навчанні мережі, тобто в такому настроюванні її параметрів, які б забезпечували мінімум функціоналу

$$J = e^2(k) = M\left\{\left[\tilde{y}(k) - \hat{y}(k)\right]^2\right\}. \quad (23.21)$$

Навчання мережі здійснюється за допомогою будь-якого методу мінімізації функціонала (23.21).

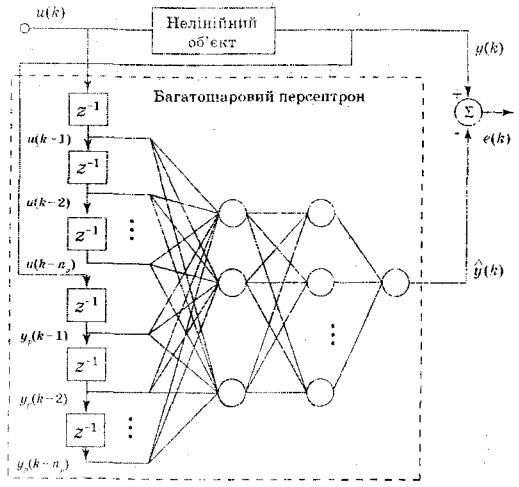


Рис. 23.4. Ідентифікація нелінійного об'єкта за допомогою багатошарового персептрона

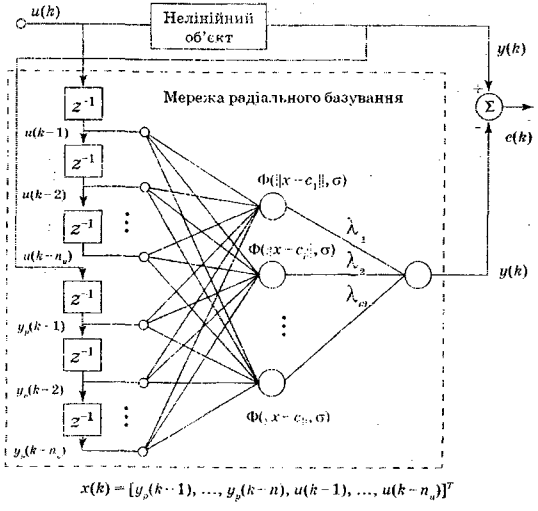


Рис. 23.5. Ідентифікація нелінійного об'єкта з допомогою РБМ

На рис. 23.6–23.11 наведено результати ідентифікації нелінійного динамічного об'єкта, описуваного рівнянням

$$y(k) = 0{,}725\beta(k)\sin\left(\frac{16u(k-1)+8y(k-1)}{\beta(k)\left(3+4u^2(k-1)+4y^2(k-1)\right)}\right)+ \\ +0{,}2u(k-1)+0{,}2y(k-1), \qquad (23.22)$$

де $\beta(k)$ — змінюваний у часі параметр, що задає ступінь нестаціонарності об'єкта.

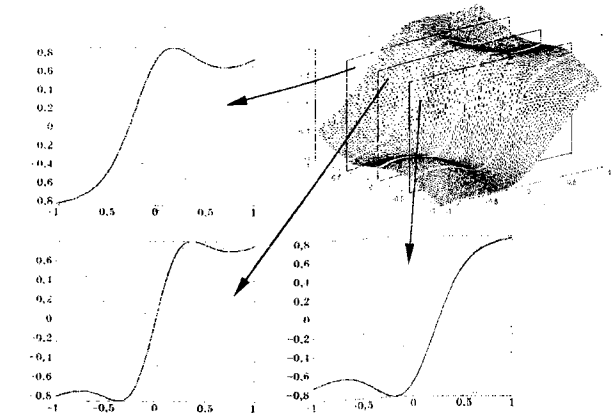
Вхідний сигнал $u(k)$ є випадковою стаціонарною послідовністю із рівномірним законом розподілу в інтервалі $[-1, 1]$. Навчальний набір містив 5000 навчальних пар. Необхідна точність задавалася на рівні $\varepsilon = 0{,}0001$. Під час моделювання БП ініціалізація ваг і зміщень здійснювалася відповідно до правила, при якому кожна вага рівномірно розподілена в діапазоні $[-\alpha, \alpha]$, де α задається формулою

$$\alpha(w_{ij}^{lm}) = \frac{1}{N^m + 1}.$$

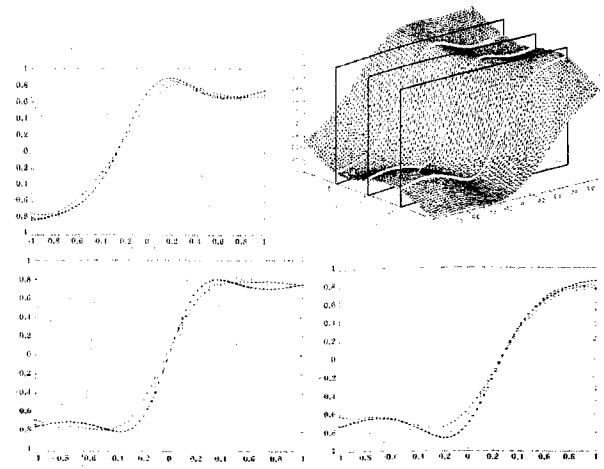
На рис. 23.6, *а* зображено саму поверхню, описувану заданим рівнянням при $\beta(k) = 1$, а також її перетини. Заданої точності розв'язання БП із архітектурою 5-5-1 досяг за 34 ітерації. Посилення обчислювальної потужності БП до архітектури 20-20-1 призвело до скорочення кількості ітерацій, необхідних для досягнення заданої точності, але при цьому істотно зріс час навчання.

Рис. 23.6, *б* відображає результати ідентифікації об'єкта (23.22) за допомогою РБМ. На цьому рисунку позначені вихідні сигнали об'єкта суцільною лінією, вихідні сигнали мережі, навченої за алгоритмом РМНК — кружками, алгоритмом Уідроу — Гоффа — хрестиками, проекційним п'ятикроковим алгоритмом — зірочками. Кількість нейронів при цьому становило відповідно 20, 24 й 22.

На рис. 23.7 наведено результати ідентифікації об'єкта (23.22) за наявності на його виході випадкової завади $\xi(k)$, що має нормальний закон розподілу в інтервалі $[-0{,}25; 0{,}25]$. Для розв'язання цієї задачі за допомогою БП використовувалася та сама структура БП мережі 5-5-1. Як видно з рисунка, мережа успішно впоралася з поставленою задачею завдяки своїм високим фільтруючим властивостям. На перетинах суцільною жирною лінією зображено реакцію об'єкта, а кружечками — вихід мережі за відсутності завади, реакція об'єкта й мережі за наявності шуму показані суцільною тонкою лінією й хрестиками відповідно.



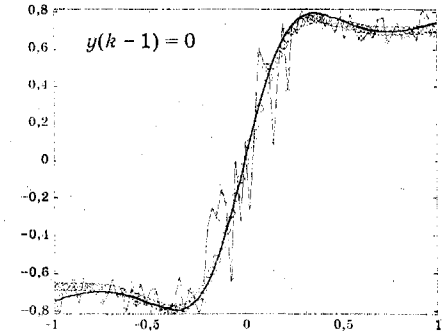
а



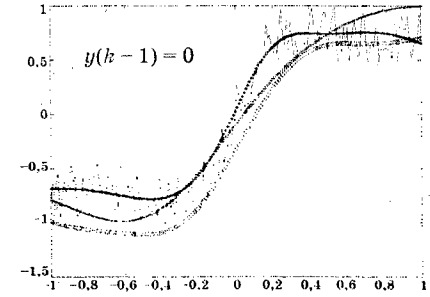
б

Рис. 23.6. Результати ідентифікації об'єкта (23.22) при $\beta(k) = 1$

При ідентифікації за допомогою РБМ кількість нейронів збільшилася й для мережі, що навчається за РМНК, склала 43, за алгоритмом Качмажа — 46, п'ятикроковим алгоритмом — 49. Хоча, як видно з рисунка, всі досліджувані алгоритми забезпечують розв'язання задач ідентифікації, найбільш точно відтворюють властивості об'єкта РБМ, що навчається за проекційним п'ятикроковим алгоритмом.



а



б

Рис. 23.7. Результати ідентифікації об'єкта (23.22) за наявності випадкової завади

На рис. 23.8 наведено результати ідентифікації нестаціонарного об'єкта, у якого після кожних 1250 пар навчання відбувалося зменшення параметра $\beta(k)$ у такий спосіб:

$$\beta(k) = \begin{cases} 1 & \text{при } k \in [0 \div 1250]; \\ 0{,}8 & \text{при } k \in [1251 \div 2500]; \\ 0{,}7 & \text{при } k \in [2501 \div 3750]; \\ 0{,}6 & \text{при } k \in [3751 \div 5000]. \end{cases}$$

Характер нестаціонарності об'єкта (23.22) при зміні параметра $\beta(k)$ наведено на рис. 23.9.

Після 5000 кроків навчання кількість нейронів для РБМ, що навчається за алгоритмом РМНК, виявилася рівною 26, за алгоритмом Уідроу — Хоффа — 31, за проекційним п'ятикроковим алгоритмом — 23.

