А.І. Безверхий

Когнітивні науки в інженерії програмного забезпечення

ЗМІСТ

ВСТУП

Когнітивістика, або інакше Cognitive Science - цей науковий напрям, який займається вивченням пізнавальних і розумових процесів і моделюванням принципів, по яких працюють природні і штучні системи.

Когнітивістика об'єднує багато напрямів і теорії : когнітивну психологію, нейрофізіологію, теорію штучного інтелекту і пізнання, а також лінгвістику.

Інтелектуальні системи на основі машинного навчання дозволяють з успіхом вирішувати проблеми розпізнавання образів, виконання прогнозів, оптимізації, асоціативній пам'яті і керування. Традиційні підходи до рішення цих проблем не завжди надають необхідну гнучкість. Багато застосувань виграють від використання нейромереж.

Штучні нейромережі є моделями нейронної структури мозку, який головним чином навчається з досвіду. Природній аналог доводить, що множина проблем, які поки що не підвладні розв'язуванню наявними комп'ютерами, можуть бути ефективно вирішені блоками нейромереж.

Тривалий період еволюції додав мозку людини багато якостей, що відсутні в сучасних комп'ютерах з архітектурою фон Неймана. До них відносяться:

- розподілене представлення інформації і паралельні обчислення;
- здатність до навчання і здатність до узагальнення;
- адаптивність;
- терпимість до помилок
- низьке енергоспоживання.

Прилади, побудовані на принципах біологічних нейронів мають перелічені характеристики, що можна вважати суттєвим здобутком у індустрії обробки даних.

В історії досліджень в галузі нейронних мереж, як і в історії будь-якої іншої науки, були свої успіхи і невдачі.

Здатність нейронної мережі до навчання вперше досліджена Дж. Маккалоком і У. Піттом. У 1943 році вийшла їх робота "Логічне числення ідей, що ві-

дносяться до нервової діяльності", в якій була побудована модель нейрона, і сформульовані принципи побудови штучних нейронних мереж.

В 1959 р. Бернард Відроу (*Bernard Widrow*) та Марсіан Хофф (*Marcian Hoff*) розробили моделі *ADALINE* та *MADALINE* (Множинні Адаптивні Лінійні Елементи (*Multiple ADAptive LINear Elements*)). *MADALINE* діяла, як адаптивний фільтр, що усував відлуння на телефонних лініях. Ця нейромережа й досі в комерційному використанні.

Значний поштовх розвитку нейрокібернетики дав американський нейрофізіолог Френк Розенблат, який запропонував в 1962 році свою модель нейронної мережі - персептрон. Незважаючи на це в 70-ті роки було запропоновано багато цікавих розробок, таких, наприклад, як когнітрон, здатний добре розпізнавати досить складні образи незалежно від повороту і зміни масштабу зображення.

У 1982 році американський біофізик Дж. Хопфілд запропонував оригінальну модель нейронної мережі, названу його ім'ям. У наступні кілька років було знайдено безліч ефективних алгоритмів: мережа зустрічного потоку, двонаправлена асоціативна пам'ять і ін.

У 1990 р. Департамент програм інноваційних досліджень захисту малого бізнесу назвав 16 основних та 13 додаткових тем, де потрібне та можливе використання нейронних мереж.

ГЛАВА 1 ОДНОШАРОВІ НЕЙРОННІ МЕРЕЖІ

1.1 Основні поняття нейронних мереж (НМ)

У останні десятиліття у світі бурхливо розвивається нова прикладна область інформатики, що спеціалізується на штучних нейронних мережах. Актуальність досліджень в цьому напрямі підтверджується масою різних застосувань НС. Це автоматизація процесів розпізнавання образів, адаптивне управління, апроксимація функціоналів, прогнозування, створення експертних систем, організація асоціативної пам'яті і багато інших застосувань. За допомогою НС можна, наприклад, передбачати показники біржового ринку, виконувати розпізнавання оптичних або звукових сигналів, створювати самонавчальні системи, здатні управляти автомашиною при парковці або синтезувати мову за текстом.

Основні властивості нейронних мереж, що відрізняють їх від інших інтелектуальних систем:

Навчання. Штучні нейронні мережі можуть міняти свою поведінку в залежності від зовнішнього середовища. Цей фактор більшою мірою, ніж будь-який інший, відповідальний за той інтерес, який вони викликають. Після пред'явлення вхідних сигналів (можливо, разом з необхідними виходами) вони Самоналагоджувальний, щоб забезпечувати необхідну реакцію. Було розроблено безліч навчальних алгоритми-мов, кожен зі своїми сильними і слабкими сторонами. Як буде зазначено в цій книзі пізніше, всі ще існують проблеми щодо того, чому мережа може навчитися і як навчання повинно проводитися.

Узагальнення. Відгук мережі після навчання може бути до певної міри нечутливість-Телень до невеликих змін вхідних сигналів. Ця внутрішньо притаманна спо-можності бачити образ крізь шум і спотворення життєво важлива для розпізнавання образів в реальному світі. Вона дозволяє подолати вимога суворої точності, що пред'являється звичайним комп'ютером, і відкриває шлях до системи, яка може мати справу з тим недосконалим світом, в якому ми живемо. Важливо відзначити, що штучна нейронна мережа робить узагальнення

автоматично завдяки своїй структурі, а не за допомогою використання «людського інтелекту» в формі спеціально написаних комп'ютерних програм.

Абстрагування. Деякі з штучних нейронних мереж мають здатність через залучати сутність з вхідних сигналів. Наприклад, мережа може бути навчена на послідовність спотворених версій букви «А». Після відповідного навчання пред'явлення такого спотвореного прикладу приведе до того, що мережа породить букву досконалої форми. У певному сенсі вона навчиться породжувати те, що ніколи не бачила. Ця здатність витягувати ідеальне з недосконалих входів ставить інтересні філософські питання. Вона нагадує концепцію ідеалів, висунуту Платоном в його «Республіці».

Застосовність. Штучні нейронні мережі не є панацеєю. Вони, очевидно, не годяться для виконання таких завдань, як нарахування заробітної плати. Схоже, однак, що їм буде віддаватися перевага у великому класі задач розпізнавання образів, з якими погано або взагалі не справляються звичайні комп'ютерні системи.

1.2 Застосування нейронних мереж

Класифікація образів. Завдання полягає у визначенні приналежності вхідного образа (наприклад, мовного сигналу чи рукописного символу), представленого вектором ознак, одному чи декільком попередньо визначеним класам. До відомих застосувань відносяться розпізнавання букв, розпізнавання мови, класифікація сигналу електрокардіограми, класифікація кліток крові.

Кластеризація / категоризація. При рішенні задачі кластеризації, що відома також як класифікація образів "без вчителя", навчальна множина з визначеними класами відсутня. Алгоритм кластеризації заснований на подобі образів і розміщує близькі образи в один кластер. Відомі випадки застосування кластеризації для видобутку знань, стиснення даних і дослідження властивостей даних.

Апроксимація функцій. Припустимо, що є навчальна вибірка ((x_1,y_1), (x_2,y_2)..., (x_n,y_n)) (пари даних вхід-вихід), яка генерується невідомою функцією F,

спотвореної шумом. Завдання апроксимації полягає в знаходженні невідомої функції F. Апроксимація функцій необхідна при рішенні численних інженерних і наукових задач моделювання.

Передбачення/прогноз. Нехай задані n дискретних відліків $\{y(t_1), y(t_2), ..., y(t_n)\}$ у послідовні моменти часу $t_1, t_2,..., t_n$. Завдання полягає в передбаченні значення $y(t_n+1)$ у деякий майбутній момент часу t_n+1. Передбачення/прогноз мають значний вплив на прийняття рішень у бізнесі, науці і техніці (передбачення цін на фондовій біржі, прогноз погоди).

Оптимізація. Численні проблеми в математиці, статистиці, техніці, науці, медицині й економіці можуть розглядатися як проблеми оптимізації. Задачею алгоритму оптимізації є знаходження такого рішення, що задовольняє системі обмежень і максимізує чи мінімізує цільову функцію.

Пам'ять, що адресується за змістом. В традиційних комп'ютерах звертання до пам'яті доступно тільки за допомогою адреси, що не залежить від змісту пам'яті. Більш того, якщо допущена помилка в обчисленні адреси, то може бути знайдена зовсім інша інформація. Асоціативна пам'ять, чи пам'ять, що адресується за змістом, доступна за вказівкою заданого змісту. Вміст пам'яті може бути викликано навіть по частковому входу чи спотвореному змісту. Асоціативна пам'ять надзвичайно бажана при створенні мультимедійних інформаційних баз даних.

Керування. Розглянемо динамічну систему, задану сукупністю $\{u(t), y(t)\}$, де $u(t)$ є вхідним керуючим впливом, а $y(t)$ - виходом системи в момент часу t. В системах керування з еталонною моделлю метою керування є розрахунок такого вхідного впливу $u(t)$, при якому система діє по бажаній траєкторії, заданою еталонною моделлю. Прикладом є оптимальне керування двигуном.

Але, незважаючи на переваги нейронних мереж в окремих галузях над традиційними обчисленнями, існуючі нейромережі є не досконалими рішеннями. Вони навчаються і можуть робити "помилки". Окрім того, не можна гарантувати, що розроблена мережа є оптимальною мережею. Застосування нейромереж вимагає від розробника виконання ряду умов.

Ці умови включають:

- множину даних, що включає інформацію, яка може характеризувати проблему;
- відповідно встановлену за розміром множину даних для навчання і тестування мережі;
- розуміння базової природи проблеми, яка буде вирішена;
- вибір функції суматора, передатної функції та методів навчання;
- розуміння інструментальних засобів розробника;
- відповідна потужність обробки.

Новий шлях обчислень вимагає вмінь розробника поза межами традиційних обчислень. Спочатку, обчислення були лише апаратними і інженери робили його працюючим. Потім, були спеціалісти з програмного забезпечення: програмісти, системні інженери, спеціалісти по базах даних та проектувальники. Тепер є нейронні архітектори. Новий професіонал повинен мати кваліфікацію відмінну від його попередників. Наприклад, він повинен знати статистику для вибору і оцінювання навчальних і тестових множин. Логічне мислення сучасних інженерів програмного забезпечення, їх емпіричне вміння та інтуїтивне відчуття гарантує створення ефективних нейромереж.

У київському інституті кібернетики з 70-х років ведуться роботи над стохастичними нейронними мережами.

1.3 Біологічний прототип

Нейрон, як і всі інші клітини, складається з ядра та цитоплазми (рис.1). Тіло клітини (сома) від навколишнього середовища відокремлене тонкою мембраною завтовшки 75 ангстрем, що складається з ліпідів та характеризується низькою провідністю. У мембрану вбудовані білкові молекули, які виконують функції рецепторів та іонних каналів. Для внутрішнього складу клітини характерною є висока концентрація іонів калію та низька концентрація іонів натрію.

Таке співвідношення є результатом роботи натрій-калієвого насоса, на функціонування якого витрачається накопичена клітиною енергія. Спеціальні транспортні білкові молекули іонних каналів захоплюють іони натрію з внутрішньої сторони мембрани та переміщують на зовнішню сторону. Аналогічно іони калію захоплюються на зовнішній стороні мембрани та переміщуються на внутрішню. Результатом роботи такого механізму є різниця потенціалів (приблизно –60мВ) між внутрішньою частиною нейрона та зовнішнім середовищем. Нейрони характеризуються також наявністю у них спеціальних відростків. По одних відростках нейрони одержують інформацію, а по других передають сигнали іншим нейронам. По деревовидних відростках (дендритах) нейрон отримує інформацію через спеціальні контакти (синапси). Передача інформації від одного нейрона до іншого відбувається шляхом розповсюдження нервового імпульсу вздовж нервового волокна–аксона.

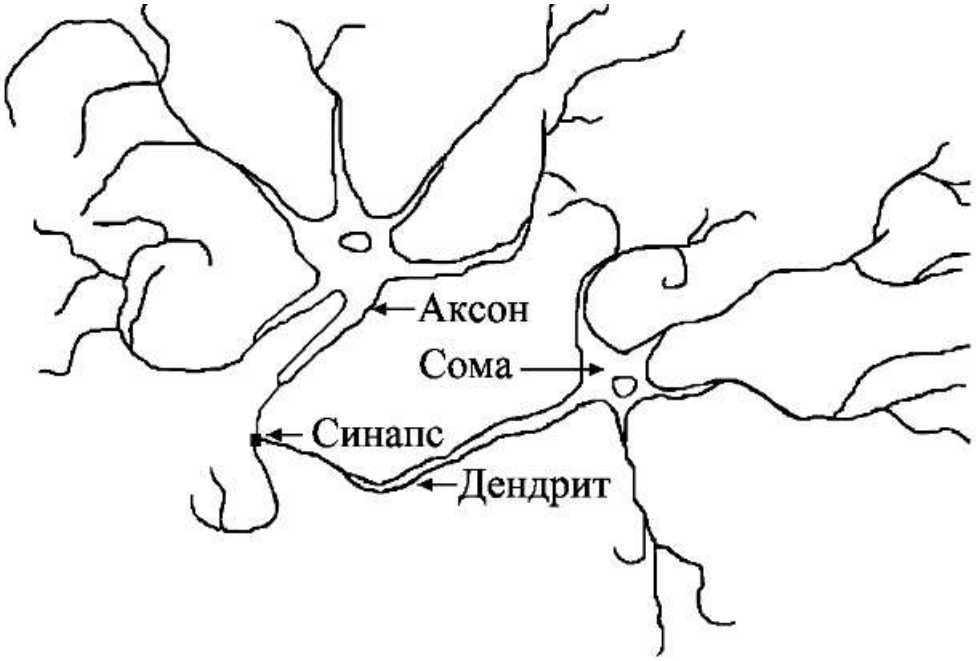

Рис. 1 Біологічні нейрони

Кожен нейрон може мати велику кількість дендритів і тільки один аксон. Волокно аксона розгалужується і створює з’єднання з дендритами інших нейронів через відповідні синаптичні контакти. Під дією сигналів з інших нейронів або рецепторів мембрана аксона змінює свою провідність. Це відбувається, ко-

ли внутрішній потенціал перевищує значення близько –50 мВ. Механізм розповсюдження нервового імпульсу базується на роботі потенціало залежних іонних каналів, які створюються за допомогою складних білкових молекул. При деполяризації, тобто зміщенні мембранного потенціалу у бік зменшення від'ємного заряду в середині клітини, відкриваються потенціало залежні натрієві канали, і натрій по хімічному і електричному градієнту входить у клітину. Тривалість цього стану складає близько 1–3 мс. Після цього натрієві канали починають закриватися і одночасно відкриваються потенціало залежні калієві канали. Калій по хімічному градієнту виходить із клітини, внаслідок чого збільшується її від'ємний заряд.

Нейронні мережі: означення та основні властивості який супроводжується потенціалом дії близько +40 мВ, розповсюджується вздовж аксона майже без затухання зі швидкістю від 100 до 1000 сантиметрів у секунду. Після проходження спайка настає період рефрактерності, який характеризується повною пасивністю нейрона протягом 200мс. Таке розповсюдження спайка вздовж аксона часто порівнюють із розповсюдженням зони горіння у бікфордовому шнурі. Відповідно до цієї аналогії зона горіння відповідає спайку, а область згорілої порохової суміші це стан рефрактерності. Частина аксона, що лежить попереду, відповідає ще не активованим натрієвим каналам. Як уже згадувалось, взаємодія нейронів відбувається через спеціальні контакти — синапси. Синапси бувають електричні та хімічні. Електричний синапс — це ділянка з'єднання двох нейронів, яка характеризується високою провідністю. Внаслідок цього електричний струм, що протікає в одному нейроні, може протікати і в другому. Такий зв'язок менш інформативний, ніж хімічний, але характеризується високою швидкістю взаємодії. Тому основним його призначенням є синхронізація пула нейронів. Хімічний синапс значно складніший у порівнянні з електричним і складається з пресинаптичної частини однієї клітини та постсинаптичної частини другої клітини розділені між собою синаптичною щілиною. Пресинаптична частина складається з декількох бутонів, якими закінчується аксон. На кожному з бутонів розташовані ділянки викиду трансмітера, який знаходиться в

бульбашках. Кожна така бульбашка містить один квант трансмітера. Постсинаптична частина — це ділянка мембрани постсинаптичної клітини, яка містить білки, що є рецепторами трансмітера. Характерною особливістю пресинаптичної частини є наявність потенціал озалежних кальцієвих каналів. Спайк, досягаючи кінця аксона, відкриває кальцієві канали, в результаті чого кальцій з навколишнього середовища входить в аксоні запускає низку кальцій залежних біохімічних реакцій. Результатом цих реакцій є вилив у синаптичну частину певної кількості квантів.

Теорема Колмогорова-Арнольда (про яку часто не підозрюють практики) служить математичною основою нейронних мереж.

Теорема Колмогорова стверджує, що будь-яка безперервна функція *f,* визначена на n - вимірному одиничному кубі, може бути представлена у вигляді суми 2n+1 суперпозицій безперервних і монотонних відображень одиничних відрізків :

$$f(x_1,...,x_n) = \sum_{q=1}^{2n+1} g_q\left(\sum_{p=1}^{n} \phi_{pq}(x_p)\right)$$

$$x = (x1, .., xn), \quad 0 \le xi \le 1$$

Ліворуч в цій формулі стоїть довільна неперервна функція, визначена на багатовимірному кубі, справа функції визначені на відрізках [0, 1].

Звичайно, треба мати на увазі, що відображення ***q***мають досить складну структуру.

Зверніть увагу, що монотонні безперервні відображення ***φ*** не залежать від конкретної функції ***f.***

У світлі цієї теореми завдання побудови і навчання мережі ставиться на строго математичну основу.

1.4 Класифікація нейронних мереж

Можна провести наступну класифікацію нейронних мереж:

Класифікація нейронних мереж за характером навчання ділить їх на:

- нейронні мережі, що використовують навчання з учителем;
- нейронні мережі, що використовують навчання без учителя.

Розглянемо це докладніше.