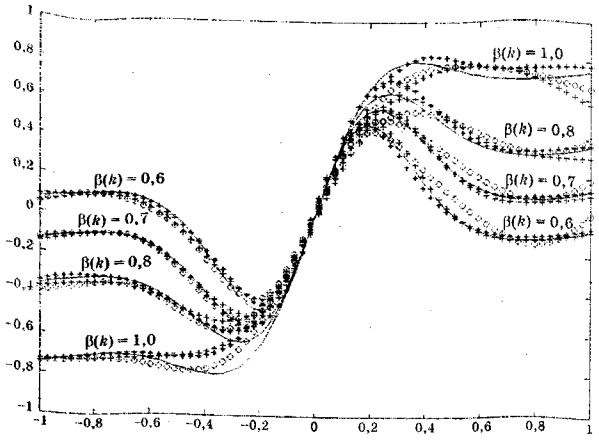


Рис. 23.8. Результати ідентифікації нестаціонарного об'єкта

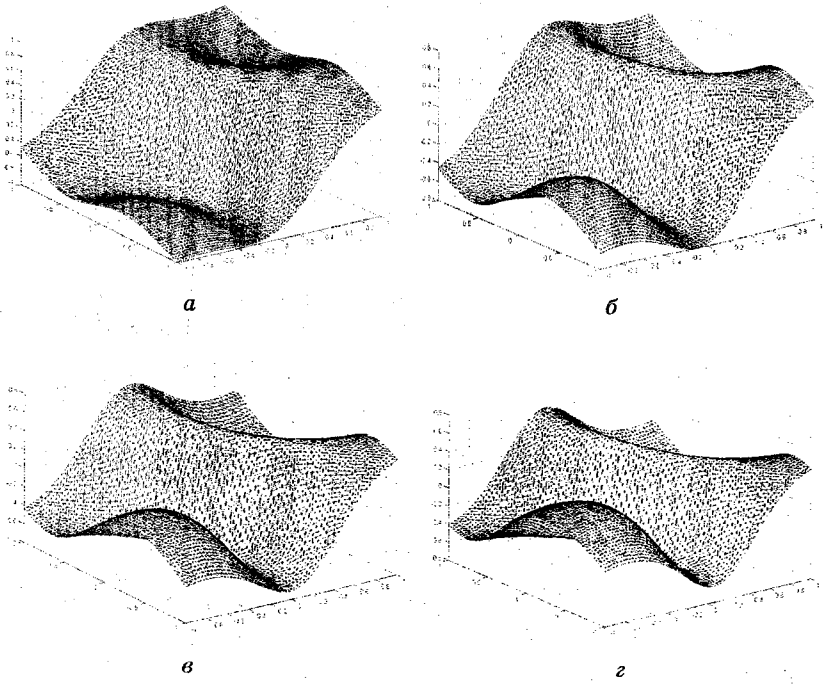


Рис. 23.9. Вплив параметра $\beta(k)$ на властивості об'єкта:
а — $\beta(k) = 1$; *б* — $\beta(k) = 0{,}8$; *в* — $\beta(k) = 0{,}7$; *г* — $\beta(k) = 0{,}6$

23.5. Нейрокерування нелінійними об'єктами

Розв'язання задачі керування реальними об'єктами засновано на використанні його математичної моделі, вид і складність якої визначають складність алгоритму керування. У ряді випадків виправдовує себе заміна нелінійної моделі лінійною й знаходження параметрів регулятора, які б забезпечували найкраще керування в деякій компромісній точці. Для оптимізації роботи об'єкта в декількох точках необхідна корекція параметрів регулятора відповідно до зміни робочих умов. У зв'язку з цим доцільною є розробка систем керування на основі адаптивного підходу в поєданні з методами теорії ШНМ, що дозволить, з одного боку, побудувати досить прості нейромережеві моделі, а з іншого — адаптивно корегувати параметри як цих моделей, так і регулятора відповідно до умов, що змінюються

Структуру побудованої в такий спосіб нейромережевої системи адаптивного керування наведено на рис. 23.10.

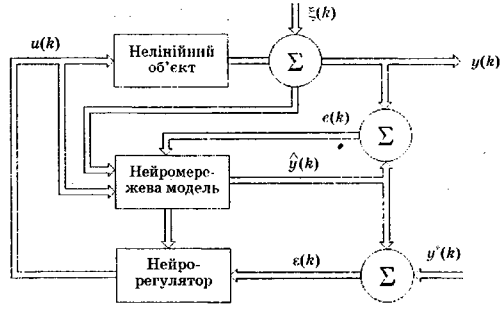


Рис. 23.10. Структура нейромережевої системи адаптивного керування

Найчастіше для розв'язання даних задач використовується БП і РБМ.

Розглянемо багатовимірний нелінійний динамічний об'єкт, описуваний рівнянням

$$y(k) = f[y(k-1),..., y(k-m), u(k-1), ..., u(k-n)] + \xi(k), \quad (23.23)$$

де $\boldsymbol{y}(i) = (y_1(i),...,y_{N_y}(i))^T$; $\boldsymbol{u}(i) = (u_1(i), ..., u_{N_u}(i))^T$ — вектори вихідних і керуючих сигналів розмірностей $N_y \times 1$ и $N_u \times 1$ відповідно; $f(\cdot) = (f_1(\cdot), f_2(\cdot), ..., f_{N_y}(\cdot))^T$ — невідома нелінійна векторна функція; $\xi(k) = [\xi_1(k), ..., \xi_{N_y}(k)]^T$ — вектор завад.

Позначивши вектор узагальненого сигналу, що надходить на вхід моделі, як

$$\boldsymbol{x}(k) = [y^T(k-1),..., y^T(k-m), u^T(k-1), ..., u^T(k-n)]^T, \quad (23.24)$$

перепишемо рівняння (23.23) у вигляді

$$\boldsymbol{y}(k) = f(\boldsymbol{x}(k)) + \xi(k). \quad (23.25)$$

Нейромережева модель об'єкта (23.25) має вигляд (23.18).

Навчання мережі, до якого зводиться завдання ідентифікації, полягає в настроюванні її параметрів на основі порівняння вихідних сигналів об'єкта $\boldsymbol{y}(k)$ і моделі $\hat{y}(k)$ та в мінімізації звичайно квадратичного функціонала помилки (23.21).

Після завершення процесу навчання мережа використовується для реалізації алгоритму керування.

Задача керування полягає у визначенні значень $\boldsymbol{u}(k)$, що забезпечують мінімум функціоналу

$$J(k) = \|\varepsilon(k)\|^2 = \frac{1}{2}\|y^*(k) - \hat{y}(k)\|^2, \quad (23.26)$$

де $\boldsymbol{y}^*(k)$ — необхідні значення вектора вихідних сигналів.

Як й у випадку ідентифікації, для знаходження мінімуму функціонала (23.26) можуть бути використані різні рекурентні алгоритми, зокрема, градієнти вигляду

$$\boldsymbol{u}(k) = \boldsymbol{u}(k-1) + \gamma(k)(\nabla_u \varepsilon(k)), \quad (23.27)$$

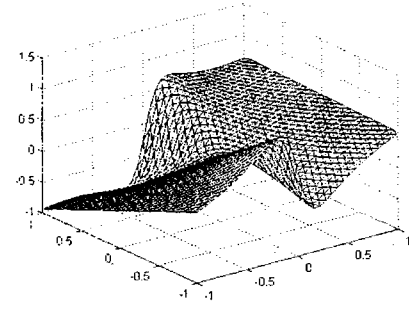
де $\nabla_u \varepsilon(k) = \dfrac{\partial \varepsilon(k)}{\partial \boldsymbol{u}(k)}$; $\gamma(k) > 0$.

На рис. 23.11–23.13 наведено результати керування нелінійним стаціонарним об'єктом, описуваним рівняннями

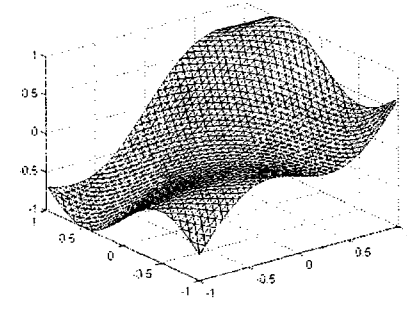
$$y_1(k) = \frac{15u_1(k-1)y_2(k-1)}{2+50[u_1(k-1)]^2} + 0{,}5u_1(k-1) - 0{,}25y_2(k-1) + 0{,}1;$$
$$y_2(k) = \frac{\sin(\pi u_2(k-1)y_1(k-1)) + 2u_2(k-1)}{3}. \quad (23.28)$$

Навчання мережі здійснювалося за допомогою РМНК, а керування — на основі (23.27).

Як вхідні сигнали під час навчання мережі використовувалися некорельовані випадкові послідовності з рівномірним законом розподілу в інтервалі [–1, 1]. Навчання здійснювалося на основі подання мережі 5000 навчальних пар. На рис. 23.11, *а*, *б* наведено поверхні, описувані першим і другим рівняннями системи (23.28) відповідно й відновлені радіально-базисною мережею із кількістю нейронів, рівним 57. Аналогічний результат був отриманий для персептронної моделі. Результати, отримані за допомогою РБМ і багатошарового персептрона, виявилися практично ідентичними [146].



а



б

Рис. 23.11. Поверхні, описувані рівняннями (23.28)

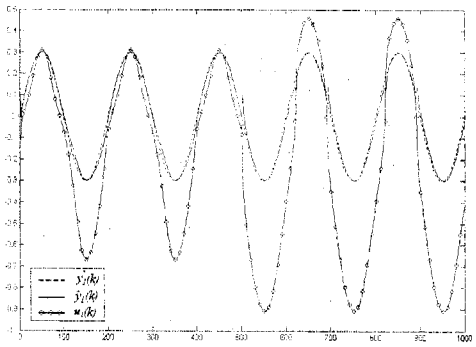
Результати роботи нейрорегулятора, що реалізує алгоритм (23.27), наведено на рис. 23.12–23.13. На всіх цих рисунках пунктирною лінією позначено необхідний вихідний сигнал $y_i^*(k)$, суцільною — реальний $\hat{y}_i(k)$, а лінією із кружками — відповідна

зміна керуючого сигналу $u_i(k)$ $(i = 1,2)$. Необхідні значення вихідних сигналів задавалися такі:

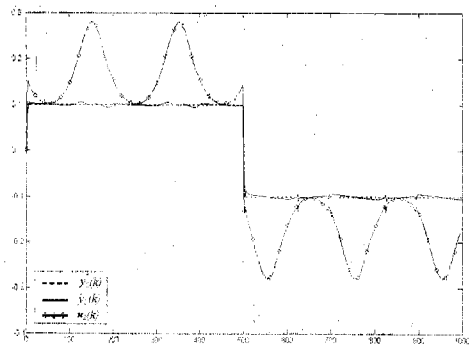
$$y_1^*(k) = 0{,}3\sin(\pi k/100);$$

$$y_2^*(k) = \begin{cases} 0{,}1 \text{ при } k = \overline{1,500}; \\ -0{,}1 \text{ при } k = \overline{501,1000}. \end{cases}$$

Рис. 23.12 відображає роботу регулятора за відсутності завад вимірювання ($\xi(k) = 0$), а рис. 23.15 — за наявності рівномірно розподіленої в інтервалі [–0,3; 0,3] випадкової завади $\xi(k)$. Поява даної завади призвела до незначного погіршення якості керування й збільшення помилки $\xi_2(k)$. Крім того, при побудові нейромережевої моделі збільшилася кількість нейронів і стала рівною 69.

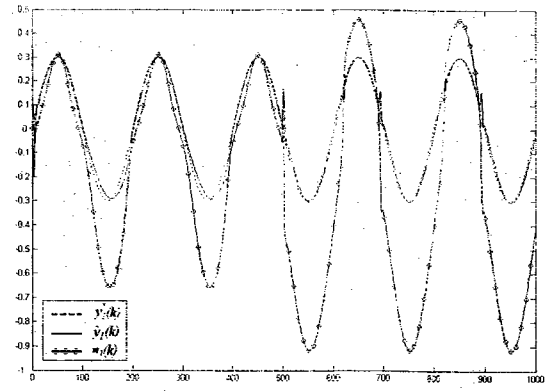


а

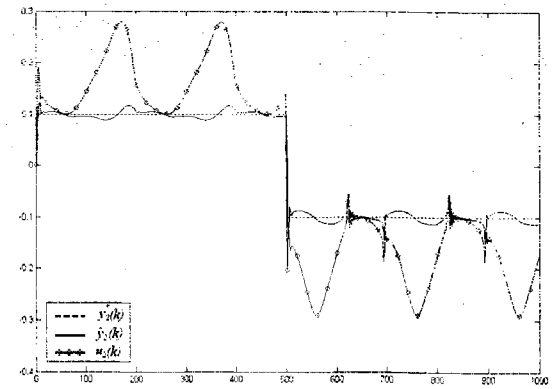


б

Рис. 23.12. Керування об'єктом, описуваним рівняннями (23.28)



а



б

Рис. 23.13. Керування об'єктом, описуваним рівняннями (23.30), за наявності завади $\xi(k)$

На рис. 23.14–23.15 наведено результати побудови нейромережевої моделі одновимірного нелінійного нестаціонарного об'єкта, описуваного рівняннями

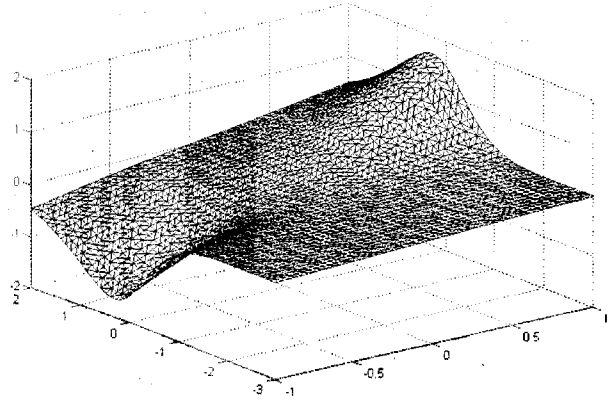
$$y(k+1) = \begin{cases} \dfrac{u(k)\left[\sin(y(k)) + 1{,}5\right]}{1 + y^2(k)}, & k = 1 \div 1250; \\ \dfrac{u(k)\left[\cos(y(k)) + 0{,}5\right]}{2 + y^2(k)}, & k = 1251 \div 2000, \end{cases} \tag{23.29}$$

і керування ним.

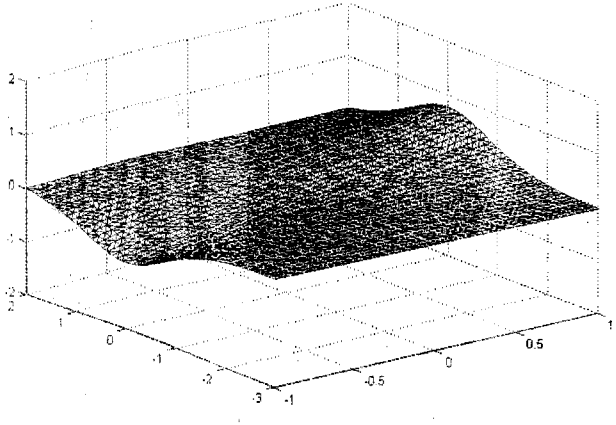
Перші 500 кроків використовуються для ідентифікації об'єкта. Після досягнення заданої точності керування здійснюється за алгоритмом (23.27).

При цьому задавалося $y_1^*(k) = \sin(\pi\, k/100)$.

На рис. 23.14 видно, що після зміни параметрів об'єкта на 1251 кроці, що призводить до появи великої помилки ідентифікації $e(k)$, відбувається корекція параметрів моделі протягом приблизно 50 кроків, а потім реалізація керування.



а



б

Рис. 23.14. Поверхні, описувані рівняннями (23.29)

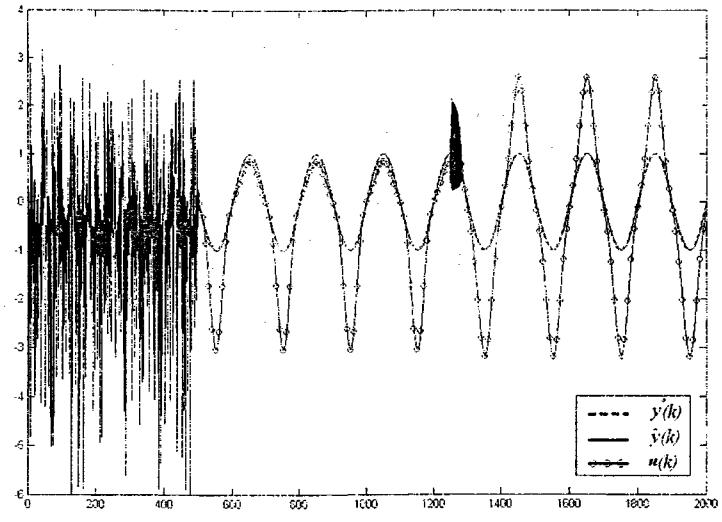


Рис. 23.15. Керування об'єктом, описуваним рівняннями (23.29)

23.6. Кластеризація даних

Розглянемо базу даних, створену *Р. Форсайтом*, що містить характеристики 101 тварини, розділені на сім класів. База даних складається з 17 полів, 15 із яких бінарні, одне містить цілі числа від нуля до восьми, і одне визначає клас тварини й у навчанні не бере участі.

Опис полів:

1. hair (наявність вовни, бінарне)
2. feathers (наявність пір'я, бінарне)
3. eggs (відкладає яйця, бінарне)
4. milk (годує молоком, бінарне)
5. airborne (здатна літати, бінарне)
6. aquatic (живе у воді, бінарне)
7. predator (хижак, бінарне)
8. toothed (наявність зубів, бінарне)
9. backbone (наявність хребта, бінарне)
10. breathes (бінарне)
11. venomous (отрутна, бінарне)
12. fins (наявність плавців, бінарне)
13. legs (кількість ніг, ціле, значення: {0,2,4,5,6,8})

14. tail (наявність хвоста, бінарне)
15. domestic (свійська, бінарне)
16. catsize (бінарне)
17. type (тип тварини, ціле, значення: [1,7])

Перелік усіх тварин, що є у базі даних

1 клас, ссавці (41 приклад): aardvark, antelope, bear, boar, buffalo, calf, cavy, cheetah, deer, dolphin, elephant, fruitbat, giraffe, girl, goat, gorilla, hamster, hare, leopard, lion, lynx, mink, mole, mongoose, opossum, oryx, platypus, polecat, pony, porpoise, puma, pussycat, raccoon, reindeer, seal, sealion, squirrel, vampire, vole, wallaby, wolf;

2 клас, птахи (20 прикладів): chicken, crow, dove, duck, flamingo, gull, hawk, kiwi, lark, ostrich, parakeet, penguin, pheasant, rhea, skimmer, skua, sparrow, swan, vulture, wren;

3 клас, плазуни (5 прикладів): pitviper, seasnake, slowworm, tortoise, tuatara;

4 клас, риби (13 прикладів): bass, carp, catfish, chub, dogfish, haddock, herring, pike, piranha, seahorse, sole, stingray, tuna;

5 клас, земноводні (4 приклади): frog, frog, newt, toad;

6 клас, комахи (8 прикладів): flea, gnat, honeybee, housefly, ladybird, moth, termite, wasp;

7 клас, ракоподібні (10 прикладів): clam, crab, crayfish, lobster, octopus, scorpion, seawasp, slug, starfish, worm.

Одним із найбільш ефективних підходів до вирішення задачі кластеризації тварин є використання мереж Когонена або самоорганізувальних мап (СОМ).