Нейронні мережі, що використовують навчання з учителем. Навчання з учителем передбачає, що для кожного вхідного вектора існує цільовий вектор, що представляє собою необхідний вихід. Разом вони називаються навчальною парою. Зазвичай мережа навчається на деякому числі таких навчальних пар. Пред'являється вихідний вектор, обчислюється вихід мережі і порівнюється з відповідним цільовим вектором. Далі ваги змінюються відповідно до алгоритму, що прагнуть мінімізувати помилку. Вектори навчальної множини пред'являються послідовно, обчислюються помилки і ваги налаштовуються для кожного вектора доти, поки помилка по всьому навчальному масиву не досягне прийнятного рівня.

Нейронні мережі, що використовують навчання без учителя. Навчання без вчителя є набагато більш правдоподібною моделлю навчання з точки зору біологічних коренів штучних нейронних мереж. Розвинена Кохоненом і багатьма іншими, вона не потребує цільового векторі для виходів і, отже, не вимагає порівняння з зумовленими ідеальними відповідями. Навчальна множина складається лише з вхідних векторів. Навчальний алгоритм налаштовує ваги мережі так, щоб виходили узгоджені вихідні вектори, т. Е. Щоб пред'явлення досить близьких вхідних векторів давало однакові виходи. Процес навчання, отже, виділяє статистичні властивості навчальної множини і групує подібні вектори в класи.

Налаштування ваг мережі з фіксованими зв'язками:

- вагові коефіцієнти нейронної мережі вибираються відразу, виходячи з умов завдання;

- мережі з динамічними зв'язками - для них в процесі навчання відбувається настройка синаптичних ваг.

Тип вхідної інформації:

- аналогова - вхідна інформація представлена в формі дійсних чисел;
- двійкова - вся вхідна інформація в таких мережах представляється у вигляді нулів і одиниць.

Деякі моделі нейронних мереж:

- **Мережі прямого поширення** - всі зв'язки направлені строго від вхідних нейронів до вихідних. До таких мереж відносяться, наприклад: найпростіший персептрон (розроблений Розенблатта) і багатошаровий персептрон.
- **Рекурентні нейронні мережі** - сигнал з вихідних нейронів або нейронів прихованого шару частково передається назад на входи нейронів вхідного шару.
- **Радіально базисні функції** - вид нейронної мережі, що має прихований шар з радіальних елементів і вихідний шар з лінійних елементів. Мережі цього типу досить компактні і швидко навчаються. **Мережі Кохонена** або карти, що само організуються - такий клас мереж, як правило, навчається без учителя і успішно застосовується в задачах розпізнавання. Мережі такого класу здатні виявляти новизну у вхідних даних: якщо після навчання мережа зустрінеться з набором даних, несхожим ні на один з відомих зразків, то вона не зможе класифікувати такий набір і тим самим виявить його новизну.

1.5 **Навчання одношарової мережі**

Для того, щоб нейронна мережа мала здатність вирішувати конкретну задачу, тобто на кожен вхідний сигнал видавати необхідний вихідний сигнал, необхідно провести налаштування параметрів мережі. Налаштування проводиться за повчальною вибіркою, яка складається з пар (вхід, бажаний вихід) - повчаль-

них прикладів. Залежно від вирішуваного завдання в навчальній вибірці використовуються ті або інші типи даних і різної розмірності вхідних-вихідних сигналів.

Вхідні дані прикладів навчальної вибірки - зображення, таблиці чисел, розподіли. Типи вхідних даних - бінарні (0 і 1), біполярні (-1 і 1) числа, цілі або дійсні числа з деякого діапазону.

Вихідні сигнали мережі - вектори цілих або дійсних чисел. Для вирішення практичних завдань часто потрібні повчальні вибірки великого об'єму. Через жорстко обмежений об'єм оперативної пам'яті комп'ютера розмістити в ній великі повчальні вибірки неможливо. Тому вибірка ділиться на сторінки - групи прикладів. У кожен момент часу лише одна сторінка прикладів розташовується в пам'яті комп'ютера, останні - на жорсткому диску. Сторінки послідовно завантажуються в пам'ять комп'ютера.

Навчання мережі відбувається по всій сукупності сторінок прикладів, по всій навчальній вибірці. В даний час відсутня універсальна методика побудови навчальних вибірок. Набір навчальних прикладів формується з розсуду користувача програми індивідуально для кожного конкретного вирішуваного завдання.

Якщо в ненавчену нейронну мережу ввести вхідний сигнал один з прикладів навчальної вибірки, то вихідний сигнал мережі істотно відрізнятиметься від бажаного вихідного сигналу, визначеного в навчальній вибірці. Функція помилки чисельно визначає схожість всіх поточних вихідних сигналів мережі і відповідних бажаних вихідних сигналів повчальної вибірки. Найбільш поширеною функцією помилки є середньоквадратичне відхилення. Проте запропоновані і інші функції помилки.

Мета навчання - мінімізувати функцію помилки, тобто знайти такі значення параметрів мережі, при яких поточні вихідні сигнали мережі мінімально відрізняються від відповідних бажаних вихідних сигналів, заданих повчальною вибіркою. Для навчання нейронних мереж можуть бути використані різні алгоритми. Можна виділити дві великі групи алгоритмів - градієнтні і стохастичні.

Градієнтні алгоритми навчання мереж засновані на обчисленні приватних похідних функції помилки по параметрах мережі.

Серед градієнтних розрізняють алгоритми першого і другого порядків. У стохастичних алгоритмах пошук мінімуму функції помилки ведеться випадковим чином.

При навчанні нейронних мереж, як правило, використовується один з двох наступних критеріїв останову:

• останов при досягненні деякого малого значення функції помилки,

• останов у разі успішного вирішення всіх прикладів повчальної вибірки.

Перед навчанням виконується ініціалізація нейронної мережі, тобто присвоєння параметрам мережі деяких початкових значень. Як правило, ці початкові значення - деякі малі випадкові числа.

Для формування навчальних вибірок, ініціалізації і навчання в програмах моделювання нейронних мереж використовуються спеціальні процедури. Можливість використання багато сторінкового навчання є дуже важливою при вирішенні практичних завдань за допомогою нейронних мереж, що моделюються на звичайних комп'ютерах.

Навчання - це ітераційна процедура, яка при реалізації на звичайних комп'ютерах, вимагає значного часу. Алгоритми навчання істотно розрізняються за швидкістю збіжності. Одній з найважливіших характеристик програм для моделювання нейронних мереж є швидкість збіжності алгоритму (або алгоритмів) навчання, які реалізовані в програмі.

Моделі НС можуть бути програмного і апаратного виконання. Надалі мова піде в основному про перший тип.

Незважаючи на істотні відмінності, окремі типи НС мають декілька загальних рис.

По-перше, основу кожної НС складають відносно прості, в більшості випадків - однотипні, елементи (осередки), що імітують роботу нейронів мозку.

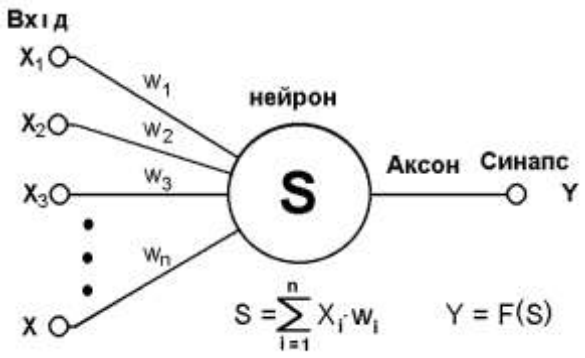

Рис.2 Штучний нейрон

Далі під нейроном матиметься на увазі штучний нейрон, тобто осередок НС. Кожен нейрон характеризується своїм поточним станом по аналогії з нервовими клітинами головного мозку, які можуть бути збуджені або загальмовані. Він має групу синапсів - однонапрямлених вхідних зв'язків, сполучених з виходами інших нейронів, а також має аксон - вихідний зв'язок цього нейрона, з яким сигнал (збудження або гальмування) поступає на синапси наступних нейронів. Загальний вигляд нейрона приведений на малюнку 1. Кожен синапс характеризується величиною синаптичного зв'язку або її вагою wi, яка по фізичному сенсу еквівалентна електричній провідності.

Поточний стан нейрона визначається, як зважена сума його входів :

$$s = \sum_{i=1}^{n} x_i \cdot w_i$$

Вихід нейрона є функція його стану :

$$y = F(S)$$

Нелінійна функція F називається активаційною і може мати різний вигляд, як показано на рис. 3. Однією з найбільш розповсюджених є нелінійна функція з насиченням, так звана логістична функція або сигмоїд (тобто функція S - подібного виду):

$$F(x) = \frac{2}{1+e^{-\alpha x}} - 1 \qquad (1.1)$$

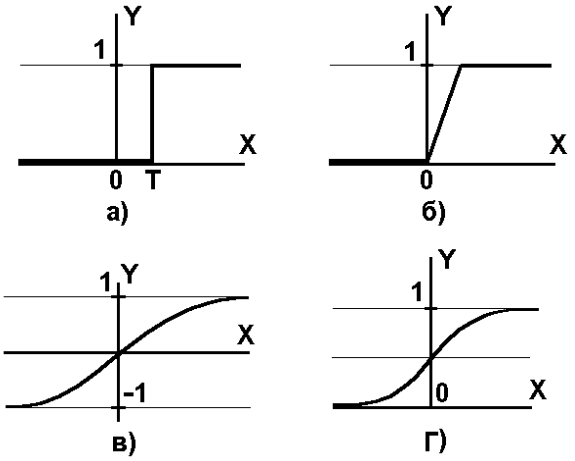

Рис.3 Активаційна функція

а) функція одиничного скачка; б) лінійний поріг; в) сигмоїд - гіперболічний тангенс; г) сигмоїд - формула (3)

При зменшенні (сигмоїд стає пологішим, в межі при (=0 вироджуючись в горизонтальну лінію на рівні 0.5, при збільшенні (сигмоїд наближається на вигляд до функції одиничного стрибка з порогом Т в точці х=0. З вираження для сигмоїда очевидно, що вихідне значення нейрона лежить в діапазоні [0,1]. Одна з цінних властивостей сигмоїдной функції - просте вираження для її похідної, застосування якого буде розглянуто надалі.

$$F(x) = \alpha \cdot F(x) \cdot (1 - F(x))$$

Слід зазначити, що сигмоїдна функція диференційована на усій осі абсцис, що використовується в деяких алгоритмах навчання. Крім того вона має властивість посилювати слабкі сигнали краще, ніж великі, і запобігає насиченню від

великих сигналів, оскільки вони відповідають областям аргументів, де сигмоїд має пологий нахил.

Повертаючись до загальних рис, властивих усім НС, відмітимо, по-друге, принцип паралельної обробки сигналів, який досягається шляхом об'єднання великого числа нейронів в так звані шари і з'єднання певним чином нейронів різних шарів, а також, в деяких конфігураціях, і нейронів одного шару між собою, причому обробка взаємодії усіх нейронів ведеться пошарово.

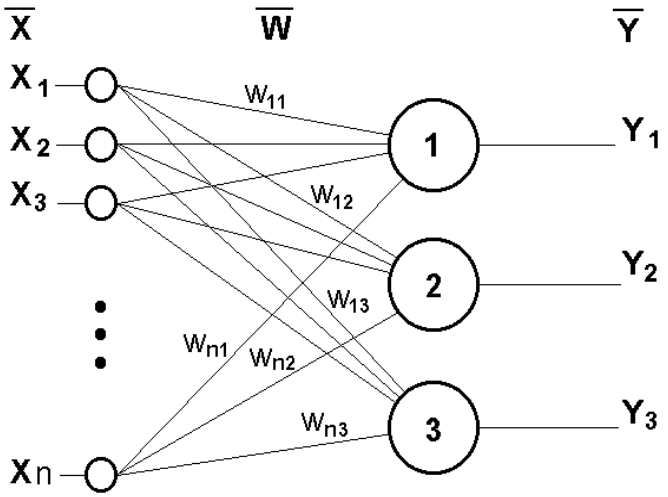

Рис.4 Одношаровий персептрон

Як приклад простої НС розглянемо тринейронний персептрон (рис.4), тобто таку мережу, нейрони якої мають активаційну функцію у вигляді одиничного стрибка* . На n входів поступають деякі сигнали, що проходять по синапсах на 3 нейрони, що утворюють єдиний шар цієї НС і що видають три вихідних сигналу, :

$$y_j = F\left[\sum_{i=1}^{n} x_i \cdot w_{ij}\right], \quad j=1..3$$

Очевидно, що усі вагові коефіцієнти синапсів одного шару нейронів можна звести в матрицю **W**, в якій кожен елемент w_{ij} задає величину і -го синаптичного зв'язку j -го нейрона. Таким чином, процес, що відбувається в НС, може бути записаний у векторній формі:

$$\mathbf{Y} = F(\mathbf{S})$$

де **X** і **Y** - відповідно до вхідної і вихідної сигнальні вектори, F(**S**) - активаційна функція, що застосовується поелементно до компонент вектору **S**.

Теоретично число шарів і число нейронів в кожному шарі може бути довільним, проте фактично воно обмежене ресурсами комп'ютера або спеціалізованої мікросхеми, на яких зазвичай реалізується НС. Чим складніше НС, тим масштабно завдання, підвладні їй.

Розглянемо сигнальний алгоритм навчання з учителем, запропонований Хеббом і доопрацьований Відроу .

1. Проініціалізувати елементи вагової матриці (зазвичай невеликими випадковими значеннями).

2. Подати на входи один з вхідних векторів, які мережа повинна навчитися розрізняти, і вичислити її вихід.

3. Якщо вихід правильний, перейти на крок 4.

Інакше обчислити різницю між ідеальним і отриманим значеннями виходу :

$$\delta = Y_I - Y \tag{1.2}$$

Модифікувати ваги відповідно до формули:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \cdot \delta \cdot x_i \tag{1.3}$$

де t і t+1 - номери відповідно до поточної і наступної ітерацій;

η - коефіцієнт швидкості навчання;

i - номер входу;

j - номер нейрона в шарі.

Очевидно, що якщо YI > Y вагові коефіцієнти будуть збільшені і тим самим зменшать помилку. Інакше вони будуть зменшені, і Y теж зменшиться, наближаючись до YI.

4. Цикл з кроку 2, поки мережа не перестане помилятися.

На другому кроці на різних ітераціях по черзі у випадковому порядку пред'являються усі можливі вхідні вектору. На жаль, не можна заздалегідь визначити число ітерацій, які потрібно буде виконати, а в деяких випадках і гарантувати повний успіх.

1.6 Проблема лінійного розподілу в НМ

Часто, для того, щоб продемонструвати обмежені можливості одношарових персептронів при рішенні завдань удаються до розгляду так званої проблеми XOR - що виключає АБО.

Суть завдання полягають в наступному. Дана логічна функція XOR - що виключає АБО. Це функція від двох аргументів, кожен з яких може бути нулем або одиницею. Вона набуває значення, коли один з аргументів дорівнює одиниці, але не обоє, інакше . Проблему можна проілюструвати за допомогою одношарової однонейронної системи з двома входами, показаної на малюнку нижче.

Позначимо один вхід через, а інший через, тоді усі їх можливі комбінації складатимуться з чотирьох точок на площині.

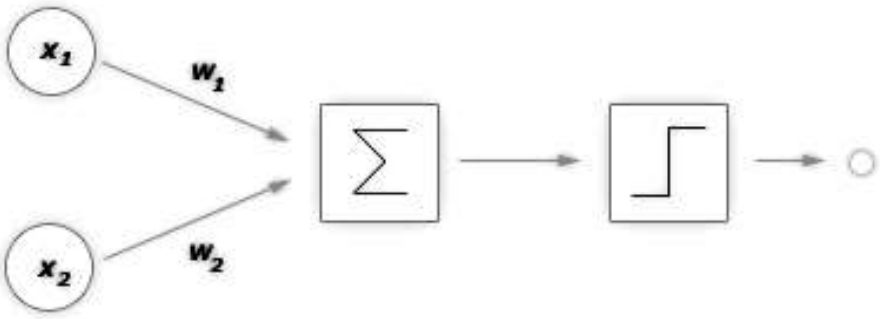

Рис.5 НМ для представлення проблеми XOR

Таблиця нижче показує необхідний зв'язок між входами і виходом, де вхідні комбінації, які повинні давати нульовий вихід, помічені Ai, одиничний вихід - Bi .

Точки	Значення	Значення	Необхідний вихід
A0	0	0	0
B0	1	0	1
A1	0	1	1
B1	1	1	0

Один нейрон з двома входами може сформувати вирішальну поверхню у вигляді довільної прямої:

$$X_1 W_1 + X_2 W_2 = T$$

Для того, щоб мережа реалізувала функцію XOR, задану таблицею вище, треба розташувати пряму так, щоб точки Ai були з одного боку прямої, а точки Bi - з іншого. Спробувавши намалювати таку пряму на малюнку , переконуємося, що це неможливо.

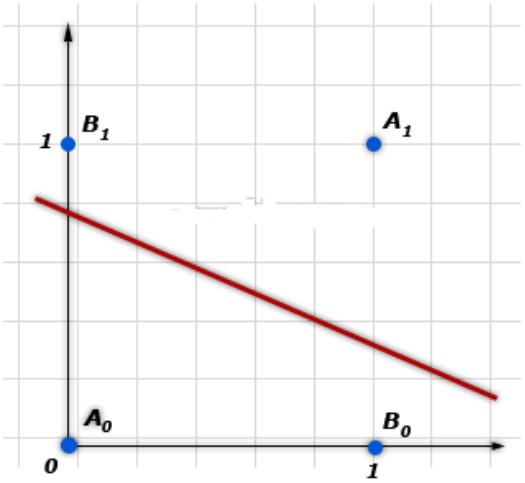

Рис. 6 Лінійний не розподіл задачі XOR

Це означає, що які б значення не приписувалися вагам і порогу, одношарова нейронна мережа нездатна відтворити співвідношення між входом і виходом, потрібне для представлення функції XOR.