На рис. 23.16 зображено уніфіковану матрицю відстаней, отриману після навчання мережі з використанням цієї бази даних.

На мапі відображено дванадцять темних ділянок, розділених світлими межами, що відповідає дванадцятьом кластерам у вхідних даних. Три кластери, що відповідають класу ссавців, обумовлені більшою різноманітністю представників даного класу й можуть бути об'єднані в один макрокластер, оскільки перебувають компактно в правій верхній ділянці мапи. Три кластери на мапі відповідають також класу ракоподібних, причому один із кластерів розташований на мапі далеко від двох інших, що викликано помилками в базі даних, яка використовувалася для навчання: хробак і морська зірка не є ракоподібними. Якщо виключити ці

Рис. 23.16. Уніфікована матриця відстаней, отримана після навчання мережі з використанням бази даних тварин

приклади з вибірки, то зникне один кластер, а два інших можуть бути об'єднані, оскільки вони розташовані близько на мапі. Також два кластери на мапі відповідають класу плазунів, однак вони розташовані на мапі досить близько. Іншим класам відповідають на мапі окремі кластери. Чітко відділені класи птахів, риб, комах.

Додаткову інформацію про закономірності, присутні у даних, можна отримати, вивчаючи мапи входів. Так, наприклад, на мапі другого входу (рис. 23.17) бачимо, що світла ділянка, яка відповідає наявності пір'я в особин, збігається за формою із кластером птахів. Це свідчить про те, що цей параметр є найбільш значущим при класифікації тварин на птахів і не птахів. На мапі четвертого входу (рис. 23.18) світла ділянка, яка відповідає особинам, що годують молоком, збігається за формою із кластером ссавців. Отже, цей параметр найбільш суттєвий при класифікації тварин на ссавців й інших. Світла ділянка мапи дванадцятого входу (наявність плавців) відповідає за формою кластеру риб (рис. 23.19). Мапа п'ятого входу (рис. 23.20) свідчить про те, що здатність літати властива тільки птахам, комахам і ссавцям. Природно, ці висновки очевидні для людини, що пройшли курс біології. Однак СОМ не проходила цей курс, тому суттєвим є те, що вона виявилася здатною виділити ці закономірності.

Слід зазначити, що приклади, що відносяться до кожного класу, групуються компактно всередині одного із кластерів (після об'єднання відповідних кластерів у макрокластери), всередині кожного кластера перебувають дані, які відносяться тільки до одного класу. Це свідчить про те, що мережа на високому рівні впоралася із завданням кластеризації тварин.

У складніших випадках, наприклад під час розв'язання фінансових задач, закономірності, що є присутніми у даних, не настільки очевидні, і СОМ є незамінною для їхнього виявлення.

Рис. 23.17. Мапа другого входу навченої мережі (наявність пір'я). Біла ділянка збігається за формою із кластером птахів на рис. 23.16

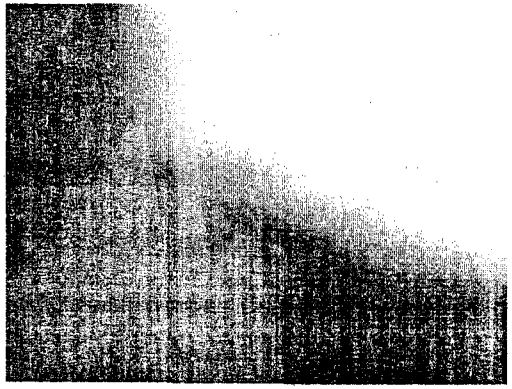


Рис. 23.18. Мапа четвертого входу навченої мережі (годування молоком)

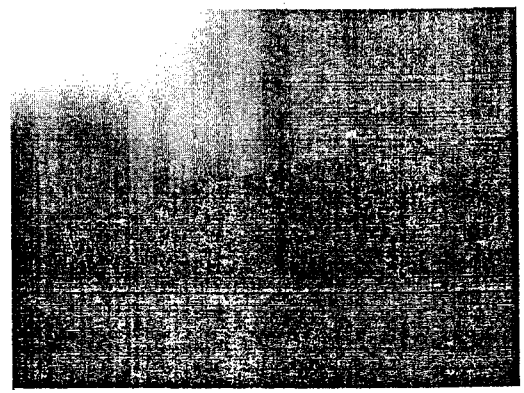


Рис. 23.19. Мапа дванадцятого входу (наявність плавців)

Рис. 23.20. Мапа п'ятого входу (здатність літати)

23.7. Стиснення даних

23.7.1. Стиснення даних за допомогою мережі Когонена

Самоорганізувальна мережа Когонена розбиває вхідний простір на кластери, всередині яких дані розрізняються незначно. Отже, якщо невеликі втрати інформації не істотні, то можна подати кожний із вхідних образів одного кластера еталонним способом, яким є вектор ваг нейрона-переможця, а потім номером еталонного образу (номером нейрона). Якщо кількість нейронів мережі значно

менша ніж кількість варіантів вхідних образів, то для кодування еталонів знадобиться значно менший обсяг інформації, ніж для кодування всіх вхідних образів. Для відновлення стисненої інформації необхідно зберігати таблицю ваг нейронів, а також номери нейронів-переможців, що відповідають вхідним образам.

Зі зменшенням кількості нейронів у мережі обсяг пам'яті, необхідний для зберігання таблиці ваг нейронів, а також для зберігання номера нейрона-переможця для кожного вхідного образу, зменшується, отже, коефіцієнт стиснення збільшується, однак помилка подання кожного вхідного вектора також збільшується, що призводить до зниження якості відновлення вхідних образів. Отже, змінюючи кількість нейронів у мережі, можна регулювати коефіцієнт стиснення і якість відновлення стисненої інформації.

Коефіцієнт стиснення інформації можна розрахувати за формулою

$$K = \frac{N_I \cdot I_I}{N_N \cdot I_I + N_I \cdot I_N},$$

де N_I — кількість вхідних образів; N_N — кількість нейронів у мережі; I_I — обсяг пам'яті, необхідний для зберігання одного образу або вектора ваг нейрона; I_N — обсяг пам'яті, необхідний для зберігання номера нейрона.

Під час стиснення інформації (зображення) за допомогою СОМ діють таким чином. Розбиваючи вхідне зображення на кадри розміром $n \times m$, записують кожен кадр у вигляді вхідного вектора, що містить $3nm$ компонент розміром один байт (по 3 байти для кожного піксела, що зображують червону, зелену й синю компоненти). Потім, використовуючи отриману вибірку, навчають мережу Когонена, що складається з N нейронів. Після заміни кожного з векторів вибірки на номер нейрона-переможця, що відповідає цьому вектору, ваги нейронів мережі, а також номери нейронів-переможців для кожного кадру зберігаються у файлі. Для відновлення зображення кожен номер нейрона-переможця замінюється на відповідний йому вектор ваг, забезпечуючи тим самим одержання інформації про кольори кожного піксела в кадрах, з деякими спотвореннями.

На рис. 23.21 наведено вихідне (*а*) і стиснене за допомогою СОМ (*б*) зображення.

а *б*

Рис. 23.21. Вихідне (*а*) і стиснене за допомогою СОМ (*б*) зображення

Обсяг пам'яті, необхідний для зберігання вихідного зображення, складає $380 \cdot 470 \cdot 3 = 535\,800$ байт. Під час стиснення зображення розбивалося на кадри розміром 2×2, мережа містила 128 нейронів. Отже, для зберігання стисненого зображення потрібно $((380 \cdot 470/(2 \cdot 2)) \cdot 7 + 128 \cdot 2 \cdot 2 \cdot 3 \cdot 8)/8 = 40\,605$ байт. Отриманий коефіцієнт стиснення $535\,800/40\,605 = 13{,}195$ зрівняємо з коефіцієнтом стиснення, одержуваним під час стиснення зображень методом JPG, якість зображення, однак, дещо гірша.

23.7.2. Стиснення зображень за допомогою ієрархічної ШНМ CMAC

З розвитком *Internet*-технологій значно зріс інтерес до методів ієрархічного кодування зображень, які дозволяють здійснювати швидку передачу зображень у реальному часі. Сьогодні для цієї мети широко застосовується метод адаптивного дискретного косинусного перетворення (АДКП), однак даний метод вимагає більших обчислювальних витрат і при кодуванні можливе виникнення так званого «блокового» ефекту. Альтернативним є метод, що використовує нейронну мережу CMAC, що дозволяє виділяти з вихідного сигналу різні частотні складові.

Для розв'язання цієї задачі використовується ієрархічна мережа CMAC, що складається з послідовно з'єднаних шарів, кожний з яких виділяє й апроксимує певну частотну складову вихідного зображення. Частота апроксимовного сигналу визначається кількістю степенів квантування ρ у мережі. Мережа першого шару, що має найбільше число степенів квантування й найменшу кількість настроюваних параметрів, навчається з максимальною швидкістю. Отримане в результаті грубе зображення, що дозволяє сформувати уявлення про вихідне зображення, за необхідності може бути уточнене в наступних шарах.

Схему кодування зображень за допомогою ієрархічної мережі CMAC, що складається з m шарів, наведено на рис. 23.22. Тут буквою Q позначені блоки, що здійснюють спеціальне кодування сигналів, які надходять на вхід мережі CMAC, «CMAC(ρ_i)» означає використання мережі з ρ_i степенями квантування. Вихідне зображення e_0 подається мережі першого шару як бажаний сигнал. Після навчання вихідний сигнал даної мережі g_1 порівнюється з e_0. Отримувана помилка e_1 передається у наступний шар. Під час декодування використовуються параметри мереж, визначені при їхньому навчанні. Результуюче зображення формується шляхом додавання вихідних сигналів мереж, що перебувають у прихованих шарах.