Проте функція XOR легко формується вже двошаровою мережею, причому багатьма способами. Розглянемо один з таких способів. Модернізуємо мережу, додавши ще один прихований шар нейронів (рис.7):

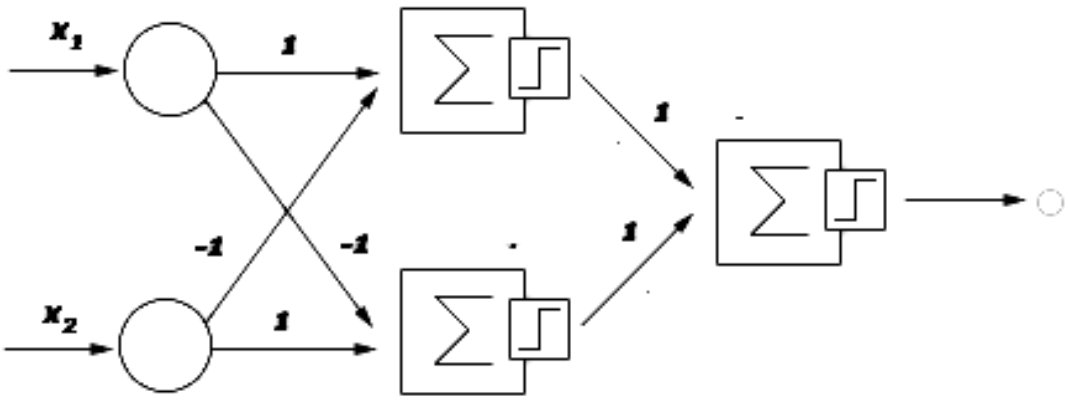

Рис. 7 НМ для представлення вирішення проблеми XOR

Відмітимо, що ця мережа дана як є, тобто можна вважати, що вона вже навчена. Цифри над стрілками показують значення синаптических ваг. Як функція активації застосуємо функцію одиничного стрибка з порогом, що має наступний графік, (рис.8):

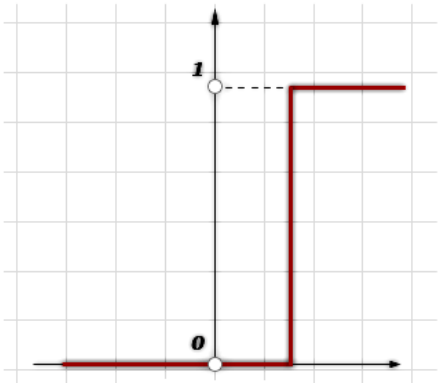

Рис. 8 Пороговий елемент НМ

Тоді результат роботи такої нейронної мережі можна представити у вигляді наступної таблиці:

Точки	Значення	Значення	Необхідний вихід	Y1	Y2	Y
A0	0	0	0	0	0	0
B0	1	0	1	1	0	1
A1	0	1	1	0	1	1
B1	1	1	0	0	0	0

Кожен з двох нейрон першого шару формує вирішальну поверхню у вигляді довільної прямої (ділить площину на дві напівплощини), а нейрон вихідного шару об'єднує ці два рішення, утворюючи вирішальну поверхню у вигляді смуги, утвореної паралельними прямими нейронів першого шару, (рис.9):

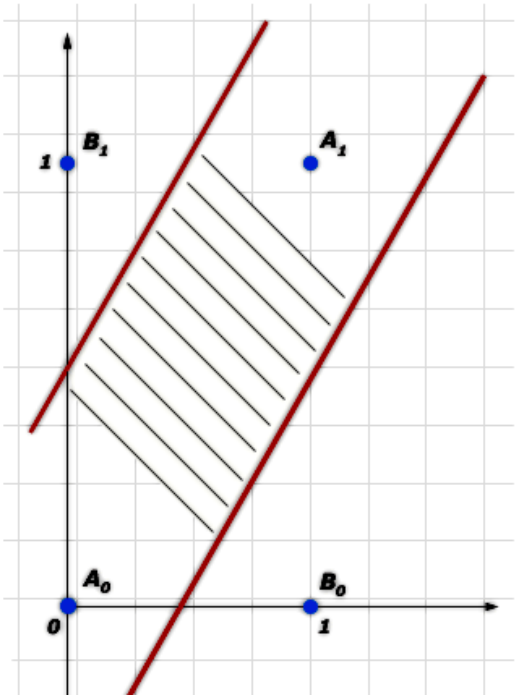

Рис. 9 Лінійний розподіл задачі XOR

Нейронна мережа для вирішення завдання XOR, примітивна і не використовує усіх можливостей багатошарових мереж. Очевидно, що багатошарові нейронні мережі мають більшу потужність, чим одношарові, тільки у разі присутності нелінійності.

1.7 Теми та алгоритми лабораторних робіт

Лабораторна робота №1

Завдання: Побудувати бінарний класифікатор з допомогою моделі штучного нейрону.

Алгоритм лабораторної роботи №1

1. Обираємо задачу класифікації образів за неявними ознаками $\mathbf{A_j}$ та готуємо базу прикладів для навчання (навчальну вибірку) у вигляді таблиці:

Xpi	**A1**	**A2**	**...**	**An**	**dp**
P1	X_{11}	X_{12}		X_{1n}	d_1
P2	X_{21}	X_{22}		X_{2n}	d_2
...					
$\mathbf{P_M}$	X_{M1}	X_{M2}		X_{Mn}	d_M

Де X_{pi} - числові значення ознаки **Aj** для прикладу $\mathbf{P_p}$,

d_p - ознака класу, $d_p = +1$ – I клас, $d_p = -1$ – II клас.

Число ознак повинно бути близько 5, а число прикладів більше 10, контрольна вибірка близько 5.

2. Виконуємо природну нормалізацію ознак:

$$\overline{X}_{pi} = \frac{X_{pi} - X_{min\,i}}{X_{max\,i} - X_{min\,i}}$$

3. Ініціалізуємо нейрон:

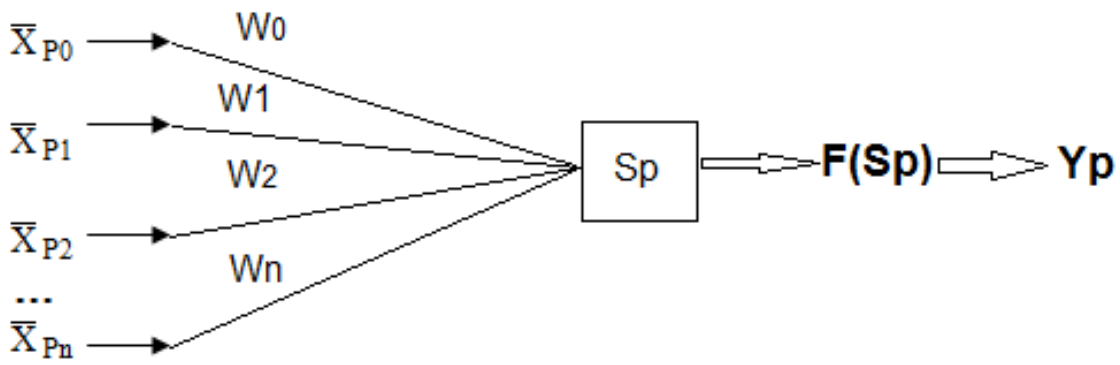

Рис.10 Модель нейрона

Початкові значення вагових параметрів –невеликі, випадкові:

$$Wi = 0.01*rand(-1,+1).$$

Параметр ітерації: t=0.

Функцію активації приймаємо у вигляді:

$$F(S) = \frac{2}{1+e^{-aS}} - 1,$$

де $a=2$.

Задаємо параметр швидкості навчання $\eta = 0.9$ і точність навчання ε =0.01.

$\overline{X}_{P0} = 1$

4. Основний цикл навчання за методом Відроу-Хебба:

а) Обираємо випадково новий приклад з навчальної вибірки з ознаками $\overline{X}_{pi}$ і обчислюємо зважену суму, вихідне значення і помилку:

$$Sp = \sum_{i=0}^{n} Wi \cdot \overline{X}pi$$

$$Yp = F(Sp),$$

$$\Delta p = (Yp - dp)$$

b) Обчислюємо вагові параметри для ітерації (t+1):

$$Wi(t+1) = Wi(t) - \eta * \overline{X}_{pi} * \Delta p$$

c) Повторити b) для i=1,2,...,n.

d) Перейти до а) до вичерпання всіх прикладів навчальної вибірки.

e) Обчислюємо сумарну помилку:

$$E = \frac{1}{2}\sum_{p=1}^{M}\Delta_p^2$$

f) При умові E > ε^2 продовжуємо ітерації *t=t+1*, переходимо до а),
при умові E <= ε^2 - вихід.

Лабораторна робота № 2

З допомогою одношарової мережі з двома нейронами на виході класифікувати навчальну вибірку із лабораторної роботи №1, використовуючи навчання без учителя сигнальним методом Хебба.

Алгоритм лабораторної роботи № 2

1. На стадії ініціалізації усім ваговим коефіцієнтам привласнюються невеликі випадкові значення.
2. На входи мережі подається вхідний образ, і сигнали збудження поширюються по усіх шарах згідно з принципами класичних прямопоточних (feedforward) мереж, тобто для кожного нейрона розраховується зважена сума його входів, до якої потім застосовується активаційна (передавальна) функція нейрона, внаслідок чого виходить його вихідне значення

$$y_i^{(n-1)},\ i=1..M_i,$$

де M_i - число нейронів в шарі i; n=1..N, а N - число шарів в мережі.

3. На підставі набутих вихідних значень нейронів по формулах

$$w_{ij}(t)=w_{ij}(t-1)+\eta\, y_i^{(n-1)}\, y_j^{(n)}$$

або

$$w_{ij}(t)=w_{ij}(t-1)+\eta\,[y_i^{(n-1)}(t) - y_j^{(n-1)}(t-1)][y_i^{(n)}(t) - y_j^{(n)}(t-1)]$$

виконується зміна вагових коефіцієнтів.

4. Цикл з кроку 2, поки вихідні значення мережі не стабілізуються із заданою точністю. Застосування цього нового способу визначення завершення навчання, відмінного від зворотного поширення, що використалося для мережі, обумовлене тим, що налаштовувані значення синапсів фактично не обмежені.

На другому кроці циклу поперемінно пред'являються усі образи з вхідного набору.

Слід зазначити, що вид відгуків на кожен клас вхідних образів не відомий заздалегідь і буде довільним поєднанням станів нейронів вихідного шару, обумовленим випадковим розподілом ваг на стадії ініціалізації. В той же час, мережа здатна узагальнювати схожі образи, відносячи їх до одного класу.

Тестування навченої мережі дозволяє визначити топологію класів у вихідному шарі. Для приведення відгуків навченої мережі до зручного представлення можна доповнити мережу одним шаром, який, наприклад, по алгоритму навчання одношарового персептрона необхідно змусити відображувати вихідні реакції мережі в необхідні образи.

ГЛАВА 2 БАГАТОШАРОВІ НЕЙРОННІ МЕРЕЖІ

2.1 Навчання багатошарових НМ

Серед різних структур нейронних мереж (НС) однією з найбільш відомих являється багатошарова структура, в якій кожен нейрон довільного шару пов'язаний з усіма аксонами нейронів попереднього шару або, у разі першого шару, з усіма входами НС. Такі НС називаються повнозв'язними.

Коли в мережі тільки один шар, алгоритм її навчання з учителем досить очевидний, оскільки правильні вихідні стани нейронів єдиного шару свідомо відомі, і підстроювання синаптичних зв'язків йде в напрямі, що мінімізує помилку на виході мережі. За цим принципом будується, наприклад, алгоритм навчання одношарового персептрона. У багатошарових же мережах оптимальні вихідні значення нейронів усіх шарів, окрім останнього, як правило, не відомі, і дво або більше шаровий персептрон вже неможливо навчити, керуючись тільки величинами помилок на виходах НС.

Один з варіантів вирішення цієї проблеми - розробка наборів вихідних сигналів, відповідних вхідним, для кожного шару НС, що, звичайно, є дуже трудомісткою операцією і не завжди здійснене.

Другий варіант - динамічне підстроювання вагових коефіцієнтів синапсів, в ході якої вибираються, як правило, найбільш слабкі зв'язки і змінюються на малу величину в ту або іншу сторону, а зберігаються тільки ті зміни, які спричинили зменшення помилки на виході усієї мережі. Очевидно, що цей метод , незважаючи на свою уявну простоту, вимагає громіздких рутинних обчислень.

Нарешті, третій, прийнятніший варіант - поширення сигналів помилки від виходів НС до її входів, в напрямі, зворотному прямому поширенню сигналів в звичайному режимі роботи . Цей алгоритм навчання НС дістав назву процедури зворотного поширення (back propogation). Саме він буде розглянутий надалі.

Згідно з методом найменших квадратів, цільовою функцією помилки НС, що мінімізується, є величина:

$$E(w) = \frac{1}{2}\sum_{j,p}(y_{j,p}^{(N)} - d_{j,p})^2 \qquad (2.1)$$

де $y_{j,p}^{(N)}$ - реальний вихідний стан нейрона j вихідного шару N нейронної мережі при подачі на її входи p -го образу;

d_{jp} - ідеальний (бажане) вихідний стан цього нейрона.

Підсумовування ведеться по усіх нейронах вихідного шару і по усіх оброблюваних мережею образах. Мінімізація ведеться методом градієнтного спуску, що означає підстроювання вагових коефіцієнтів таким чином:

$$\Delta w_{ij}^{(n)} = -\eta \cdot \frac{\partial E}{\partial w_{ij}} \qquad (2.2)$$

Тут w_{ij} - ваговий коефіцієнт синаптичного зв'язку, що сполучає і –й нейрон шару n – 1 з j –м нейроном шару n, η - коефіцієнт швидкості навчання, 0<(<1.

Очевидно що,

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{ds_j} \cdot \frac{\partial s_j}{\partial w_{ij}} \qquad (2.3)$$

Тут під yj, як і раніше, мається на увазі вихід нейрона j, а під s_j – зважена сума його вхідних сигналів, тобто аргумент активаційної функції. Оскільки множник dy_j/ds_j є похідним цій функції по її аргументу, з цього виходить, що похідна активаційної функція має бути визначена на усій осі абсцис. У зв’язку з цим функція одиничного стрибка і інші активаційні функції з не підходять для даних НС. У них застосовуються такі гладкі функції, як гіперболічний тангенс або класичний сигмоїд з експонентою. У разі гіперболічного тангенса

$$\frac{dy}{ds} = 1 - s^2 \qquad (2.4)$$

Третій множник $(s_j/(w_{ij})$, очевидно, дорівнює виходу нейрона попереднього шару $y_i(n - 1)$.

Що стосується першого множника в (3), він легко розкладається таким чином:

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{ds_k} \cdot \frac{\partial s_k}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{ds_k} \cdot w_{jk}^{(n+1)} \qquad (2.5)$$

Тут підсумовування по k виконується серед нейронів шару n+1.

Ввівши нову змінну

$$\delta_j^{(n)} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{ds_j} \qquad (2.6)$$

ми отримаємо рекурсивну формулу для розрахунків величин (j(n) шару n з величин (k(n+1) більше старшого шару n+1.

$$\delta_j^{(n)} = \left[\sum_k \delta_k^{(n+1)} \cdot w_{jk}^{(n+1)} \right] \cdot \frac{dy_j}{ds_j} \qquad (2.7)$$

Для вихідного ж шару

$$\delta_l^{(N)} = (y_l^{(N)} - d_l) \cdot \frac{dy_l}{ds_l} \qquad (2.8)$$

Тепер ми можемо записати (2) в розкритому виді:

$$\Delta w_{ij}^{(n)} = -\eta \cdot \delta_j^{(n)} \cdot y_i^{(n-1)} \qquad (2.9)$$

Іноді для надання процесу корекції ваг деякої інерційності, що згладжує різкі скачки при переміщенні по поверхні цільової функції, (9) доповнюється значенням зміни ваги на попередній ітерації

$$\Delta w_{ij}^{(n)}(t) = -\eta \cdot (\mu \cdot \Delta w_{ij}^{(n)}(t-1) + (1-\mu) \cdot \delta_j^{(n)} \cdot y_i^{(n-1)}) \qquad (2.10)$$

де (μ - коефіцієнт інерційності, t - номер поточної ітерації.

Тепер торкнемося питання місткості НМ, тобто числа образів, що пред'являються на її входи, які вона здатна навчитися розпізнавати. Для мереж з числом шарів більше двох, він залишається відкритим. Як показано в [4], для НМ з двома шарами, тобто вихідним і одним прихованим шаром, детерміністська місткість мережі Cd оцінюється так:

$$Nw/Ny<Cd<Nw/Ny(\log(Nw/Ny),$$

де Nw - число налаштованих ваг, Ny - число нейронів у вихідному шарі.

Слід зазначити, що цей вираз отриманий з урахуванням деяких обмежень. По-перше, число входів Nx і нейронів в прихованому шарі Nh повинне задовольняти нерівності Nx+Nh>Ny. По-друге, Nw/Ny>1000. Проте наведена вище оцінка виконувалася для мереж з активаційними функціями нейронів у вигляді порогу, а ємність мереж з гладкими активаційними функціями зазвичай більше.

Крім того, прикметник "детерміністський", що фігурує в назві місткості означає, що отримана оцінка місткості підходить абсолютно для усіх можливих вхідних образів, які можуть бути представлені Nx входами. Насправді розподіл вхідних образів, як правило, має деяку регулярність, що дозволяє НМ проводити узагальнення і, таким чином, збільшувати реальну місткість. Оскільки розподіл образів, в загальному випадку, заздалегідь не відомий, ми можемо говорити про таку місткість тільки імовірно, але зазвичай вона разу в два перевищує місткість детерміністську.

Також логічно порушити питання про необхідну потужність вихідного шару мережі, що виконує остаточну класифікацію образів. Річ у тому, що для розподілу множини вхідних образів, наприклад, по двох класах досить усього одного виходу. При цьому кожен логічний рівень - "1" і "0" - означатиме окремий клас. На двох виходах можна закодувати вже 4 класи, на m виходах - 2^m. Проте результати роботи мережі, організованої таким чином, - не дуже надійні.

Для підвищення достовірності класифікації бажано ввести надмірність шляхом виділення кожному класу одного нейрона у вихідному шарі або, що ще краще, декількох, кожен з яких навчається визначати приналежність образу до класу зі своєю мірою достовірності, наприклад: високою, середньою і низькою. Такі НМ дозволяють проводити класифікацію вхідних образів, об'єднаних в нечіткі (розмиті або пересічні) множини. Ця властивість наближає подібні НС до умов реального життя.

Дана НМ має декілька недоліків. По-перше, в процесі навчання може виникнути ситуація, коли великі позитивні або негативні значення вагових коефіцієнтів змістять робочу точку на сигмоїдах багатьох нейронів в область насичення. Малі величини похідної від логістичної функції приведуть у відповідність з і (8) до зупинки навчання, що паралізує НМ.

По-друге, застосування методу градієнтного спуску не гарантує, що буде знайдений глобальний, а не локальний мінімум цільової функції. Ця проблема пов'язана ще з однією, а саме - з вибором величини швидкості навчання. Доказ збіжності навчання в процесі зворотного поширення заснований на похідних, тобто прирости ваг і, отже, швидкість навчання мають бути нескінченно малими, проте в цьому випадку навчання відбуватиметься неприйнятний повільно.

З іншого боку, занадто великі корекції ваг можуть привести до постійної нестійкості процесу навчання. Тому в якості (зазвичай вибирається число менше 1, але не дуже маленьке, наприклад, 0.1, і воно, взагалі кажучи, може поступово зменшуватися в процесі навчання. Крім того, для виключення випадкових попадань в локальні мінімуми іноді, після того, як значення вагових коефіцієнтів стабілізуются, короткочасно сильно збільшують, щоб почати градієнтний спуск з нової точки. Якщо повторення цієї процедури кілька разів приведе алгоритм в один і той же стан НМ, можна більш менш упевнено сказати, що знайдений глобальний максимум, а не якийсь інший.

2.2 Алгоритм зворотного поширення помилки

Алгоритм навчання багатошарової НМ за допомогою процедури зворотного поширення помилки будується так:

0. Ініціалізувати початкові значення вагових параметрів

$$W_{ij} = 0.01*rand(-1,+1).$$

1. Подати на входи мережі один прикладів навчальної вибірки і в режимі звичайного функціонування НМ, коли сигнали поширюються від входів до виходів, розрахувати значення останніх. Нагадаємо, що

$$s_j^{(n)} = \sum_{i=0}^{m_{n-1}} y_i^{(n-1)} \cdot w_{ij}^{(n)}$$

де m_{n-1} - число нейронів в шарі n-1 з урахуванням нейрона з постійним вхідним станом +1, що задає зміщення;

$$y_j(n) = F(s_j^{(n)}),$$

де F() – сигмоїд

$$y_q(0)=x_q$$

де x_q - q -а компонента вектору вхідного образу.

2. Розрахувати $\delta_l^{(N)}$ для вихідного шару по формулі

$$\delta_l^{(N)} = (y_l^{(N)} - d_l) \cdot \frac{dy_l}{ds_l}$$

3. Розрахувати по формулах відповідні $\delta_j^{(n)}$ для усіх інших шарів, n=N - 1,..1.

$$\delta_j^{(n)} = \left[\sum_k \delta_k^{(n+1)} \cdot w_{jk}^{(n+1)} \right] \cdot \frac{dy_j}{ds_j}, \qquad \frac{dy_j}{ds_j} = 1 - s_j^2$$

4. Розрахувати по формулі

$$\Delta w_{ij}^{(n)} = -\eta \cdot \delta_j^{(n)} \cdot y_i^{(n-1)}$$

або .

$$\Delta w_{ij}^{(n)}(t) = -\eta \cdot (\mu \cdot \Delta w_{ij}^{(n)}(t-1) + (1-\mu) \cdot \delta_j^{(n)} \cdot y_i^{(n-1)})$$

зміни ваги $\Delta w_{ij}^{(n)}$ шару n.

5. Скоригувати усі ваги в НМ

$$w_{ij}^{(n)}(t) = w_{ij}^{(n)}(t-1) + \Delta w_{ij}^{(n)}(t)$$

6. Перейти на крок 1 до вичерпання всіх прикладів навчальної вибірки.

7. Якщо функція помилки мережі E > ε для всіх M прикладів навчальної вибірки , перейти на крок 1. Інакше - вихід.

$$E = \frac{1}{2} \sum_{p=1}^{M} \sum_{i=1}^{m} \Delta_{ip}^2 ,$$

де $\Delta_{ip} = (y_{lp}^{(N)} - d_{lp})$

Мережі на кроці 1 поперемінно у випадковому порядку пред'являються усі тренувальні приклади, щоб мережа, образно кажучи, не забувала одні у міру запам'ятовування інших.

2.3Налаштування НМ для вирішення задач

Незважаючи на численні успішні застосування зворотного поширення, воно не є панацеєю. Найбільше прикростей приносить невизначено довгий процес навчання. У складних завданнях для навчання мережі можуть потрібно дні або навіть тижні, вона може і взагалі не навчитися. Тривалий час навчання може бути результатом неоптимального вибору довжини кроку. Невдачі в навчанні зазвичай виникають з двох причин: паралічу мережі і попадання в локальний мінімум.

Параліч мережі. В процесі навчання мережі значення вагів можуть в результаті корекції стати дуже великими величинами. Це може привести до того,

що усе або більшість нейронів функціонуватимуть при дуже великих значеннях OUT, в області, де похідна стискуючої функції дуже мала. Оскільки посилана назад в процесі навчання помилка пропорційна цій похідній, то процес навчання може практично завмерти. У теоретичному відношенні ця проблема погано вивчена. Зазвичай цього уникають зменшенням розміру кроку η, але це збільшує час навчання. Різні евристики використовувалися для оберігання від паралічу або для відновлення після нього, наприклад тимчасове зменшення параметру сигмоїду ***a***, оскільки причиною паралічу НМ часто є нульове значення похідної від активаційної функції F'(S) для великих значень S і це зупиняє процес навчання. Після відновлення навчання значення параметра ***a*** повертається.

Локальні мінімуми. Зворотне поширення використовує різновид градієнтного спуску, тобто здійснює спуск вниз по поверхні помилки, безперервно налаштовуючи ваги у напрямі до мінімуму. Поверхня помилки складної мережі сильно порізана і складається з пагорбів, долин, складок і ярів в просторі високої розмірності. Мережа може потрапити в локальний мінімум (неглибоку долину), коли поруч є набагато глибший мінімум. У точці локального мінімуму усі напрями ведуть вгору, і мережа нездатна з нього вибратися. Статистичні методи навчання можуть допомогти уникнути цієї пастки, але вони повільні. У [1] запропонований метод, що об'єднує статистичні методи машини Коші з градієнтним спуском зворотного поширення і призводить до системи, яка знаходить глобальний мінімум, зберігаючи високу швидкість зворотного поширення. Застосовується також метод «струшування», тобто зміна деяких вагових параметрів для виходу з локального мінімуму і продовження процесу навчання.

Розмір кроку. Уважний розбір збіжності показує, що корекції ваг передбачаються нескінченно малими. Ясно, що це нездійсненно на практиці, оскільки веде до нескінченного часу навчання. Розмір кроку **η** повинен братися кінцевим, і в цьому питанні доводиться спиратися тільки на досвід. Якщо розмір кроку **η** дуже малий, то збіжність занадто повільна, якщо ж дуже великий, то може виникнути параліч або постійна нестійкість. У [11] описаний адаптив-

ний алгоритм вибору кроку, що автоматично коригує розмір кроку в процесі навчання.

Тимчасова нестійкість. Якщо мережа вчиться розпізнавати букви, то немає сенсу учити "Б", якщо при цьому забувається "А". Процес навчання має бути таким, щоб мережа навчалася на усій повчальній множині без пропусків того, що вже вивчене. Необхідні зміни ваг повинні обчислюватися на усій множині, а це вимагає додаткової пам'яті; після ряду таких повчальних циклів ваги зійдуться до мінімальної помилки. Цей метод може виявитися даремним, якщо мережа знаходиться в зовнішньому середовищі, що постійно міняється, так що другий раз один і той же вектор може вже не повторитися. В цьому випадку процес навчання може ніколи не зійтися, безцільно блукаючи або сильно осцилюючи. У цьому сенсі зворотне поширення не схоже на біологічні системи.

Попередня обробка вхідних даних. Процес навчання можна суттєво покращити попередньою обробкою навчальної вибірки. Приклади для кожного класу повинні бути типовими проте різними. При майже однакових прикладах спостерігається синдром «перенавчання», коли нейронна мережа у процесі функціонування добре розпізнає образи близькі до прикладів і не розпізнаються віддалені образи. Разом з тим, дуже «зашумлені» приклади у навчальній вибірці не дозволяють навчити НС взагалі. Суттєво може прискорити навчання попередня нормалізація навчальної вибірки, наприклад методом природної нормалізації.

Згасання помилки у алгоритмі BackProp. Спостерігається згасання помилки при зворотному розповсюдженні, що зупиняє процес навчання. Виходом можуть бути застосування інших методів навчання, зокрема, чисельних методів оптимізації чи використання глибинного навчання.

2.4Навчання НМ без вчителя

Навчання людського мозку, на перший погляд, відбувається без вчителя: на зорові, слухові, тактильні та інші рецептори поступає інформація ззовні, і

усередині нервової системи відбувається якась самоорганізація. Проте, не можна заперечувати і того, що в житті людини не мало вчителів - і в буквальному, і у переносному розумінні, - які координують зовнішні дії. Разом в тим, чим би не закінчилася суперечка прихильників цих двох концепцій навчання, вони обидві мають право на існування.

Головна межа, що робить навчання без вчителя привабливим, - це його "самостійність". Процес навчання, як і у разі навчання з вчителем, полягає в підстроюванні важелів синапсів. Деякі алгоритми, правда, змінюють і структуру мережі, тобто кількість нейронів і їх взаємозв'язку, але такі перетворення правильніше назвати ширшим терміном - самоорганізацією, і в рамках даної статті вони розглядатися не будуть. Очевидно, що підстроювання синапсів може проводитися тільки на підставі інформації, доступної в нейронні, тобто його стану і вже наявних вагових коефіцієнтів. Виходячи з цього міркування і, що важливіше, по аналогії з відомими принципами самоорганізації нервових клітин, побудовані алгоритми навчання Хебба.

Сигнальний метод навчання Хебба полягає в зміні ваг за наступним правилом:

$$w_{ij}(t)=w_{ij}(t-1)+\eta\, y_i^{(n-1)}\, y_j^{(n)} \qquad (2.11)$$

де $y_i^{(n-1)}$ - вихідне значення нейронна і шару; $y_j^{(n)}$- вихідне значення нейронна j шару n; $w_{ij}(t)$- ваговий коефіцієнт синапсу, що сполучає ці нейрони, на ітераціях t і t-1 відповідно; а η- коефіцієнт швидкості навчання. Тут і далі, для спільності, під n мається на увазі довільний шар мережі. При навчанні по даному методу посилюються зв'язки між збудженими нейронами.

Існує також і диференціальний метод навчання Хебба

$$w_{ij}(t)=w_{ij}(t-1)+\eta\,[y_i^{(n-1)}(t) - y_j^{(n-1)}(t-1)][y_i^{(n)}(t) - y_j^{(n)}(t-1)] \qquad (2.12)$$

Повний алгоритм навчання із застосуванням наведених вище формул виглядатиме так:

1. На стадії ініціалізації всім ваговим коефіцієнтам привласнюються невеликі випадкові значення.

2. На вході мережі подається вхідний образ, і сигнали збудження розповсюджуються по всіх шарах згідно принципам класичних прямоточних (FeedForward) мереж тобто для кожного нейрона розраховується зважена сума його входів, до якої потім застосовується активаційна (передавальна) функція нейронна, внаслідок чого виходить його вихідне значення $y_i^{(n)}$, i=0...Mi, де Mi - число нейронів в шарі i; n=1...N, а N - число шарів в мережі.

3. На підставі набутих вихідних значень нейронів по формулах проводиться зміна вагових коефіцієнтів. Цикл з кроку 2, поки вихідні значення мережі не стабілізуються із заданою точністю. Застосування цього нового способу визначення завершення навчання, відмінного від зворотного розповсюдження, що використалося для мережі, обумовлене тим, що підстроюванні значення синапсів фактично не обмежені.

На другому кроці циклу поперемінно пред'являються всі образи з вхідного набору. Слід зазначити, що вид відгуків на кожен клас вхідних образів не відомий заздалегідь і буде довільним поєднанням станів нейронів вихідного шаруючи, обумовлене випадковим розподілом важелів на стадії ініціалізації. Разом з тим, мережа здатна узагальнювати схожі образи, відносячи їх до одного класу. Тестування навченої мережі дозволяє визначити топологію класів у вихідному шарі. Для приведення відгуків навченої мережі до зручного уявлення можна доповнити мережу одним шаром, який, наприклад, по алгоритму навчання одношарового персептрона необхідно змусити відображати вихідні реакції мережі в необхідні образи.

Інший алгоритм навчання без учителя - алгоритм Кохонена - передбачає підстроювання синапсів на підставі їх значень від попередньої ітерації.

$$w_{ij}(t)=w_{ij}(t-1)+\eta[\, y_i^{(n-1)} - w_{ij}^{(n)}(t-1)] \qquad (2.13)$$

З наведеної вище формули видно, що навчання зводиться до мінімізації різниці між вхідними сигналами нейрона, що поступають з виходів нейронів попереднього шару yi(n - 1), і ваговими коефіцієнтами його синапсів.

Повний алгоритм навчання має приблизно таку ж структуру, як в методах Хебба, але на кроці 3 з усього шару вибирається нейрон, значення синапсів якого максимально схожі на вхідний образ, і підстроювання ваг по формулі (3) проводиться тільки для нього. Ця, так звана, акредитація може супроводжуватися гальмуванням усіх інших нейронів шару і введенням вибраного нейрона в насичення. Вибір такого нейрона може здійснюватися, наприклад, розрахунком скалярного твору вектору вагових коефіцієнтів з вектором вхідних значень. Максимальний твір дає нейрон, що виграв.

Інший варіант - розрахунок відстані між цими векторами в p - мірному просторі, де p - розмір векторів.

$$D_j = \sqrt{\sum_{i=0}^{p-1} (y_i^{(n-1)} - w_{ij})^2}\ , \qquad (2.14)$$

де j - індекс нейрона в шарі n, i - індекс підсумовування по нейронах шару (n - 1), w_{ij} - вага синапсу, що сполучає нейрони; виходи нейронів шару (n - 1) є вхідними значеннями для шару n. Корінь у формулі (2.14) брати не обов'язково, оскільки важлива лише відносна оцінка різних Dj.

В даному випадку, "перемагає" нейрон з найменшою відстанню. Іноді занадто часто нейрони, що одержують акредитацію примусово виключаються з розгляду, щоб "зрівняти права" усіх нейронів шару. Простий варіант такого алгоритму полягає в гальмуванні нейрона, що тільки що виграв.

При використанні навчання по алгоритму Кохонена існує практика нормалізації вхідних образів, а так само - на стадії ініціалізації - і нормалізації початкових значень вагових коефіцієнтів.

$$x_i = \frac{x_i}{\sqrt{\sum_{j=1}^{n-1} x_j^2}}, \qquad (2.15)$$

де x_i - і -а компонента вектору вхідного образу або вектору вагових коефіцієнтів, а n - його розмірність.

Це дозволяє скоротити тривалість процесу навчання. Ініціалізація вагових коефіцієнтів випадковими значеннями може привести до того, що різні класи, яким відповідають щільно розподілені вхідні образи, зіллються або, навпаки, роздрібняться на додаткові підкласи у разі близьких образів одного і того ж класу. Для уникнення такої ситуації використовується метод опуклої комбінації[3]. Суть його зводиться до того, що вхідні нормалізовані образи піддаються перетворенню :

$$x_{\text{i}} = \alpha x_{\text{i}} + \frac{1 - \alpha}{\sqrt{\text{n}}}, \qquad (2.16)$$

де xi - і -а компонента вхідного образу, n - загальне число його компонент, □ (t) - коефіцієнт, що змінюється в процесі навчання від нуля до одиниці, внаслідок чого спочатку на входи мережі подаються практично однакові образи, а з часом вони все більше сходяться до початкових. Вагові коефіцієнти встановлюються на кроці ініціалізації рівними величині

$$w_{\text{i}} = \frac{1}{\sqrt{\text{n}}}, \qquad (2.17)$$

де n - розмірність вектору ваг для нейронів ініціалізованого шару.

На основі розглянутого вище методу будуються нейронні мережі особливого типу - так звані структури, що самоорганізуються, - self - organizing feature maps (цей сталий переклад з англійського, на мій погляд, не дуже вдалий, оскільки, йдеться не про зміну структури мережі, а тільки про підстроювання синапсів). Для них після вибору з шару n нейрона j з мінімальною відстанню Dj (2.14) навчається по формулі (2.13) не лише цей нейрон, але і його сусіди, розташовані в околиці R. Величина R на перших ітераціях дуже велика, так що навчаються усі

нейрони, але з часом вона зменшується до нуля. Таким чином, чим ближче кінець навчання, тим точніше визначається група нейронів, що відповідають кожному класу образів.

2.5 Контрастування НМ

Виробництво явних знань з накопичених даних - проблема, яка набагато старше чим комп'ютери. Навчені нейронні мережі можуть виробляти з даних приховані знання: створюється навичка пророцтва, класифікації, розпізнавання образів і тому подібне, але її логічна структура зазвичай залишається прихованою від користувача. Проблема прояву (контрастування) цієї прихованої логічної структури вирішується в роботі шляхом приведення нейронних мереж до спеціального "логічно прозорого" розрідженого виду.