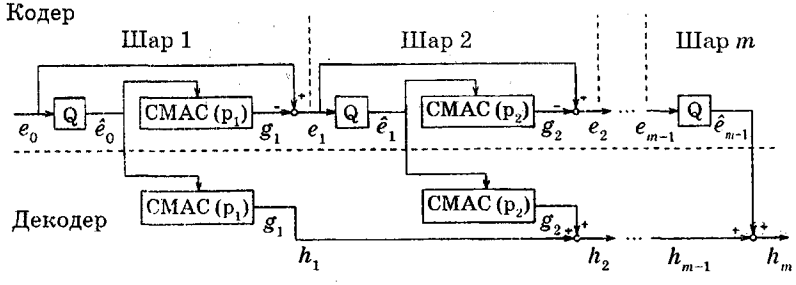
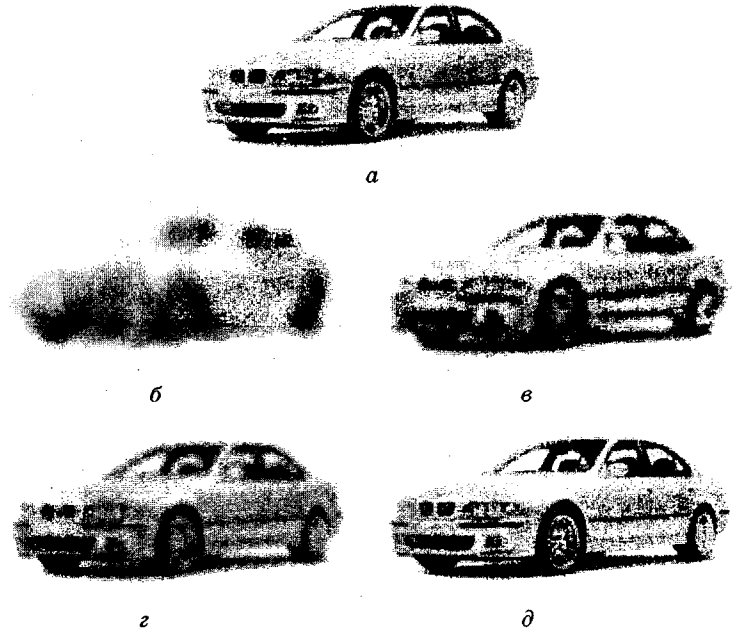


Рис. 23.22. Схема кодування зображень

Приклад кодування зображення наведено на рис. 23.23. На рис. 23.23, *а* наведено вихідне зображення розміром 100 × 200 пікселів. У мережі першого шару використовувалося $\rho_1 = 50$ степенів квантування, а кількість настроюваних параметрів було рівним 1484. Результат апроксимації вихідного зображення такою мережею зображено на рис. 23.23, *б*. У наступних підмережах кількість степенів квантування зменшувалося, а частота апроксимуючого сигналу зростала. У результаті отримано зображення, наведене на рис. 23.23, *д*. При цьому для зберігання всіх настроюваних параметрів мереж необхідно було 7250 комірок пам'яті замість необхідних для зберігання вихідного зображення 20 000 комірок, тобто обсяг необхідної для зберігання зображення пам'яті скоротився в 2,8 рази.



а *б* *в* *г* *д*

Рис. 23.23. Кодування зображення за допомогою мережі CMAC: *а* — вихідне зображення; *б* — шар 1, $\rho_1 = 50$; *в* — шар 2, $\rho_2 = 20$; *г* — шар 3, $\rho_3 = 10$; *д* — шар 4, $\rho_4 = 5$

23.8. Фінансове прогнозування на основі ШНМ

ШНМ досить широко використовуються у фінансовому прогнозуванні, зокрема для прогнозування ринку акцій і курсу обміну валют. Нижче ми коротко розглянемо дві системи прогнозування ринку акцій, що є вільною ринковою економічною системою,

динаміка якої залежить від закону попиту та пропозиції. На зростання і падіння цін акцій впливають різні економічні й психологічні фактори. Наявність стохастичного людського фактора, а також нелінійності, багатовимірності й найчастіше високого ступеня невизначеності, властивих природі операцій з акціями, роблять неефективним застосування традиційних методів прогнозування.

23.8.1. Прогнозування ринку акцій з використанням багатошарового персептрона

Прогнозуючу систему, зображену на рис. 23.24, описано в роботі [147]. Вона складається з декількох нейромережевих модулів, кожний з яких є тришаровим персептроном і використовується для навчання співвідношень між різними технічними й економічними факторами, а також для видачі рішення про покупку або продаж акцій. Кожен модуль має свою навчальну вибірку.

Входами системи є технічні й економічні показники (індекс Доу-Джонса, іноземні курси обміну, швидкість інтересу й ін.), а виходом — рішення про покупку або продаж акцій, отримуване усередненням всіх прогнозів. На рис. 23.24 TOPIX (*Tokyo Stock Exchange Price Indices*) розшифровуються як «Токійські індекси цін обміну акцій».

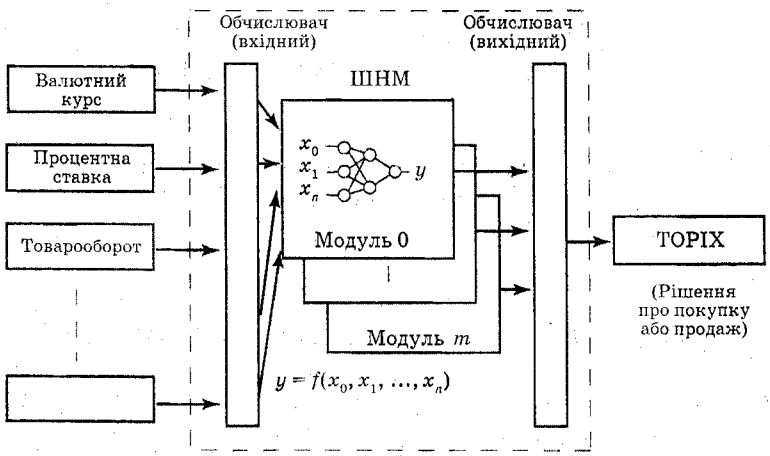


Рис. 23.24. Архітектура ШНМ для прогнозування ринку акцій

Перед початком навчання мережі визначалися довірчі інтервали (припустимі помилки) для вихідних сигналів, а для навчання мережі використовувався алгоритм зворотного поширення. У процесі навчання ваги настроювалися тільки тоді, коли вихідна помилка перевищувала припустиму. Ті ж дані, для яких вихідні помилки не перевищували припустимих, з навчальної множини виключалися, прискорюючи тим самим (зменшенням кількості обчислень) процес навчання.

Вхідні дані, що являють собою ковзні середні щотижневих зазначених вище факторів, попередньо оброблялися шляхом логарифмування або іншим способом, а потім нормалізувалися, тобто подавалися величинами з інтервалу [0, 1]. Навчальні дані для рішення купити/продати — це зважена сума тижневих повернень. Значення виходу мережі, що дорівнюють або більше 0,5, означало купити акції, а менше 0,5 — продати.

На рис. 23.25 наведено результати роботи прогнозуючої системи за більш ніж 33 місяці, починаючи із січня 1987 р., коли початковий індекс прибутку був прийнятий за 1,0, по вересень 1989 р. Як випливає з рис. 23.25, ефективність роботи системи виявилася вищою, ніж при звичайному прогнозуванні.

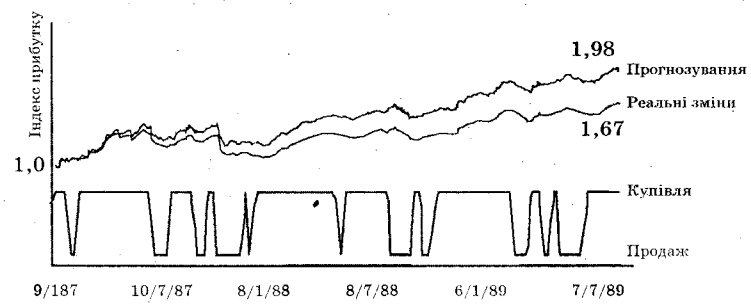


Рис. 23.25. Результати прогнозування ринку акцій

23.8.2. Прогнозування ринку акцій на основі мереж ART

Відносно молодою і непередбачуваною є тайванська біржа TSE — *Taiwan Stock Exchange*, початковий індекс якої дорівнював 100, за 30 років перевищив оцінку 9000, а протягом декількох місяців змінювався в ту або іншу сторону більш ніж на 50 %. Все це робить непридатним класичні методи, які не встигають стежити за такою швидкою зміною ситуації. Тому тут використано ШНМ, які швидко навчаються, а саме мережі ART-1.

Для розв'язання задачі прогнозування індексу TSE обрано 5 факторів (обмінний грошовий курс, обсяг імпорту, обсяг експорту, ліквідність і ціновий індекс споживання), для яких методом регресійного аналізу встановлено кореляційні залежності. Зв'язки проаналізовано за допомогою мережі ART-1, що мала для кожного фактора 3 нейрони вхідного шару й різну кількість нейронів вихідного шару. Входом був нормалізований з точністю до трьох знаків, як й обрані п'ять факторів, індекс TSE. Проаналізовано результати короткострокового прогнозування для різних значень параметра подібності $p \in (0, 1]$, що дозволило встановити ідентичність результатів, отримуваних методом регресійного аналізу й за допомогою мережі ART-1.

Для прогнозування тренду індексу TSE використано багатошарову ШНМ прямого поширення, на вхід якої подавалися сигнали з виходу мережі ART-1. Така комбінація мереж дозволила, як стверджують автори роботи [148], істотно підвищити точність прогнозу.

23.9. Нейромережева інформаційно-довідкова система

Багато фірм, які займаються розробкою складної продукції, що має величезну кількість деталей і вузлів, використовують для зберігання всіх відомостей про ці деталі й вузли інформаційно-довідкові системи. Це дозволяє усунути дублювання в роботі різних груп розроблювачів, а також якщо знадобиться використати вже наявні розробки у знову створюваних виробах. Такі системи тим більше необхідні, якщо кількість найменувань вимірюється десятками й сотнями тисяч.

Зазвичай інформаційно-довідкові системи використовують спеціальну індексацію або кодування найменувань виробів. Такий підхід найчастіше неефективний, оскільки час пошуку потрібної інформації може бути значним. Це змусило фахівців авіакомпанії «Boeing» шукати інші підходи до розв'язання цієї задачі. У результаті ними була створена нейромережева інформаційно-довідкова система, що використовує геометричні дані про деталі, прийняті в CAD, і що класифікувала й зберігала більше 20 тисяч різних найменувань авіаційних деталей. Ця система мала так звану NIRS (*Neural Information Retrieval System*) архітектуру, що складалася із трьох рівнів, кожний з яких відповідав певним

властивостям груп виробів. У свою чергу, кожен рівень складався з модулів, що являють собою мережі ART-1 [149]. На нижньому рівні зберігалися зразки, об'єднані в групи на основі подібності їхніх обрисів. На наступному рівні враховувалися вигини, а на верхньому — отвори.

Отже, дана мережа, використовуючи як критерій, що є її вхідним сигналом, різні комбінації обрисів, вигинів й отворів, здійснювала пошук деталей. Пошук починався з нижнього рівня, і якщо знаходився модуль, до якого відносилася пропонована деталь, здійснювався перехід до другого рівня тощо.

Навчання мережі здійснювалося шляхом подання їй образів, які згодом вона й повинна була зберігати. Нижній рівень зберігав двовимірні CAD-зображення (контури) деталей у вигляді картинок з 400 × 400 пікселів, поданих вектором у двійковому коді. Двійковий вектор відповідної розмірності був вхідним сигналом, і здійснювалося розбивання образів на групи, збережені в окремих модулях. Аналогічний підхід застосовувався для опису вигинів й отворів.

Навчання тривало протягом 12 годин, причому більша частина часу була витрачена на створення CAD-описів. Час же пошуку інформації й видачі результату складав від 30 до 45 секунд.

Контрольні запитання

1. Що являє собою мережа *NETtalk*?
2. Яким чином можуть використовуватися ШНМ для керування автомобілем?
3. Як за допомогою ШНМ може бути вирішена задача ідентифікації реальних об'єктів? Яку структуру при цьому має модель?
4. Наведіть приклади архітектур ШНМ, використовуваних при фінансовому прогнозуванні.
5. Яким способом може бути реалізована на базі ШНМ інформаційно-довідкова система?

ЛІТЕРАТУРА

1. *Mc Culloch W. S., Pitts W.* A Logical Calculus of the Ideas Immanent in Nervous Activity // Bulletin of Mathematical Biophysics. — 1943. — № 5. — P. 115–133.
2. *Hebb D.* The Organization of Behavior. — New York: Wiley Publications, 1949.
3. *Rochester N. J., Holland J. H., Haibt L. H., Duda W. L.* Test on a Cell Assembly Theory on the Action of the Brain, Using a Large Digital Computer // IRE Trans. on Inf. Theory. — 1956. — IT-2. — P. 80–93.
4. *Rosenblatt F.* The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain // Psychological Review. — 1958. — № 65. — P. 386–408.
5. *Розенблатт Ф.* Принципы нейродинамики. Персептрон и теория механизмов мозга: Пер. с англ. — М.: Мир, 1965.
6. *Widrow B., Hoff M. E.* Adaptive Switching Circuits / IRE WESCON Convention Record — New York, IRE, 1960. — P. 96–104.
7. *Widrow B., Winter R.* Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition // IEEE Computer. — 1988. — № 21. — № 3. — P. 25–39.
8. *Минский М. Л., Пейперт С.* Персептроны: Пер. с англ. — М.: Мир, 1971.
9. *Kohonen T.* Correlation Matrix Memories // IEEE Trans. on Computers. — 1972. — 21. — P. 253–259.
10. *Kohonen T.* Associative Memory: A System Theoretic Approach. — Berlin: Springer, 1977.
11. *Kohonen T.* Self-Organized Formation of Topologically Correct Feature Maps // Biological Cybernetics. — 1982. — 43. — P. 59–69.
12. *Kohonen T.* The Self-Organizing Map // Proc. of IEEE. — 1990. — 78 — № 9. — P. 1464–1480.
13. *Kohonen T.* The «Neural» Phonetic Typewriter // IEEE Computer. — 1988. — 21. — № 3. — P. 11–22.
14. *Kohonen T.* Self-Organization and Associative Memory. 3rd ed. — New York: Springer Verlag, 1989.
15. *Anderson J. A.* A Memory Model Using Spatial Correlation Functions // Kybernetik. — 1968. — 5. — P. 113–119.
16. *Anderson J. A.* Two Models for Memory Organization // Mathematical Biosciences. — 1970. — 8. — P. 137–160.
17. *Werbos P. J.* Bejond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Ph.D. thesis. — Cambridge, MA, Harvard University, 1974.
18. *Parker D.* Learning Logic // Technical Report TR-87, Center for Computational Research in Economics and Management Sciences — Cambridge, MA: MIT Press, 1985.
19. *Rumelhart D. E., Hinton G. E., Williams R. J.* Learning Internal Representations by Error Propagation / In Parallel Distributed Processing: Explorations in the Microstructure of Cognition. D.E. Rumelhart, J.L. McClelland (eds). — Cambridge: MIT Press, 1986. — Vol. 1. — Chapt. 8. — P. 318–364.
20. *Grossberg S.* Embedding Fields: A Theory of Learning with Physiological Implications // J. of Math. Psychology. — 1969. — 6. — P. 209–239.
21. *Grossberg S.* Competitive Learning: From Interactive Activation do Adaptive Resonance // Cognitive Science. — 1987. — 11. — P. 23–63.
22. *Cohen M. A., Grossberg S.* Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks // IEEE Tr. on Systems, Man, and Cybernetics. — 1983. — 13. — P. 815–826.
23. *Hopfield J. J.* Neural Networks and Physical Systems with Emergent Collective Computational Abilities // Proc. of the National Academy of Science. — 1982. — 79. — P. 2554–2558.
24. *Hopfield J. J.* Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons // Proc. of the National Academy of Science. — 1982. — 81. — P. 3088–3092.
25. *Hopfield J. J., Tank D. W.* Neural Computation of Decisions in Optimization Problems // Biol. Cybernetics. — 1985. — 52. — P. 141–152.
26. *Sejnowski T. J., Rosenberg Ch.* NETtalk: A Parallel Network that Learns to Read Aloud. John Hopkins University Electrical Engineering and Computer Science Technical Report JHU / EECS — 86/01. — 1986. — 32 p.
27. *Hecht-Nielsen R.* Kolmogorov's Mapping Neural Network Existence Theorem / Proc. of IEEE First Ann. Int. Conf. on Neural Networks. — San Diego, 1987. — 3. — P. 11–13.
28. *Hecht-Nielsen R.* Theory of the Backpropagation Neural Network / Proc. of Int. Joint. Conf. on Neural Networks. — Washington, D.C., 1989. — 1. — P. 593–606.
29. *Колмогоров А. Н.* О представлении непрерывных функций нескольких переменных суперпозициями функций меньшего числа переменных // Докл. АН СССР. — 1956. — 108. — № 2. — С. 179–182.
30. *Колмогоров А. Н.* О представлении непрерывных функций нескольких переменных в виде суперпозиции непрерывных функций одного переменного и сложения // Докл. АН СССР, 1957. — 114. — С. 953–956.
31. *Cybenko G.* Approximation by Superposition of a Sigmoidal Function // Math. Contr. Sign. Syst. — 1989. — 2. — P. 303–314.
32. *Hornik K., Stinchcombe M., White H.* Multilayer Feedforward Networks are Universal Approximators // Neural Networks. — 1989. — 2. — P. 359–366.
33. *Hornik K.* Approximation Capabilities of Multilayer Feedforward Networks // Neural Networks. — 1991. — 4. — P. 251–257.
34. *Patterson D.* Artifical Neural Networks, Theory and Application. — Singapore: Prentice Hall Inc., 1996.
35. *Bishop C. M.* Neural Networks for Pattern Recognition — Oxford: Clarendon Press, 1995.
36. Image Processing and Pattern Recognition / T. Leondes (ed) — London: Academic Press, 1998.
37. *Pham D. T., Liu X.* Neural Networks for Identification, Prediction and Control. — London: Springer Verlag, 1997.
38. Control and Dynamic Systems / T. Leondes (ed) — London: Academic Press, 1998.

39. Industrial and Manufacturing Systems / T. Leondes (ed) — London: Academic Press, 1998.

40. *Омату С., Халид М., Юсоф Р.* Нейроуправление и его приложения: Пер. с англ. — М.: ИПРЖР, 2000.

41. Neural Networks for Control / Miller W.T., Satton R.S., Werbos P.J. (eds) — Cambridge: MIT Press, 1991.

42. *Zbikowski R., Hunt K. J.* Neural Adaptive Control. — Berlin: Springer — Verlag, 1996.

43. *Masters T.* Neural, Novel and Hybrid Algorithms for Time Series Prediction. — N.Y.: John Willey and Sons, Inc., 1995.

44. *Beltratti A., Margarita S., Terna P.* Neural Networks for Economic and Financial Modeling. — London: Int. Thomson Computer Press, 1996.