Досліджуються два питання, що встають перед кожним дослідником, що вирішив використовувати нейронні мережі, : "Скільки нейронів потрібно для вирішення завдання"? і "Яка має бути структура нейронної мережі"? Об'єднуючи ці два питання, ми отримуємо третій: "Як зробити роботу нейронної мережі зрозумілою для користувача (логічно прозорою) і які вигоди може принести таке розуміння"? Описані способи отримання логічно прозорих нейронних мереж. Наведений приклад з області соціально-політичних пророцтв. Для визначеності розглядаються тільки нейронні мережі, навчання яких є мінімізація оцінок (помилок) з використанням градієнта. Градієнт оцінки обчислюється методом двоїстості (його окремий випадок - метод зворотного поширення помилки).

Скільки нейронів треба використовувати? При відповіді на це питання існує дві протилежні точки зору. Одна з них стверджує, що чим більше нейронів використовувати, тим більше надійна мережа вийде. Прибічники цієї позиції посилаються на приклад людського мозку. Дійсно, ніж більше нейронів, тим більше число зв'язків між ними, і тим більше складні завдання здатна вирішити нейронна мережа. Крім того, якщо використовувати свідомо більше число нейронів, чим необхідно для вирішення завдання, то нейронна мережа точно навчиться. Якщо ж починати з невеликого числа нейронів, то мережа може вияви-

тися нездатною навчитися рішенню задачі, і увесь процес доведеться повторювати спочатку з великим числом нейронів.

Ця точка зору (чим більше - тим краще) популярна серед розробників нейромережного програмного забезпечення. Так, багато з них як одне з основних достоїнств своїх програм називають можливість використання будь-якого числа нейронів.

Другий підхід визначає потрібне число нейронів як мінімально необхідне. Основним недоліком є те, що це, мінімально необхідне число, заздалегідь невідомо, а процедура його визначення шляхом поступового нарощування числа нейронів дуже трудомістка. Спираючись на досвід роботи в області медичної діагностики , космічній навігації і психології можна відмітити, що в усіх цих завданнях жодного разу не було потрібно більше декількох десятків нейронів.

Підводячи підсумок аналізу двох крайніх позицій, можна сказати наступне: мережа з мінімальним числом нейронів повинна краще ("правильніше", більше гладко) апроксимувати функцію, але з'ясування цього мінімального числа нейронів вимагає великих інтелектуальних витрат і експериментів по навчанню мереж. Якщо число нейронів надмірне, то можна отримати результат з першої спроби, але існує ризик побудувати "погану" апроксимацію.

Істина, як завжди буває в таких випадках, лежить посередині: треба вибирати число нейронів великим чим необхідно, але не набагато. Це можна здійснити шляхом подвоєння числа нейронів в мережі після кожної невдалої спроби навчання. Проте існує надійніший спосіб оцінки мінімального числа нейронів - використання процедури контрастування. Крім того, процедура контрастування дозволяє відповісти і на друге питання: яка має бути структура мережі.

Контрастування на основі оцінки. Розглянемо мережу, що правильно вирішує усі приклади повчальної великої кількості. Позначимо через w_p,, $p=1,...,n$ ваги усіх зв'язків. При зворотному функціонуванні мережі за принци-

пом двоїстості або методом зворотного поширення помилки мережа обчислює вектор градієнта функції оцінки E по вагах зв'язків w_k : $\nabla E = \left\{ \frac{\partial E}{\partial w_k} \right\}_{k=1}^{K}$.

Нехай w^0_p- поточний набір ваг зв'язків, а E_0 - помилка поточного прикладу. Тоді в лінійному наближенні можна записати функцію оцінки E в точці w_k як :

$$\mathrm{E}(\mathrm{w_k}) = E^0 + \sum_{k=1}^{K} \frac{\partial E}{\partial w_k}(w_k - w_k^0)$$

Використовуючи це наближення можна оцінити зміну оцінки при заміні w_k на w_k^* як:

$$\chi(k, p) = \left| \frac{\partial E}{\partial w_k} \right| \cdot \left| w_k^* - w_k^0 \right|,$$

де p - номер прикладу навчальної вибірки, для якого були обчислені оцінка і градієнт. Величину $\chi(k, p)$ називатимемо показником чутливості до заміни w_p на w_p^* для прикладу p . Далі необхідно обчислити показник чутливості, незалежний від номера прикладу. Для цього можна скористатися будь-якою нормою. Зазвичай використовується рівномірна норма (максимум модуля):

$$\chi(k) = \max_p \chi(k, p).$$

Уміючи обчислювати показники чутливості, можна приступати до процедури контрастування.

Приведемо простий варіант цієї процедури :

1. Обчислюємо показники чутливості для кожної ваги w_k .
2. Знаходимо мінімальний серед показників чутливості для різних прикладів навчальної вибірки.
3. Замінюємо відповідний цьому показнику чутливості вагу на w_k^0 , і виключаємо його з процедури навчання.

4. Пред'явимо мережі усі приклади навчальної вибірки. Якщо мережа не допустила жодної помилки, то переходимо до другого кроку процедури.
5. Намагаємося навчити відконтрастовану мережу. Якщо мережа навчилася безпомилковому рішенню задачі, то переходимо до першого кроку процедури, інакше переходимо до шостого кроку.
6. Відновлюємо мережу в стан до останнього виконання третього кроку. Якщо в ході виконання кроків з другого по п'ятий була відконтрастована хоч би одна вага, (число навчених ваг змінилося), то переходимо до першого кроку. Якщо жодна вага не була відконтрастована, то отримана мінімальна мережа.

Можливе використання різних узагальнень цієї процедури. Наприклад, контрастувати за один крок процедури не одну вагу, а задане користувачем число ваг. Найбільш радикальна процедура полягає в контрастуванні половини ваг зв'язків. Якщо половину ваг відконтрастувати не вдається, то намагаємося відконтрастувати чверть і так далі. Відмітимо, що при описаному методі обчислення показників чутливості, передбачається можливим обчислення функції оцінки і проведення процедури навчання мережі, а також передбачається відомою повчальна множина. Можливий і інший шлях.

Контрастування без погіршення. Нехай нам дана тільки навчена нейронна мережа і повчальна множина. Допустимо, що вид функції оцінки і процедура навчання нейронної мережі невідомі. В цьому випадку так само можливе контрастування мережі. Припустимо, що ця мережа ідеально вирішує задачу. Тоді нам необхідно так відконтрастувати ваги зв'язків, щоб вихідні сигнали мережі при рішенні усіх завдань змінилися не більше ніж на задану величину. В цьому випадку контрастування ваг виконується по-нейронах. На вході кожного нейрона стоїть адаптивний суматор, який підсумовує вхідні сигнали нейрона, помножені на відповідні ваги зв'язків. Для нейрона найменш чутливою буде та вага, яка при рішенні прикладу дасть найменший вклад в суму. Позначивши через

x_k^p вхідні сигнали даного нейрона при рішенні *p*-го прикладу отримуємо формулу для показника чутливості ваг:

$$\chi(k, p) = \left|x_k^p\right| \cdot \left|w_k^* - w_k^0\right|.$$

Аналогічно раніше розглянутому отримуємо: $\chi(k) = \max_p \left|x_k^p\right| \cdot \left|w_k^* - w_k^0\right|$.

В самій процедурі контрастування є тільки одна відмінність - замість перевірки на наявність помилок при пред'явленні усіх прикладів перевіряється, що нові вихідні сигнали мережі відрізняються від первинних не більше ніж на задану величину.

Логічно прозорі нейронні мережі. Одним з основних недоліків нейронних мереж, з точки зору багатьох користувачів, є те, що нейронна мережа вирішує задачу, але не може розповісти як. Іншими словами з навченої нейронної мережі не можна витягнути алгоритм рішення задачі. Проте спеціальним чином побудована процедура контрастування дозволяє вирішити і це завдання.

Задамося класом мереж, які вважатимемо логічно прозорими (тобто такими, які вирішують задачу зрозумілим для нас способом, для якого легко сформулювати словесне описи у вигляді явного алгоритму). Наприклад зажадаємо, щоб усі нейрони мали не більше трьох вхідних сигналів.

Задамося нейронною мережею у якої усі вхідні сигнали подаються на усі нейрони вхідного шару, а усі нейрони кожного наступного шару приймають вихідні сигнали усіх нейронів попереднього шару. Навчимо мережу безпомилковому рішенню задачі.

Після цього вироблятимемо контрастування у декілька етапів. На першому етапі контрастуватимемо тільки ваги зв'язків нейронів вхідного шару. Якщо після контрастування у деяких нейронів залишилися більше трьох вхідних сигналів, то збільшимо число вхідних нейронів. Потім аналогічну процедуру виробимо по черзі для усіх інших шарів. Після завершення описаної процедури буде отримана логічно прозора мережа. Можна виробити додаткове контрастування мережі, щоб отримати мінімальну мережу.

На рис. 11 приведено вісім мінімальних мереж. Якщо під логічно прозорими мережами розуміти мережі, у яких кожен нейрон має не більше трьох входів, то усі мережі окрім п'ятої і сьомої є логічно прозорими. П'ята і сьома мережі демонструють той факт, що мінімальність мережі не спричиняє за собою логічної прозорості.

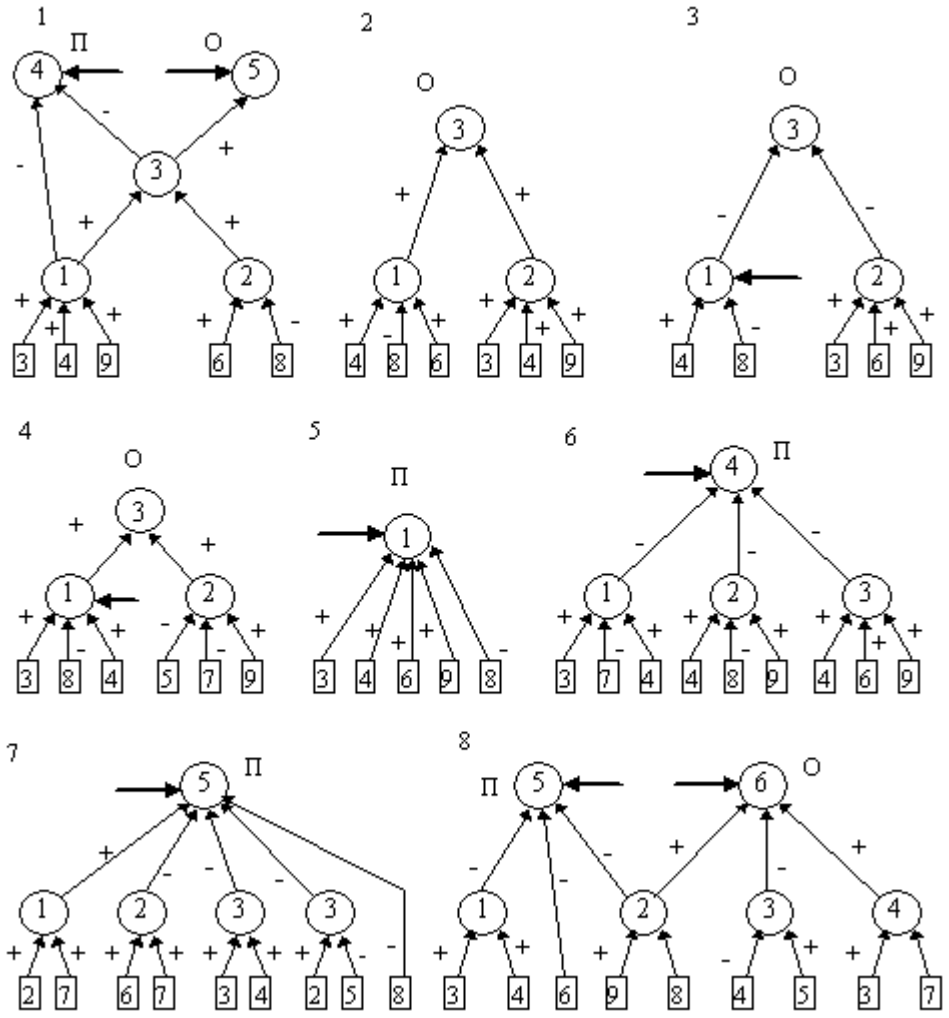

Рис. 11 Приклади логічно прозорих НМ

Технологія отримання явних знань з даних за допомогою навчених нейронних мереж виглядає досить просто і начебто не викликає проблем - необхідно її просто реалізовувати і користуватися.

Перший етап: навчаємо нейронну мережу вирішувати базову задачу. Зазвичай базовим є завдання розпізнавання, пророцтва (як в попередньому розділі) і тому подібне. В більшості випадків її можна трактувати як завдання про

заповнення пропусків в даних. Такими пропусками є і ім'я образу при розпізнаванні, і номер класу, і результат прогнозу, та ін.

Другий етап: за допомогою аналізу показників значущості, контрастування і доучування (усе це застосовується, найчастіше, неодноразово) приводимо нейронну мережу до логічно прозорого виду - так, щоб отриману мережу можна було "прочитати".

Отриманий результат неоднозначний - якщо стартувати з іншої початкової карти, то можна отримати іншу логічно прозору структуру. Кожній базі даних відповідає декілька варіантів явних знань. Можна вважати це недоліком технології, але ми вважаємо, що, навпаки, технологія, що дає єдиний варіант явних знань, недостовірна, а неєдиність результату є фундаментальною властивістю виробництва явних знань з даних.

2.6 Стохастичні методи навчання

Стохастичні методи корисні як для навчання штучних нейронних мереж, так і для отримання виходу від вже навченої мережі. Стохастичні методи навчання приносять велику користь, дозволяючи виключати локальні мінімуми в процесі навчання. Але з ними також пов'язаний ряд проблем.

Штучна нейронна мережа навчається за допомогою деякого процесу, ваги, що модифікує її. Якщо навчання успішне, то пред'явлення мережі безлічі вхідних сигналів призводить до появи бажаної безлічі вихідних сигналів. Є два класи повчальних методів : детерміністський і стохастичний.

Детерміністський метод навчання крок за кроком здійснює процедуру корекції вагів мережі, засновану на використанні їх поточних значень, а також величин входів, фактичних виходів і бажаних виходів. Навчання персептрона є прикладом подібного детерміністського підходу (див. гл. 1).

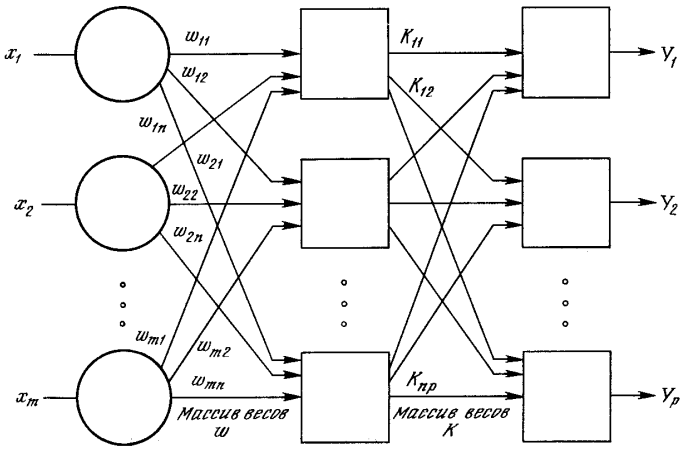

Рис. 12 Двошарова мережа без зворотних зв'язків

Стохастичні методи навчання виконують псевдовипадкові зміни величин ваг, зберігаючи ті зміни, які ведуть до поліпшень. Щоб побачити, як це може бути зроблено, розглянемо рис. , на якому зображена типова мережа, в якій нейрони сполучені за допомогою ваг. Вихід нейрона є тут зваженою сумою його входів, яка, перетворена за допомогою нелінійної функції.

Для навчання мережі може бути використана наступна процедура:

1. Вибрати вагу випадковим чином і підкоригувати її на невелике випадкове Пред'явити безліч входів і вичислити виходи, що виходять.

2. Порівняти ці виходи з бажаними виходами і вичислити величину різниці між ними. Загальноприйнятий метод полягає в знаходженні різниці між фактичним і бажаним виходами для кожного елементу навченої пари, зведення різниць в квадрат і знаходження суми цих квадратів. Метою навчання є мінімізація цієї різниці, часто званою цільовою функцією.

3. Вибрати вагу випадковим чином і підкоригувати її на невелике випадкове значення. Якщо корекція допомагає (зменшує цільову функцію), то зберегти її, інакше повернутися до первинного значення ваги.

4. Повторювати кроки з 1 до 3 до тих пір, поки мережа не буде навчена достатньою мірою.

Цей процес прагне мінімізувати цільову функцію, але може потрапити, як в пастку, в невдале рішення. На рис. 13 показано, як це може мати місце в системі з єдиною вагою. Допустимо, що спочатку вага узята рівною значенню в

точці А. Якщо випадкові кроки по вазі малі, то будь-які відхилення від точки А збільшують цільову функцію і будуть знехтувані. Краще значення ваги, що приймається в точці В, ніколи не буде знайдено, і система буде спіймана в пастку локальним мінімумом, замість глобального мінімуму в точці В. Якщо ж випадкові корекції ваги дуже великі, то як точка А, так і точка В часто відвідуватимуться, але те ж саме матиме місце і для кожної іншої точки. Вага мінятиметься так різко, що він ніколи не встановиться в бажаному мінімумі.

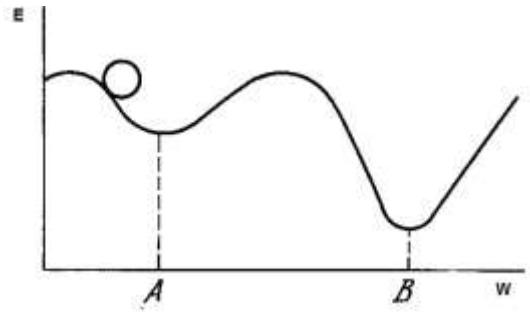

Рис. 13 Проблема локальних мінімумів

Корисна стратегія для уникнення подібних проблем полягає у великих початкових кроках і поступовому зменшенні розміру середнього випадкового кроку. Це дозволяє мережі вириватися з локальних мінімумів і в той же час гарантує остаточну стабілізацію мережі.