45. *Zirilli J.* Financial Prediction using Neural Network — London: Int. Thomson Computer Press, 1997.

46. *Pham D. T., Liu X.* Modeling and Prediction using GMDH Networks of Adalines with Nonlinear Preprocessors // Int. J. System Science. — 1994. — 25. — № 11. — P. 1743–1759.

47. *Галушкин А. И.* Теория нейронных сетей. — М.: ИПРЖР, 2000. — Кн.1: Учебное пособие для вузов.

48. *Hirai Y.* VLSI Neural Network System. — Phan, 1995.

49. *Шеперд Г.* Нейробиология: В 2 т.: Пер. с англ. — М.: Мир, 1987. — Т.1.

50. *Хьюбел Д., Стивенс Ч., Кэндел Э. и др.* Мозг: Пер. с англ. — М.: Мир, 1982.

51. *Хьюбел Д.* Глаз, мозг, зрение: Пер. с англ. — М.: Мир, 1990.

52. *Стоун І.* Бунт розуму: Пер. з англ. // Всесвіт, 1991. — № 4. — С. 3–52, № 5. — С. 121–163.

53. *Anderson J. A., Rosenfeld E.* Neurocomputing: Foundations of Research. — Cambridge, MA: MIT Press, 1988.

54. *Ham F. M., Kostanic I.* Principles of Neurocomputing for Science and Engineering. — N.Y.: Mc Graw-Hill Inc, 2001.

55. *Zell A.* Simulation neuronaler Netze. — München: R. Oldenburg Verlag, 2000.

56. *Nauck D. B., Klawonn F., Kruse R.* Neuronale Netze und Fuzzy-Systeme. — Braunschweig / Wiesbaden: Vieweg, 1996.

57. *Bothe H. H.* Neuro-Fuzzy Methoden: Einführung in Theorie und Anwendungen — Berlin: Springer Verlag, 1998.

58. *Rojas R.* Theorie der neuronalen Netze: eine systematische Einführung. — Berlin, Springer, 1996.

59. *Медведев В. С., Потемкин В. Г.* Нейронные сети. MATLAB 6 / Под общ. ред. В. Г. Потемкина. — М.: Диалог — МИФИ, 2002.

60. *Осовский С.* Нейронные сети для обработки информации. — М.: Финансы и статистика, 2002.

61. *Fukushima K.* Cognitron: A Self-organizing Multiplayered Neural Network //Biol. Cybern. — 1975. — 20. — P. 121–136.

62. *Fukushima K.* Neocognitron: A Self-organizing Neural Network for a Mechanism of Pattern Recognition Unaffected by Shift in Position // Biol. Cybern. — 1980. — 36. — P. 193–202.

63. *Fukushima K., Miyake S., Ito T.* Neocognitron: A Neural Network Model for a Mechanism of Visual Pattern Recognition // IEEE Trans. on Syst., Man and Cybern. — 1983. — 13. — P. 826–834.

64. *Fukushima K.* A Neural Network for Visual Pattern Recognition // IEEE Computer. — 1988. — 21. — № 3. — P. 65–75.

65. *Haykin S.* Neural Networks. A Comprehensive Foundation. — Upper Saddle River, N. Y.: Prentice Hall, Inc., 1999.

66. *Oja E.* Simplified Neuron Model as a Principal Component Analyzer // J. Math. Biology. — 1982. — 15. — P. 267–273.

67. *Oja E.* Principal Components, Minor Components and Linear Neural Networks // Neural Networks. — 1992. — 5. — P. 927–935.

68. *Поляк Б. Т.* Введение в оптимизацию. — М.: Мир, 1984.

69. *Вазан М.* Стохастическая аппроксимация. — М.: Мир, 1972.

70. *Алберт А.* Регрессия, псевдоинверсия и рекуррентное оценивание: Пер. с англ. — М.: Наука, 1977.

71. *Руденко О. Г.* Проекционные алгоритмы обучения искусственных нейронных сетей // Доп. НАН України. — 2003. — № 7. — С. 69–73.

72. *Браммер К., Зиффлинг Г.* Фильтр Калмана — Бьюси. — М.: Наука, 1982.

73. *Suykens J. A. K., Vandeville J. P. L., De Moor B. L. R.* Artifical Neural Networks for Modeling and Control of Non-Linear Systems. — Boston: Kluwer Academic Publishers, 1996.

74. *Shepherd A. J.* Second-order Methods for Neural Networks. — London: Springer-Verlag, 1997.

75. *Barto A. G., Anandan P.* Pattern Recognition Stochastic Learning Automata // IEEE Trans. Systems, Man, and Cybernetics. — 1985. — SMC — 15. — P. 360–375.

76. *Barto A. G., Jordan M. I.* Gradient Following Without Backpropagation in Layered Networks // IEEE 1 st. Int. Conf. Neural Networks, San Diego. — 1987. — 2. — P. 629–636.

77. *Бодянский Е. В., Руденко О. Г.* Искусственные нейронные сети: архитектуры, обучение, применения. — Харьков: Телетех, 2004.

78. *Zakharian S., Ladevig-Riebler P., Tores S.* Neuronale Netze für Ingenieure: Arbeits- und Übungsbuch für regelungstechnische Anwendungen. — Braunschweig: Vieweg, 1998.

79. *Widrow B., Lehr M.* 30 Years of Adaptive Neural Networks: Perceptron, Madaline and Backpropagation // Proc. IEEE. — 1990. — 78. — № 9. — P. 1415–1442.

80. *Ивахненко А. Г.* Самоорганизующиеся системы распознавания и автоматического управления. — К.: Техніка, 1969.

81. *Ивахненко А. Г.* Системы эвристической самоорганизации в технической кибернетике. — К.: Техніка, 1971.

82. *Lippman R. P.* An Introduction to Computing with Neural Nets // IEEE ASSP Magazine. — 1987. — № 4. — P. 4–22.

83. *Себер Дж.* Линейный регрессионный анализ: Пер. с англ. — М.: Мир, 1980.

84. *Kosko B.* Adaptive Biderectional Associative Memories // Appl. Optics. — 1987. — 26. — № 33. — P. 4947–4960.

85. *Kosko B.* Biderectional Associative Memories // IEEE Trans. on Syst., Man, and Cybern. — 1988. — 18. — P. 49–60.

86. *Anderson J. A.* Cognitive and Psychological Computation with Neural Models // IEEE Trans. on Syst., Man, and Cybern. — 1983. — 13. — P. 799–815.

87. *Banzhatt W., Haken H.* A Network for Recognition and Classification of Continues Pattern // Neural Networks. — 1988. — № 1. — P. 6.

88. *Haken H., Fuchs A., Banzhatt W.* Mustererkennung durch synergetische Computer. Teil 1 und 2 // Design and Elektronik, 1989.

89. *Pineda F. J.* Dynamic and Architectures for Neural Computation // J. of Complexity. — 1988. — 4. — P. 216–245.

90. *Pineda F. J.* Recurrent Backpropagation and the Dynamic Approach to Adaptive Neural Computation // Neural Computation. — 1989. — 1. — P. 161–172.

91. *Almeida L. B.* A Learning Rule for Asynchronous Perceptron with Feedback in a Combinatorial Environment // Proc. of the First IEEE Int. Conf. on Neural Networks, USA, San Diego, 1987. — Vol. 2. — P. 609–618.

92. *Jordan M. I.* Attractor Dynamics and Parallelism in a Connectionist Sequential Machine // Proc. of the Eight Annual Conf. of the Cognitive Science Society, Erlbaum, Hillsdale NJ, 1986. — P. 531–546.

93. *Elman J. L.* Finding Structure in Time // Cognitive Science. — 1990. — 14. — P. 179–211.

94. *Nelles O., Ernst S., Isermann R.* Neuronale Netze zur Identifikation nichtlinearer, dynamischer Systeme: Ein Überblick // Automatisierungstechnik. — 1997. — 45. — № 6. — S. 251–262.

95. *Williams R. S., Zipser D.* A Learning Algorithm for Continually Running Fully Recurrent Neural Network // Neural Computations. — 1989. — 1. — P. 270–280.

96. *Hinton G. E., Sejnowski T. J.* Analyzing Cooperative Computation / Proc. of the Firth Annual Conf. on the Cognitive Sci. Cociety — Rochester, NY, 1983. — P. 488–553.

97. *Ackley D. H., Hinton G. E., Sejnowski T. J.* A Learning Algorithm for Boltzmann Machines // Cognitive Sci. — 1985. — 9. — P. 147–169.

98. *Hinton G. E.* Connectionist Lerning Procedures // Art. Intelligence. — 1989. — 40. — P. 185–234.

99. *Kaczmarz S.* Angenäherte Auslösung von System linearer Gleichungen // Bull. Int. Acad. Polon. Sci. Lett., CI, Sci. Math. Nat., Ser. A, 1937. — S. 355–357.

100. *Geman S., Gemen D.* Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images // IEEE Trans. on Pattern Analysis and Machine Intelligence. — 1984. — G. — P. 721–741.

101. *Specht D. F.* Probabilistic Neural Network for Classification, Mapping or Associative Memory / Proc. of the IEEE Int. Conf. on Neural Networks, SanDiego, 1988. — Vol. 1. — P. 525–532.

102. *Specht D. F.* A General Regression Neural Network // IEEE Trans. on Neural Networks. — 1991. — 2. — № 6. — P. 568–576.

103. *Фукунага К.* Введение в статистическую теорию распознавания образов: Пер. с англ.– М.: Наука, 1979.

104. *Parzen E.* An Estimation of a Probability Density Function and Mode // Ann. Math. Stat. — 1962. — 33. — P. 1065–1076.

105. *Cacoullos T.* Estimation of Multivariable Density // Ann. Inst. Stat. Math. — 1966. — № .18. — P. 179–189.