Пастки локальних мінімумів докучають усім алгоритмам навчання, заснованим на пошуку мінімуму, включаючи персептрон і мережі зворотного поширення, і представляють серйозну і широко поширену трудність, якої часто не помічають. Стохастичні методи дозволяють розв'язати цю проблему. Стратегія корекції ваг, що змушує ваги набувати значення глобального оптимуму в точці В, можлива.

Як пояснююча аналогія припустимо, що на рис.13 зображена кулька на поверхні в коробці. Якщо коробку сильно потрясти в горизонтальному напрямі, то кулька швидко перекочуватиметься від одного краю до іншого. Ніде не затримуючись, в кожен момент кулька з рівною імовірністю знаходитиметься в будь-якій точці поверхні.

Якщо поступово зменшувати силу струшування, то буде досягнута умова, при якій кулька на короткий час "застряватиме" в точці В. При ще слабкішому струшуванні кулька на короткий час зупинятиметься як в точці А, так і в точці В. При безперервному зменшенні сили струшування буде досягнута критична точка, коли сила струшування достатня для переміщення кульки з точки А в точку В, але недостатня для того, щоб кулька могла видертися з У в А. Таким чином, остаточно кулька зупиниться в точці глобального мінімуму, коли амплітуда струшування зменшиться до нуля.

Штучні нейронні мережі можуть навчатися по суті тим же самим чином за допомогою випадкової корекції ваг. Спочатку робляться великі випадкові корекції із збереженням тільки тих змін ваг, які зменшують цільову функцію. Потім середній розмір кроку поступово зменшується, і глобальний мінімум врешті-решт досягається.

Це сильно нагадує відпал металу, тому для її опису часто використовують термін "імітація відпалу". У металі, нагрітому до температури, що перевищує його точку плавлення, атоми знаходяться в сильному безладному русі. Як і в усіх фізичних системах, атоми прагнуть до стану мінімуму енергії (єдиному кристалу в даному випадку), але при високих температурах енергія атомних рухів перешкоджає цьому. В процесі поступового охолодження металу виникають усе більш низько енергетичні стани, поки врешті-решт не буде досягнуто найнижче з можливих станів, глобальний мінімум. В процесі відпалу розподіл енергетичних рівнів описується наступним співвідношенням:

$$P(e) = \exp(-e/kT)$$

де *P(e)* - вірогідність того, що система знаходиться в змозі з енергією e; k - постійна Больцмана; Т - температура за шкалою Кельвіна.

При високих температурах Р(е) наближається до одиниці для усіх енергетичних станів. Таким чином, високо енергетичний стан майже так же ймовірний, як і низько енергетичний. У міру зменшення температури вірогідність ви-

соко енергетичних станів зменшується в порівнянні з низько енергетичними. При наближенні температури до нуля стає дуже маловірогідним, щоб система знаходилася у високо енергетичному стані.

Навчання Больцмана. Цей стохастичний метод безпосередньо застосований до навчання штучних нейронних мереж :

1. Визначити змінну T, що представляє штучну температуру. Надати T велике початкове значення.
2. Пред'явити мережі безліч входів і вичислити виходи і цільову функцію.
3. Дати випадкову зміну вазі і перерахувати вихід мережі і зміну цільової функції відповідно до зробленої зміни ваги.
4. Якщо цільова функція зменшилася (покращала), то зберегти зміну ваги.

Якщо зміна ваги призводить до збільшення цільової функції, то вірогідність збереження цієї зміни обчислюється за допомогою розподілу Больцмана :

$$P(c) = \exp(-c/kT)$$

де P(c) - вірогідність зміни з в цільовій функції; k - константа, аналогічна константі Больцмана, вибирана залежно від завдання; T - штучна температура.

Вибирається випадкове число r з рівномірного розподілу від нуля до одиниці. Якщо P(c) більше, ніж r, то зміна зберігається, інакше величина ваги повертається до попереднього значення.

Це дозволяє системі робити випадковий крок в напрямі, що псує цільову функцію, дозволяючи їй тим самим вириватися з локальних мінімумів, де будь-який малий крок збільшує цільову функцію.

Для завершення больцманівского навчання повторюють кроки 3 і 4 для кожного з ваг мережі, поступово зменшуючи температуру T, поки не буде досягнуто допустимо низьке значення цільової функції. У цей момент пред'являється інший вхідний вектор і процес навчання повторюється. Мережа навчається на усіх векторах повчальної великої кількості, з можливим повторенням, поки цільова функція не стане допустимою для усіх них.

Величина випадкової зміни ваги на кроці 3 може визначатися різними способами. Наприклад, подібно до теплової системи вагова зміна w може вибиратися відповідно до розподілу гауса :

$$P(w) = \exp(-w^2/T^2)$$

де P(w) - вірогідність зміни ваги на величину w, T - штучна температура.

Такий вибір зміни ваги призводить до системи, аналогічною .

Оскільки потрібна величина зміни ваги Δw, а не вірогідність зміни ваги, що має величину w, то метод Монте-Карло може бути використаний таким чином:

1. Знайти кумулятивну вірогідність, відповідну P(w). Це є інтеграл від P(w) в межах від 0 до w. Оскільки в даному випадку P(w) не може проінтегрувати аналітично, вона повинна інтегруватися чисельно, а результат потрібне затабулировать.

2. Вибрати випадкове число з рівномірного розподілу на інтервалі (0,1). Використовуючи цю величину як значення P(w}, знайти в таблиці відповідне значення для величини зміни ваги.

Властивості машини Больцмана широко вивчалися. У роботі [12] показано, що швидкість зменшення температури має бути назад пропорційна логарифму часу, щоб була досягнута збіжність до глобального мінімуму. Швидкість охолодження в такій системі виражається таким чином:

$$T(t) = \frac{T_0}{\log(1+t)}$$

де T(t) - штучна температура як функція часу; T_0 - початкова штучна температура; t - штучний час.

Цей результат, що розчаровує, передбачає дуже повільну швидкість охолодження (і ці обчислення). Цей вивід підтвердився експериментально. Машини Больцмана часто вимагають для навчання дуже великого ресурсу часу.

Навчання Коші. У цьому методі при обчисленні величини кроку розподіл Больцмана замінюється на розподіл Коши. Розподіл Коши має, як показано на рис. 14 , довші "хвости", збільшуючи тим самим вірогідність великих кроків. Насправді розподіл Коши має нескінченну (невизначену) дисперсію. За допомогою такої простої зміни максимальна швидкість зменшення температури стає обернено пропорційній лінійній величині, а не логарифму, як для алгоритму навчання Больцмана. Це різко зменшує час навчання. Цей зв'язок може бути виражений таким чином:

$$T(t) = \frac{T_0}{1+t}$$

Розподіл Коши має вигляд

$$P(x) = \frac{T(t)}{T(t)^2 + x^2}$$

де P(x) є вірогідність кроку величини x.

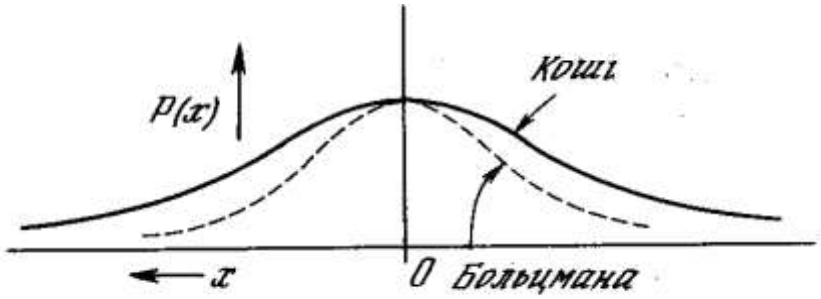

Рис. 14 Розподіл Коші і розподіл Больцмана

У рівнянні P(x) можна про інтегрувати стандартними методами. Вирішуючи відносно x, отримуємо

$$x_c = (T(t)\ tg(P(x)), \qquad ($$

де (- коефіцієнт швидкості навчання; xc - зміна ваги.

Тепер застосування методу Монте Карло стає дуже простим. Для знаходження x в цьому випадку вибирається випадкове число з рівномірного розпо-

ділу на відкритому інтервалі (–π/2, π/2) (необхідно обмежити функцію тангенса). Воно підставляється у формулу (2.6) як P(x), і за допомогою поточної температури обчислюється величина кроку.

2.7 Теми і алгоритми лабораторних робіт

Лабораторна робота № 3

Створити нейрокомп'ютерну систему розпізнавання рукописних символів, використавши алгоритм зворотного поширення помилки для навчання багатошарової НМ.

Алгоритм лабораторної роботи №3

1. Створити графічний інтерфейс для введення прикладів і розпізнавання символів.
2. Створити навчальну вибірку рукописних символів або використати стандартний набір MNIST з Internet(рис.15)

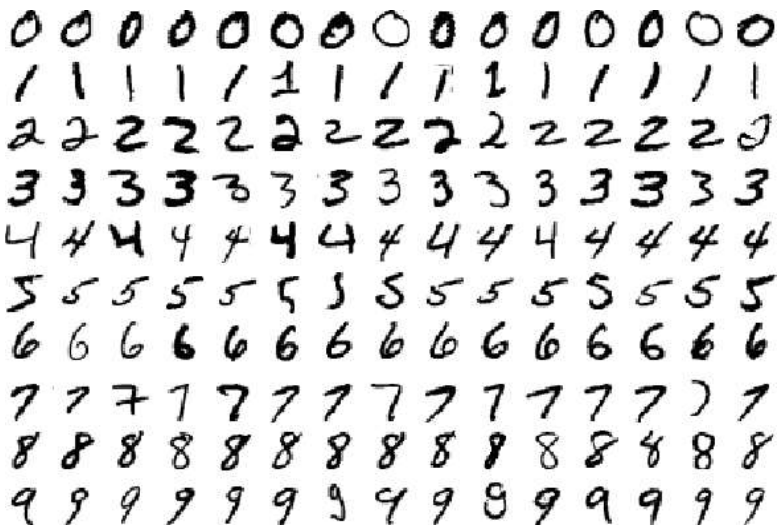

Рис. 15 Набір рукописних символів MNIST

3. Спроектувати нейронну мережу задаючись на вході достатньою кількістю рецепторів для відображення символів через мегапікселі двовимірної сітки, а на виході НС достатньою кількістю нейронів для кодування символів (кількість різних символів <=2^m, де m –кількість нейронів на виході). Кількість нейронів у проміжних шарах визначити експериментально.
4. Навчити НМ згідно з алгоритмом BackProp (розділ 1.7).
5. Протестувати роботу НС на контрольній частині навчальної вибірки і при необхідності донавчити НС.

Лабораторна робота № 4

1. Створити нейромережу прогнозування результатів президентських виборів США у 2016р. для заданої навчальної вибірки.
2. Максимально відконтрастувати двошарову нейронну мережу
3. Процес контрастування зобразити графічно для заданої НМ.

Алгоритм контрастування НМ

1. Обчислюємо показники чутливості.

2. Знаходимо мінімальний серед показників чутливості.

3. Замінюємо відповідну цьому показнику чутливості вагу □на □0, і виключаємо її з процедури навчання.

4. Пред'явимо мережі усі приклади навчальної вибірки. Якщо мережа не допустила жодної помилки, то переходимо до другого кроку процедури.

5. Намагаємося навчити відконтрастовану мережу. Якщо мережа навчилася безпомилковому рішенню задачі, то переходимо до першого кроку процедури, інакше переходимо до шостого кроку.

6. Відновлюємо мережу в стан до останнього виконання третього кроку. Якщо в ході виконання кроків з другого по п'ятий була відконтрастована хоч би одна вага, (число навчених ваг змінилося), то переходимо до першого кроку. Якщо жодна вага не була відконтрастована, то отримана мінімальна мережа.

Питання для тестування:

1. Правляча партія була при владі більше одного терміну?
2. Правляча партія отримала більше 50% голосів на минулих виборах?
3. У рік виборів була активна третя партія?
4. Була серйозна конкуренція при висуненні від правлячої партії?

5. Кандидат від правлячої партії був президентом у рік виборів?

6. Чи був рік виборів часом спаду або депресії?

7. Чи був зростання середнього національного валового продукту на душу населення більше 2.1%?

8. Чи справив правлячий президент істотні зміни в політиці?

9. Під час правління були істотні соціальні хвилювання?

10. Адміністрація правлячої партії винна в серйозній помилці чи скандалі?

11. Кандидат від правлячої партії - національний герой?

12. Кандидат від опозиційної партії - національний герой?

Історія президентських виборів в США (1-так, 0-ні)

Питання	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**перемогла**
1860	1	0	1	1	0	0	1	0	1	0	0	0	опозиційна
1864	0	0	0	0	1	0	0	1	1	0	0	0	правляча
1868	1	1	0	0	0	0	1	1	1	0	1	0	правляча
1872	1	1	0	0	1	0	1	0	0	0	1	0	правляча
1876	1	1	0	1	0	1	0	0	0	1	0	0	опозиційна
1880	1	0	0	1	0	0	1	1	0	0	0	0	правляча
1884	1	0	0	1	0	0	1	0	1	0	1	0	опозиційна
1888	0	0	0	0	1	0	0	0	0	0	0	0	правляча
1892	0	0	1	0	1	0	0	1	1	0	0	1	опозиційна
1896	0	0	0	1	0	1	0	1	1	0	1	0	опозиційна
1904	1	1	0	0	1	0	0	0	0	0	1	0	правляча
1908	1	1	0	0	0	0	0	1	0	0	0	1	правляча
1912	1	1	1	1	1	0	1	0	0	0	0	0	опозиційна
1916	0	0	0	0	1	0	0	1	0	0	0	0	правляча
1920	1	0	0	1	0	0	0	1	1	0	0	0	опозиційна

1924	0	1	1	0	1	0	1	1	0	1	0	0	правляча
1928	1	1	0	0	0	0	1	0	0	0	0	0	правляча
1932	1	1	0	0	1	1	0	0	1	0	0	1	опозиційна
1936	0	1	0	0	1	1	1	1	0	0	1	0	правляча
1940	1	1	0	0	1	1	1	1	0	0	1	0	правляча
1944	1	1	0	0	1	0	1	1	0	0	1	0	правляча
1948	1	1	1	0	1	0	0	1	0	0	0	0	правляча
1952	1	0	0	1	0	0	1	0	0	1	0	1	опозиційна
1956	0	1	0	0	1	0	0	0	0	0	1	0	правляча
1960	1	1	0	0	0	1	0	0	0	0	0	1	опозиційна
1964	0	0	0	0	1	0	1	0	0	0	0	0	правляча
1968	1	1	1	1	0	0	1	1	1	0	0	0	опозиційна
1972	0	0	0	0	1	0	1	1	1	0	0	0	правляча
1976	1	1	0	1	1	0	0	0	0	1	0	0	опозиційна
1980	0	0	1	1	1	1	0	0	0	1	0	1	опозиційна
1992	0	1	0	0	1	0	1	0	0	0	0	1	правляча
1996	1	1	0	0	1	0	0	1	0	0	0	0	правляча
2000	1	1	0	0	0	0	1	0	0	1	0	0	правляча
2004	1	1	1	0	1	0	1	0	0	1	0	0	правляча
2008	1	1	1	0	1	0	1	0	0	1	0	0	опозиційна
2012	1	1	0	0	1	0	1	0	0	0	0	0	правляча
2016	1	1	0	0	1	0	1	0	0	0	0	0	
Питання	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**перемогла**

ГЛАВА 3 НЕЙРОННІ МЕРЕЖІ ЗІ СКЛАДНОЮ АРХІТЕКТУРОЮ

3.1 Нейронні мережі зустрічного поширення (НМЗП)

Можливості мережі зустрічного розповсюдження, перевершують можливості одношарових мереж. Час же навчання в порівнянні із зворотним розповсюдженням може зменшуватися в сто разів. Зустрічне розповсюдження не таке загальне, як зворотне розповсюдження, але воно може давати рішення в тих застосуваннях, де довга навчаюча процедура неможлива. Окрім подолання обмежень інших мереж зустрічне розповсюдження володіє власними цікавими і корисними властивостями.

У зустрічному розповсюдженні об'єднані два добре відомих алгоритми: карта Кохонена, що самоорганізується та зірка Гроссберга. Їх об'єднання веде до властивостей, яких немає у жодного з них окремо. Методи, які подібно до зустрічного розповсюдження, об'єднують різні мережеві парадигми як будівельні блоки, можуть привести до мереж, ближчих до мозку по архітектурі, чим будь-які інші однорідні структури. На рис. 16 показана спрощена версія прямої дії мережі **зустрічного розповсюдження**. На ньому ілюструються функціональні властивості цієї парадигми. Повна двонаправлена мережа заснована за тими же принципами, вона обговорюється в цьому розділі пізніше.

Нейрони шару 0 (показані колами) служать лише точками розгалуження і не виконують обчислень. Кожен нейрон шару 0 сполучений з кожним нейроном шару 1 (званого шаром Кохонена) окремою вагою w_{mn}. Ці ваги в цілому розглядуються як матриця вагів W. Аналогічно, кожен нейрон в шарі Кохонена (шарі 1) сполучений з кожним нейроном в шарі Гроссберга (шарі 2) вагою vnp.

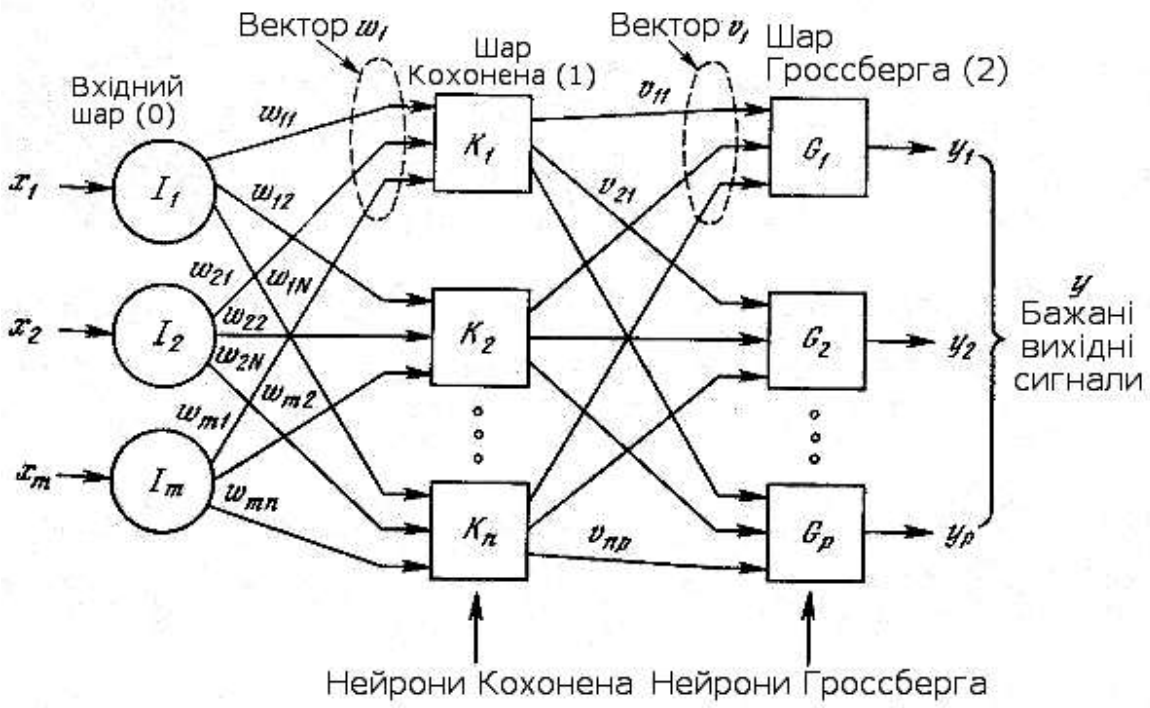

Рис. 16 Мережа зустрічного поширення сигналу

Ці ваги утворюють матрицю ваг V. Все це нагадує інші мережі, що зустрічалися в попередніх розділах, відмінність, проте, полягає в операціях, що виконуються нейронами Кохонена і Гроссберга. Як і багато інших мереж, зустрічне розповсюдження функціонує в двох режимах: у нормальному режимі, при якому приймається вхідний вектор X і видається вихідний вектор Y, і в режимі навчання, при якому подається вхідний вектор і ваги коректуються, щоб дати необхідний вихідний вектор.

Як і багато інших мереж, зустрічне поширення функціонує в двох режимах: в нормальному режимі, при якому приймається вхідний вектор X і видається вихідний вектор Y, і в режимі навчання, при якому подається вхідний вектор і ваги коригуються, щоб дати необхідний вихідний вектор.

Шар Кохонена. У своїй простій формі шар Кохонена функціонує в дусі "переможець забирає усе", т. е. для цього вхідного вектору один і тільки один нейрон Кохонена видає на виході логічну одиницю, усі інші видають нуль. Нейрони Кохонена можна сприймати як набір електричних лампочок, так що для будь-якого вхідного вектору спалахує одна з них.

Асоційоване з кожним нейроном Кохонена безліч вагів сполучає його з кожним входом. Наприклад, на мал. 4.1 нейрон Кохонена К1 має ваги w11, w21, ., wm1, що становлять ваговий вектор W1. Вони соединяются-через вхідний шар з вхідними сигналами x1, x2, ., xm, що становлять вхідний вектор X. Подібно до нейронів більшості мереж вихід NET кожного нейрона Кохонена є просто сумою зважених входів. Це може бути виражено таким чином:

$$NETj = w_{1j}x_1 + w_{2j}x_2 + . + w_{mj}x_m \qquad (3.1)$$

де NETj - це вихід NET нейрона Кохонена j

$$NET_j = \sum_i x_i w_{ij} \qquad (3.2)$$

чи у векторному записі

$$\mathbf{N = XW} \qquad \mathbf{(3.3)}$$

де N - вектор виходів NET шару Кохонена.

Нейрон Кохонена з максимальним значенням NET є "переможцем". Його вихід дорівнює одиниці, у інших він дорівнює нулю.

Шар Гроссберга. Шар Гроссберга функціонує в схожій манері. Його вихід NET є зваженою сумою виходів k1, k2, .., kn шару Кохонена, що утворюють вектор До. Вектор сполучаючих вагів, позначений через V, складається з вагів v11, v21, .., vnp. Тоді вихід NET кожного нейрона Гроссберга є

$$NET_j = \sum_i k_i w_{ij} , \qquad (3.4)$$

де NETj - вихід j -го нейрона Гроссберга, або у векторній формі

$$Y = KV \qquad (3.5)$$

де Y - вихідний вектор шару Гроссберга, До - вихідний вектор шару Кохонена, V - матриця вагів шару Гроссберга.

Якщо шар Кохонена функціонує таким чином, що лише у одного нейрона величина NET дорівнює одиниці, а у інших дорівнює нулю, то лише один елемент вектору До відмінний від нуля, і обчислення дуже прості. Фактично кожен

нейрон шару Гроссберга лише видає величину ваги, яка зв'язує цей нейрон з єдиним ненульовим нейроном Кохонена.

Шар Кохонена класифікує вхідні вектори в групи схожих. Це досягається за допомогою такого підстроювання вагів шару Кохонена, що близькі вхідні вектори активують один і той же нейрон цього шару. Потім завданням шару Гроссберга є отримання необхідних виходів.

Навчання шару Кохонена є самонавчанням, що протікає без учителя. Тому важко (і не треба) передбачати, який саме нейрон Кохонена активуватиметься для заданого вхідного вектору. Необхідно лише гарантувати, щоб в результаті навчання розділялися несхожі вхідні вектори.

Усім вагам мережі перед початком навчання слід надати початкові значення. Загальноприйнятою практикою при роботі з нейронними мережами являється привласнення вагам невеликих випадкових значень. При навчанні шару Кохонена випадково вибрані вагові вектори слід нормалізувати. Остаточні значення вагових векторів після навчання співпадають з нормалізованими вхідними векторами. Тому нормалізація перед початком навчання наближає вагові вектори до їх остаточних значень, скорочуючи, таким чином, повчальний процес.

Найбільш бажане рішення полягає в тому, щоб розподіляти вагові вектори відповідно до щільності вхідних векторів, які мають бути розділені, поміщаючи тим самим більше вагових векторів в околиці великого числа вхідних векторів. На практиці це нездійсненно, проте існує декілька методів наближеного досягнення тих же цілей.

Одне з рішень, відоме під назвою методу **опуклої комбінації** (convex combination method), полягає в тому, що усе ваги прирівнюються одній і тій же величині

$$w_i = \frac{1}{\sqrt{n}},$$

де n - число входів і, отже, число компонент кожного вагового вектору. Завдяки цьому усі вагові вектори співпадають і мають одиничну довжину. Кожній же компоненті входу X надається значення

$$x_i = \alpha x_i + \frac{1 - \alpha}{\sqrt{n}},$$

де n - число входів. На початку (дуже мало, внаслідок чого усі вхідні вектори мають довжину, близьку до 0, і майже співпадають з векторами ваг. В процесі навчання мережі (поступово зростає, наближаючись до одиниці. Це дозволяє розділяти вхідні вектори і остаточно приписує їм їх істинні значення. Вагові вектори відстежують один або невелику групу вхідних векторів і у кінці навчання дають необхідну картину виходів. Метод опуклої комбінації добре працює, але уповільнює процес навчання, оскільки вагові вектори налаштовуються до мети, що змінюється.

Інший підхід полягає в **додаванні шуму** до вхідних векторів. Тим самим вони піддаються випадковим змінам, схоплюючи врешті-решт ваговий вектор. Цей метод також працездатний, але ще повільніший, ніж метод опуклої комбінації.

Третій метод починає з **випадкових ваг**, але на початковій стадії повчального процесу налаштовує усе ваги, а не тільки пов'язані з нейроном, що виграв, Кохонена. Тим самим вагові вектори переміщаються ближче до області вхідних векторів. В процесі навчання корекція ваг починає вироблятися лише для найближчих до переможця нейронів Кохонена. Цей радіус корекції поступово зменшується, так що врешті-решт коригуються тільки ваги, пов'язані з нейроном, що виграв, Кохонена.

Ще один метод наділяє кожен нейрон Кохонена "Почуттям справедливості". Якщо він стає переможцем гущавині своєї законної долі часу (приблизно 1/k, де k - число нейронів Кохонена), він тимчасово збільшує свій поріг, що зменшує його шанси на виграш, даючи тим самим можливість навчатися і іншим нейронам.

У багатьох застосуваннях точність результату істотно залежить від розподілу вагів. На жаль, ефективність різних рішень вичерпним чином не оцінена і залишається проблемою.

Шар Гроссберга навчається відносно просто. Вхідний вектор, що є виходом шару Кохонена, подається на шар нейронів Гроссберга, і виходи шару Гроссберга обчислюються, як при нормальному функціонуванні. Далі, кожна вага коригується лише у тому випадку, якщо він сполучений з нейроном Кохонена, що має ненульовий вихід. Величина корекції ваги пропорційна різниці між вагою і необхідним виходом нейрона Гроссберга, з яким він сполучений. У символьному записі

$$v_{ijн} = v_{ijc} + \beta(y_j - v_{ijc})k_i, \qquad (3.6)$$

де ki - вихід і -го нейрона Кохонена (тільки для одного нейрона Кохонена він відмінний від нуля); yj - j -а компонента вектору бажаних виходів.

Спочатку (береться рівним ~0,1 і потім поступово зменшується в процесі навчання.

Звідси видно, що ваги шару Гроссберга сходитимуться до середніх величин від бажаних виходів, тоді як ваги шару Кохонена навчаються на середніх значеннях входів. Навчання шару Гроссберга - це навчання з учителем, алгоритм має в розпорядженні бажаний вихід, по якому він навчається. Шар Кохонена, що навчається без учителя, , дає виходи в недетермінованих позиціях.

3.2 Нейронна мережа Хопфілда

Серед різних конфігурацій штучних нейронних мереж (НС) зустрічаються такі, при класифікації яких за принципом навчання, строго кажучи, не підходять ні навчання з учителем , ні навчання без учителя. У таких мережах вагові коефіцієнти синапсів розраховуються тільки один раз перед початком функціонування мережі на основі інформації про оброблювані дані, і усе навчання мережі зводиться саме до цього розрахунку. З одного боку, пред'явлення апріорної інформації можна розцінювати, як допомога учителя, але з іншої - мережа фактично просто запам'ятовує зразки до того, як на її вхід поступають реальні дані, і не може змінювати свою поведінку, тому говорити про ланку зворотного зв'язку з "світом" (учителем) не доводиться. З мереж з подібною логікою робо-

ти найбільш відомі мережа Хопфілда і мережа Хеммінга, які зазвичай використовуються для організації асоціативної пам'яті.

Структурна схема мережі Хопфілда приведена на рис.17. Вона складається з єдиного шару нейронів, число яких є одночасно числом входів і виходів мережі. Кожен нейрон пов'язаний синапсами з усіма іншими нейронами, а також має один вхідний синапс, через який здійснюється введення сигналу. Вихідні сигнали, як завжди, утворюються на аксонах.

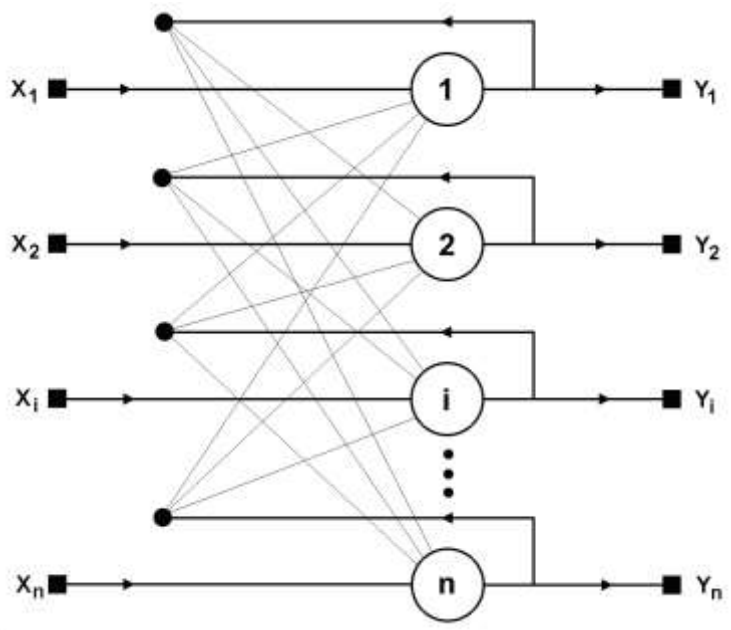

Рис. 17 Структурна схема мережі Хопфілда

Завдання, що вирішується цією мережею як асоціативна пам'ять, як правило, формулюється таким чином. Відомий деякий набір двійкових сигналів (зображень, звукових оцифрувань, інших даних, що описують деякі об'єкти або характеристики процесів), які вважаються зразковими. Мережа повинна уміти з довільного неідеального сигналу, поданого на її вхід, виділити ("згадати" за частковою інформацією) відповідний зразок (якщо такий є) або "дати ув'язнення" про те, що вхідні дані не відповідають жодному із зразків.

У загальному випадку, будь-який сигнал може бути описаний вектором X = { xi: i=1..n }, n - число нейронів в мережі і розмірність вхідних і вихідних векторів. Кожен елемент xi рівний або +1, або - 1. Позначимо вектор, k, що описує, -й зразок, через X_k, а його компоненти, відповідно, - x_{ik}, k=1..m , m - чи-

сло зразків. Коли мережа розпізнає (або "згадає") який-небудь зразок на основі пред'явлених їй даних, її виходи міститимуть саме його, тобто Y = X_k, де Y - вектор вихідних значень мережі : Y = { yi: i=0,..n - 1}. Інакше, вихідний вектор не співпаде ні з одним зразковим.

Якщо, наприклад, сигнали є деякими зображеннями, то, відобразив в графічному виді дані з виходу мережі, можна буде побачити картинку, повністю співпадаючу з однією із зразкових (у разі успіху) або ж "вільну імпровізацію" мережі (у разі невдачі).

На стадії ініціалізації мережі вагові коефіцієнти синапсів встановлюються таким чином :

$$w_{ij} = \begin{cases} \sum_{k=0}^{m-1} x_i^k x_j^k, i \neq j \\ 0, \qquad i = j \end{cases} \tag{3.7}$$

Тут i і j - індекси, відповідно, пресинаптического і постсинаптичного нейронів; x_{ik}, x_{jk} - i -й і j -й елементи вектору k -го зразка.

Алгоритм функціонування мережі наступний (p - номер ітерації) :

1. На входи мережі подається невідомий сигнал. Фактично його введення здійснюється безпосередньою установкою значень аксонів :

$$yi(0) = xi,\ i = 0..n - 1, \tag{3.8}$$

тому позначення на схемі мережі вхідних синапсів в явному виді носить чисто умовний характер. Нуль в дужці праворуч від *yi* означає нульову ітерацію в циклі роботи мережі.

2. Розраховується новий стан нейронів

$$s_j(p+1) = \sum_{i=0}^{n-1} w_{ij} y_i(p), \ j=0..n-1 \tag{3.9}$$

і нові значення аксонів

$$y_j(p+1) = F\left[s_j(p+1)\right] \tag{3.10}$$

де F - активаційна функція у вигляді стрибка, приведена на рис.18а.

3. Перевірка, чи змінилися вихідні значення аксонів за останню ітерацію. Якщо так - перехід до пункту 2, інакше (якщо виходи стабілізувались) - кінець. При цьому вихідний вектор є зразком, що якнайкраще поєднується з вхідними даними.

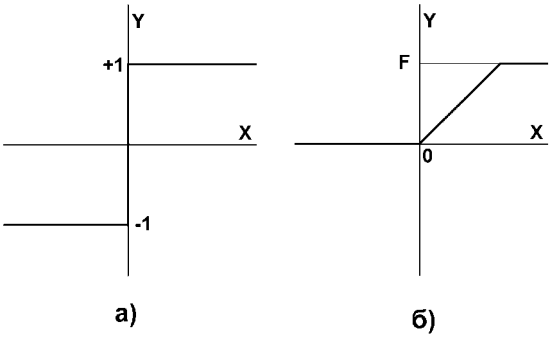

Рис. 18 Активаційні функції

Як говорилося вище, іноді мережа не може провести розпізнавання і видає на виході неіснуючий образ. Це пов'язано з проблемою обмеженості можливостей мережі. Для мережі Хопфілда число образів m, що запам'ятовуються, не повинне перевищувати величини, приблизно рівної 0.15*n. Крім того, якщо два образи А і б сильно схожі, вони, можливо, викликатимуть у мережі перехресні асоціації, тобто пред'явлення на входи мережі вектору А приведе до появи на її виходах вектору б і навпаки.

3.3 Нейронна мережа Хеммінга

Коли немає необхідності, щоб мережа в явному виді видавала зразок, тобто досить, скажімо, отримувати номер зразка, асоціативну пам'ять успішно реалізує мережа Хеммінга.