106. *Powell M. J. D.* Radial Basis Functions for Multivariable Interpolation: A review / Proc. of IMA Conf. on Algorithms for the Approximation of Functions and Data, Shrivenham, UK. — 1985. — P. 143–167.

107. *Broomhead D. S., Lowe D.* Multivariable Function Interpolation and Adaptive Networks // Complex Systems. — 1988. — 2. — P. 321–355.

108. *Lowe D.* Adaptive Radial Basis Function Nonlinearities and the Problem of Generalization / Proc. of IEE Int. Conf. of Artifical Neural Networks, London, UK. — 1989. — P. 171–175.

109. *Poggio T., Girosi F.* Networks for Approximation and Learning // Proc. of IEEE. — 1990. — 78(9). — P. 1481–1497.

110. *Айзерман М. А., Браверман Э. М., Розоноэр Л. И.* Метод потенциальных функций в теории обучения машин. — М.: Наука, 1970.

111. *Albus J. S.* A New Approach to Manipulator Control: the Cerebellar Model Articulation Controller (CMAC) // ASME Trans., J. Dynamic Systems, Measurement and Control. — 1975. — 97. — № 3. — P. 220–227.

112. *Albus J. S.* Data Storage in Cerebellar Model Articulation Controller (CMAC) // ASME Trans. J. Dynamic Systems, Measurement and Control. — 1975. — 97. — № 3. — P. 228–233.

113. *Кнут Д. Э.* Искусство программирования. — 2-е изд. — М.: Вильямс, 2000. — Т. 3. Сортировка и поиск. — С. 549–597.

114. *Zi-Qin Wang, Schiano L. J., Ginsberg M.* Hash-Coding in CMAC Neural Networks // IEEE Int. Conf. on Neural Networks, 1996. — 3. — P. 1698–1703.

115. *Hecht-Nilsen R.* Neurocomputing. — Addison-Wesley, 1990.

116. *Fahlman S. E., Lebiere C.* The Cascade-Correlation Learning Architecture / Carnegi Mellon Report. Nr. CMU-CS-88-162, 1990.

117. *Waibel A., Hanazawa T., Hinton G., Shikano K., Lang K. J.* Phoneme Recognition using Time — Delay Neural Network // IEEE Trans. on Acoustics, Speech and Signal Processing, 1989. — 37. — № 3. — P. 328–339.

118. *Waibel A.* Consonant Recognition by Modular Construction of Large Phonetic Time — Delay Neural Network. / In Advances in Neural Information Processing Systems 1. D.S. Touretzky (ed.), 1989. — P. 215–223.

119. *Waibel A., Hampshire J.* Building Blocks for Speech // BYTE. — 1989. — Aug. — P. 235–242.

120. *Haffner P., Waibel A., Sawai H., Shikano K.* Fast Back — Propagation Learning Methods for Large Phonemic Neural Network / Proc. of Euro Speech, 1989. — P. 553–556.

121. *Haffner P., Waibel A.* Time — Delay Neural Network Embedding Time Alignment: a Performance Analysis / Proc. of Euro Speech, 1991.

122. *Lang K. J., Waibel A. H., Hinton G. E.* A Time — Delay Neural Network Architecture for Isolated Word Recognition // Neural Network. — 1990. — 3. — № 1. — P. 23–43.

123. *Carpenter G. A., Grossberg S.* The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network // Computer. — 1988. — March. — P. 77–88.

124. *Grossberg S.* Nonlinear Neural Networks: Principles, Mechanisms and Architectures // Neural Networks, 1988. — 1. — № 1. — P. 17–61.

125. *Carpenter G., Grossberg S.* ART2: Self-organization of Stable Category Recognition Codes for Analog Input Patterns // Applied Optics. — 1987. — 26. — P. 4919–4930.

126. *Carpenter G., Grossberg S.* ART3: Hierarchical Search Using Chemical Transmitters in Self-Organizing Pattern Recognition Architecture // Neural Networks. — 1990. — 3. — №1. — P. 129–152.

127. *Carpenter G., Grossberg S., Rosen D. B.* Fuzzy ART: Fast Stable Learning and Categorization of Analog Input Patterns by an Adaptive Resonance System // Neural Networks. — 1991. — 4. — P. 759–771.

128. *LeCun Y., Denker J. S., Solla S. A.* Optimal Brain Damage / In Advanced in Neural Processing Systems 2. Touretzky D. S. (ed). — Morgan-Kaufmann, 1990. — P. 598–605.

129. *Hassibi B., Stork D. G.* Second Order Derivatives for Network Pruning: Optimal Brain Surgeon / Hansen S. J., Cowan J. D. In Advanced in Neural Processing Systems 5. Giles C. L. (eds). — Morgan-Kaufmann, 1993.

130. *Hassibi B., Stork D. G., Woltt G.* Optimal Brain Surgeon and General Network Pruning / IEEE Int. Conf. on Neural Network, San Francisco, CA., 1993. — Vol. 1. — P. 293–299.

131. *Mozer M. C., Smolensky P.* Skeletonization: a Technique for Trimming the Fat from a Network Via Relevance Assesment // In Advanced in Neural Information Processing Systems 1. Tourezky D. S.(ed), 1989. — P. 107–105.

132. *Тихонов А. Н., Арсенин В. Я.* Методы решения некорректных задач. — М.: Наука, 1980.

133. *Hanson S. J., Pratt L. Y.* Comparing Biases for Minimal Network Construction with Back-Propagation // In Advanced in Neural Information Processing Systems 1. Tourezky D.S.(ed), 1989. — P. 177–185.

134. *Chauvin Y.* A Back-Propagation Algorithm with Optimal Use of Hidden Units // Advanced in Neural Information Processing Systems 1. Tourezky D.S.(ed), 1989. — P. 519–526.

135. *Holland J. L.* Adaptation in Natural and Artificial Systems: An Introductory Analysis with Application to Biology, Control and Artificial Systems. — The University of Michigan Press, Ann Arbor. — 1975.

136. *Дубинин Н. П.* Общая генетика. — М.: Наука, 1986.

137. *Whitley D., Dominic S., Das R.* Genetic Reinforcement Learning with Multilayer Neural Networks / In Belew and Booker. — 1991. — P. 562–570.

138. *Tsoukalas L. H., Uhrig R. E.* Fuzzy and Neural Approaches in Engineering. — New York: John Wiley Sons, Inc. — 1997.

139. *Harp S., Samad T.* Genetic Optimization of Neural Networks Architectures for Electric Utility Applications. / Final Report. Electric Power Research Institute, Research Project №8016-04, Palo Alto, CA. March. — 1994.

140. *Koza J. P.* Genetic Programming: On the Programming of Computers by Means of Natural Selection. — Cambridge, MA, MIT Press, 1992.

141. *Koza J. P.* Genetic Programming II: Automatic Discovery of Rensable Programs. — Cambridge, MA, MIT Press, 1994.

142. *Koza J. P.* Genetic Programming III: Darwinian Invention and Problem Solving. — San Francisco, CA, Morgan Kaufman, 1999.

143. *Bothe H. H.* Artitical Visual Speech, Generated by Fuzzy Interence Methods // Proc. 2nd TIDE Congres. Paris. — Amsterdam Press, 1995. — P. 302–305.

144. *Thorpe C. Hebert M., Kanade T., Shafer S.* Toward Autonomous Driving: The CMU Navlab // IEEE Expert. — 1991. — 8. — P. 31–42.

145. *Kande T., Reed M. L., Weiss L. E.* New Technologies and Aplications in Robotics // Communications of the ACM. — 1994. — 37. — № 3. — P. 58–67.

146. *Руденко О. Г., Бессонов А. А.* Адаптивное управление многомерными нелинейными объектами на основе радиально-базисных сетей // Кибернетика и системный анализ. — 2005. — № 2. — С. 168–175.

147. *Kimoto T., Asakawa K.* Stock Market Prediction System with Modular Neural Networks / Proc. of Int. Conf. on Neural Networks. SanDiego, 1990. — 1. — P. 1–6.

148. *Szu H., Liu K. W., Chao C. C., Liu K. F., Hsu K. T., Medsker L.* Proc. of the IJCNN-92, Beijing, 1992. — P. I-333, I-339.

149. *Smith S. D., Escobedo R., Candell T. P.* An Industrial Strength Neural Network Applicathion / Proc. of INNS Meeting, World Congress on Neural Networks, Portland, OR., 1993. — Vol. 1. — P. 490–494.

ЗМІСТ

Навчальне видання

РУДЕНКО Олег Григорійович
[illegible]ЧНСЬКИЙ Євгеній Володимирович

ШТУЧНІ НЕЙРОННІ МЕРЕЖІ

Навчальний посібник

Редактор *Ю. В. Статкевич*
Комп'ютерна верстка *О. А. Федосєєвої*
Дизайн обкладинки *А. В. Пивоварова*

Видано за рахунок державних коштів. Продаж заборонено

Підписано до друку 7.08.2006. Формат 60×90/16. Папір офсетний.
Гарнітура SchoolBookC. Друк офсетний. Умов. друк. 25,25 арк.
Обл.-вид. арк. 26,5. Тираж 2950 прим. Зам. № __

Видавництво «Компанія СМІТ»
61166, м. Харків, просп. Леніна,14
Тел. 8(057) 717-54-94,702-08-16
Факс: 8(057) 702-13-07
E-mail: book@smit.kharkov.ua
http://www.smit-book.com

Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру видавців, виготівників і розпровсюджувачів
видавничої продукції ДК № 435 від 26.04.2001

Друк — ТОВ «СІМ»
61012, м. Харків, пров. Лопанський, 3-а
Тел.: 8(057)719-19-30
E-mail: tipa_graf@pisem.net