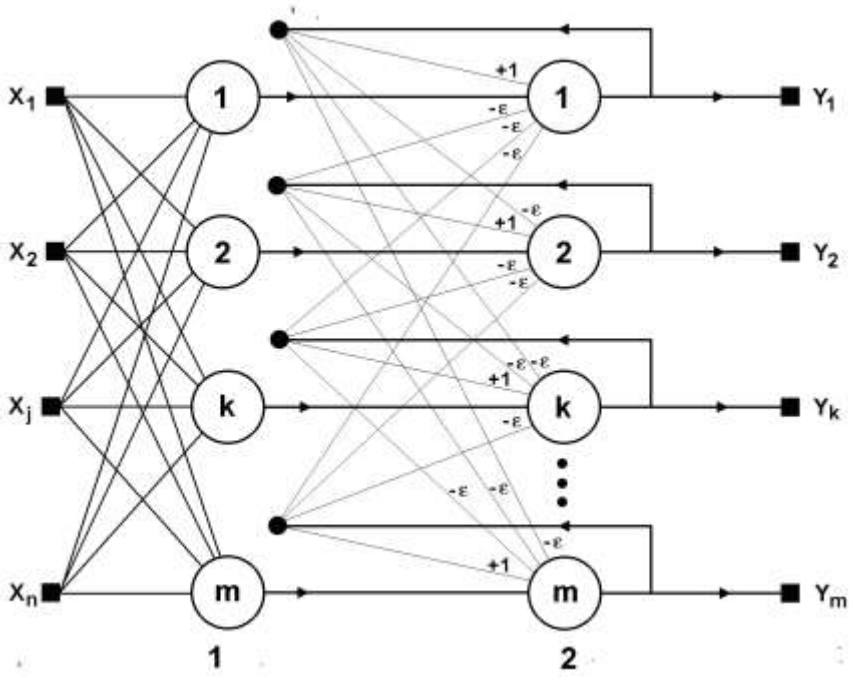

Рис. 19 Структурна схема мережі Хеммінга

Ця мережа характеризується, в порівнянні з мережею Хопфілда, меншими витратами на пам'ять і об'ємом обчислень, що стає очевидним з її структури (рис. 19).

Мережа складається з двох шарів. Перший і другий шари мають по m нейронів, де m - число зразків. Нейрони першого шару мають по n синапсів, сполучених з входами мережі (що утворюють фіктивний нульовий шар). Нейрони другого шару пов'язані між собою ингибиторными (негативними зворотними) синаптическими зв'язками. Єдиний синапс з позитивним зворотним зв'язком для кожного нейрона сполучений з його ж аксоном.

Ідея роботи мережі полягає в знаходженні відстані Хеммінга від тестованого образу до усіх зразків. Відстанню Хеммінга називається число бітів, що відрізняються, в двох бінарних векторах. Мережа повинна вибрати зразок з мінімальною відстанню Хеммінга до невідомого вхідного сигналу, внаслідок чого буде активізований тільки один вихід мережі, відповідний цьому зразку.

На стадії ініціалізації ваговим коефіцієнтам першого шару і порогу активаційної функції привласнюються наступні значення:

$$w_{ik} = \frac{x_i^k}{2},\ i=0..n-1,\ k=0..m-1 \qquad (3.11)$$

$$Tk = n/2,\ k = 0..m-1 \qquad (3.12)$$

Тут x_{ik} - і -й елемент k -го зразка.

Вагові коефіцієнти гальмівних синапсів в другому шарі беруть рівними деякій величині $0 < \varepsilon < 1/m$. Синапс нейрона, пов'язаний з його ж аксоном має вагу +1.

Алгоритм функціонування мережі Хеммінга наступний:

1. На входи мережі подається невідомий вектор X = {xi: i=0..n - 1}, виходячи з якого розраховуються стани нейронів першого шару (верхній індекс в дужках вказує номер шару) :

$$y_j^{(1)} = s_j^{(1)} = \sum_{i=0}^{n-1} w_{ij} x_i + T_j,\ j=0..m-1 \qquad (3.13)$$

Після цього набутих значень ініціалізувалися значення аксонів другого шару :

$$yj(2) = yj(1),\ j = 0..m-1 \qquad (3.14)$$

2. Вичислити нові стани нейронів другого шару :

$$s_j^{(2)}(p+1) = y_j(p) - \varepsilon \sum_{k=0}^{m-1} y_k^{(2)}(p),\ k \neq j,\ j = 0...m-1 \qquad (3.15)$$

і значення їх аксонів :

$$y_j^{(2)}(p+1) = F\left[s_j^{(2)}(p+1)\right],\ j = 0...m-1 \qquad (3.16)$$

Активаційна функція F має вигляд порогу (мал. 2б), причому величина F має бути досить великою, щоб будь-які можливі значення аргументу не призводили до насичення.

3. Перевірити, чи змінилися виходи нейронів другого шару за останню ітерацію. Якщо так - перейди до кроку 2. Інакше - кінець.

З оцінки алгоритму видно, що роль першого шару дуже умовна: скориставшись один раз на кроці 1 значеннями його вагових коефіцієнтів, мережа більше не звертається до нього, тому перший шар може бути взагалі виключений з

мережі (замінений на матрицю вагових коефіцієнтів), що і було зроблено в її конкретній реалізації, описаній нижче.

3.4 Двонаправлена асоціативна пам'ять (ДАП)

Пам'ять людини часто є асоціативною; один предмет нагадує нам про інше, а цей інший про третє. Якщо дозволити нашим думкам, вони переміщатимуться від предмета до предмета по ланцюжку розумових асоціацій. Крім того, можливе використання здатності до асоціацій для відновлення забутих образів.

Двонаправлена асоціативна пам'ять (ДАП), яку запропонував Бартоломео Коско, є гетероассоціативною; вхідний вектор поступає на один набір нейронів, а відповідний вихідний вектор виробляється на іншому наборі нейронів. Вона є логічним розвитком парадигми мережі Хопфілда, до якої для цього досить додати другий шар. Структура ДАП представлена на рис.20.

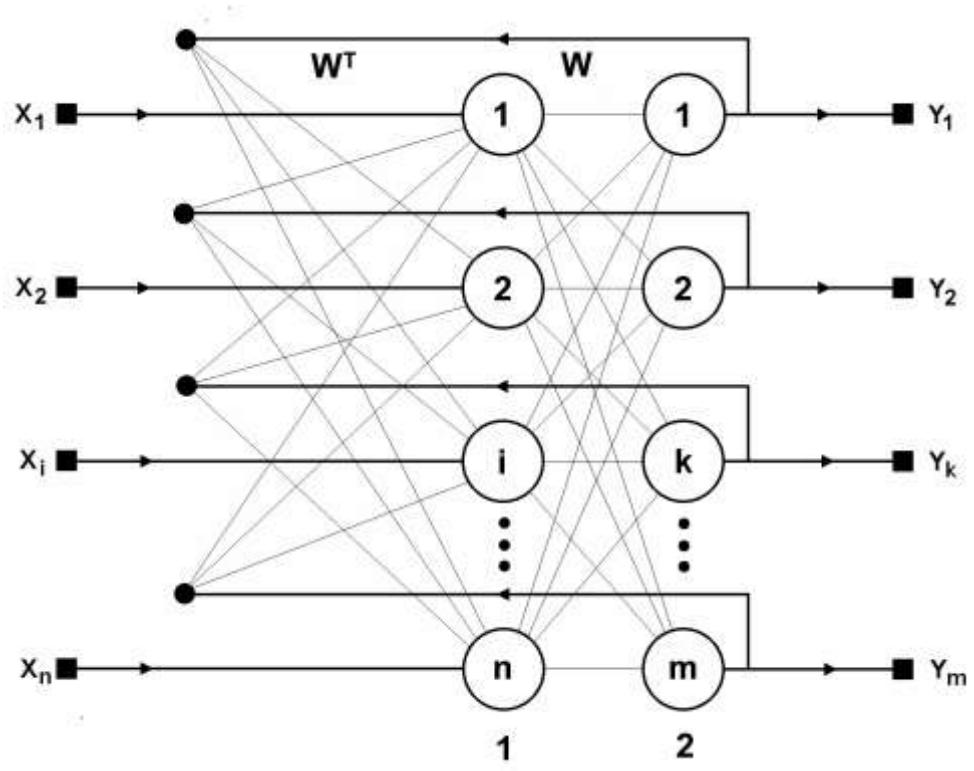

Рис. 20 Структурна схема ДАП

Мережа здатна запам'ятовувати пари асоційованих один з одним образів. Нехай пари образів записуються у вигляді векторів-рядків:

$\mathbf{X_k} = \{x_{ip}: i=1...n\}$ і $\mathbf{Y_k} = \{y_{jp}: j=1...m\}$,

p=1...M , де M - число пар.

Подача на вхід першого шару деякого вектору $\mathbf{P} = \{p_i: i=1...n\}$ викликає утворення на вході другого шару деякого іншого вектору $\mathbf{Q} = \{q_j: j=1..m\}$, який потім знову поступає на вхід першого шару. При кожному такому циклі вектору на виходах обох шарів наближаються до пари зразкових векторів, перший з яких - **X** - найбільш схожий на **P**, який був поданий на вхід мережі на самому початку, а другий - **Y** - асоційований з ним на **Q**. Асоціації між векторами кодуються у ваговій матриці **W** першого шару. Вагова матриця другого шару дорівнює транспонованій першій - $\mathbf{W^T}$.

Процес навчання, також як і у мережі Хопфілда, полягає в попередньому обчисленні елементів матриці **W** (і відповідно $\mathbf{W^T}$) за формулою:

$$w_{ij} = \sum_k x_i y_j, i = 1...n, j = 1...m \tag{3.17}$$

Ця формула є розгорнутим записом матричного рівняння

$$\mathbf{W} = \sum_{k=1}^{M} \mathbf{X}_k^{\mathrm{T}} \mathbf{Y}_k \tag{3.18}$$

для окремого випадку, коли образи записані у вигляді векторів, при цьому добуток двох матриць розміром відповідно [n*1] і [1*m] призводить до (11).

Якщо задана активаційна функція F(S), то вихідні вектори обчислюються за формулами:

$$\mathbf{Y}=F(\mathbf{X}\ \mathbf{W}), \tag{3.19}$$

$$\mathbf{X}=F(\mathbf{Y}\ \mathbf{W^T}) \tag{3.20}$$

При цьому, бінарні вектори рекомендується кодувати -1 та +1 і активаційна функція повинна повертати значення на проміжку[-1,+1].

При функціонуванні ДАП вектори **X** і **Y** обчислюються по черзі до стабілізації значень. Стабілізовані значення і будуть відновленою парою асоціативних образів, що збігається з найближчою парою із навчальної вибірки.

Об’єм ДАП- кількість пар образів, що може відновлюватись, визначається формулою:

$$M \le \frac{\min(m,n)}{4\log_2 \min(m,n)} \tag{3.21}$$

У висновку можна зробити наступне узагальнення. Мережі Хопфілда, Хеммінга і ДАП дозволяють просто і ефективно дозволити завдання відтворення образів за неповною і спотвореною інформацією. Невисока місткість мереж (число образів, що запам'ятовуються) пояснюється тим, що, мережі не просто запам'ятовують образи, а дозволяють проводити їх узагальнення.

3.5 Рекурентні нейронні мережі

Мережа Джордана - частково рекурентна мережа (Jordan networks are partially recurrent networks and similar to Elman networks).

Її можна розглядати як мережа прямого поширення з додатковими нейронами контексту у вхідному шарі.

Ці контекстні нейрони приймають вхід від себе (прямий зворотний зв'язок, direct feedback) і з вихідних нейронів. Контекстні нейрони зберігають поточний стан мережі. У мережі Джордана кількість контекстних і вихідних нейронів має бути однаковою.

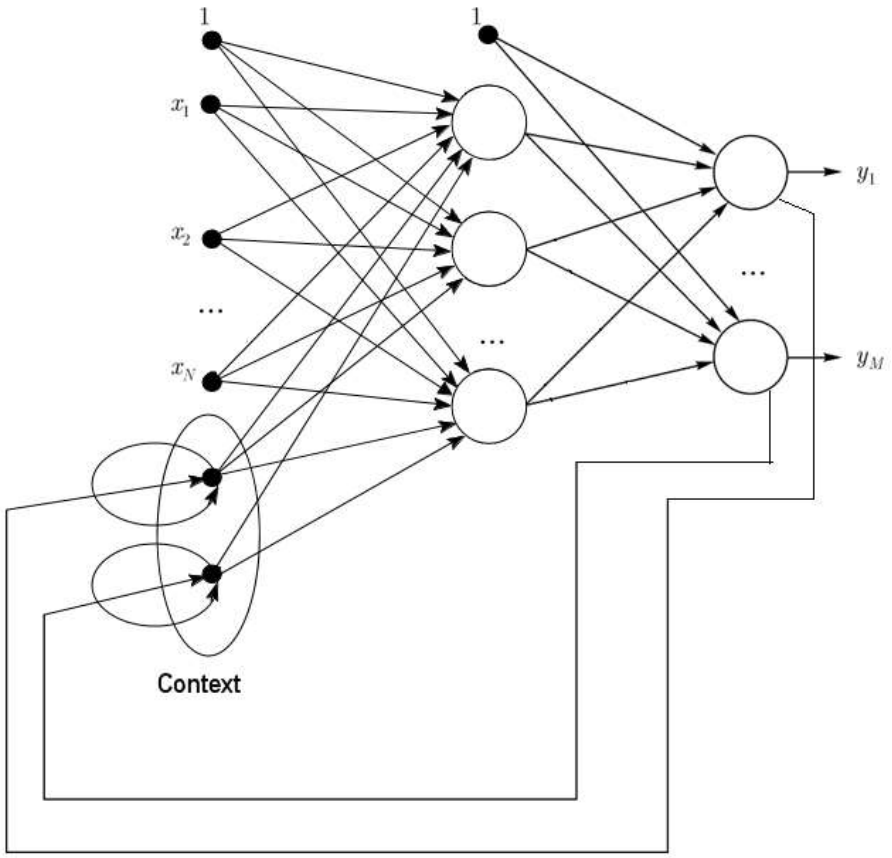

Рис. 21 Структурна схема мережі Джордана

Мережа Елмана - частково рекурентна мережа, подібна до мереж Джордана (Elman networks are partially recurrent networks and similar to Jordan networks). Різниця між мережею Елмана і Джордана в тому, що в мережі Елмана контекстні нейрони беруть вхід не від вихідних нейронів, а від прихованих. Крім того, в контекстних нейронах немає ніякого прямого зворотного зв'язку.

У мережі Елмана число контекстних і прихованих нейронів має бути однаковим. Головна перевага мереж Елмана полягає в тому, що число контекстних нейронів визначається не розмірністю виходу (як в мережі Джордана) а кількістю прихованих нейронів, що робить її гнучкішою. Можна легко додати або прибрати приховані нейрони, на відміну від кількості виходів.

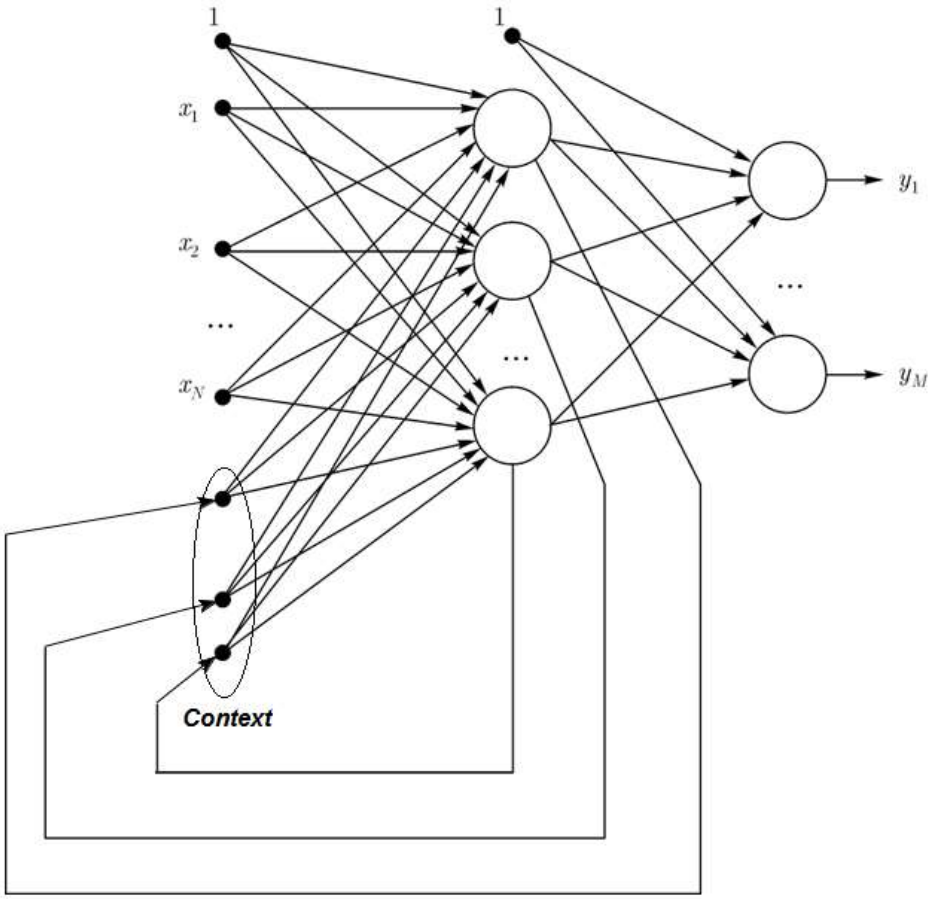

Рис. 22 Структурна схема мережі Елмана

Мережа радіальних базисних функцій (RBF) - нейронна мережа прямого поширення, яка містить проміжний (прихований) шар радіально симетричних нейронів. Такий нейрон перетворює відстань від цього вхідного вектору до відповідного йому "центру" за деяким нелінійним законом (зазвичай функція Гауса).

Мережі RBF мають ряд переваг перед багатошаровими мережами прямого поширення. По-перше, вони моделюють довільну нелінійну функцію за допомогою усього одного проміжного шару, тим самим, позбавляючи розробника від необхідності вирішувати питання про число шарів. По-друге, параметри лінійної комбінації у вихідному шарі можна повністю оптимізувати за допомогою добре відомих методів лінійної оптимізації, які працюють швидко і не зазнають труднощів з локальними мінімумами, що так заважають при навчанні з використанням алгоритму зворотного поширення помилки. Тому мережа RBF

навчається дуже швидко - на порядок швидше, ніж з використанням алгоритму ОР (зворотного поширення).

Недоліки мереж RBF : ці мережі мають погані екстраполюючі властивості і виходять дуже громіздкими при великій розмірності вектору входів.

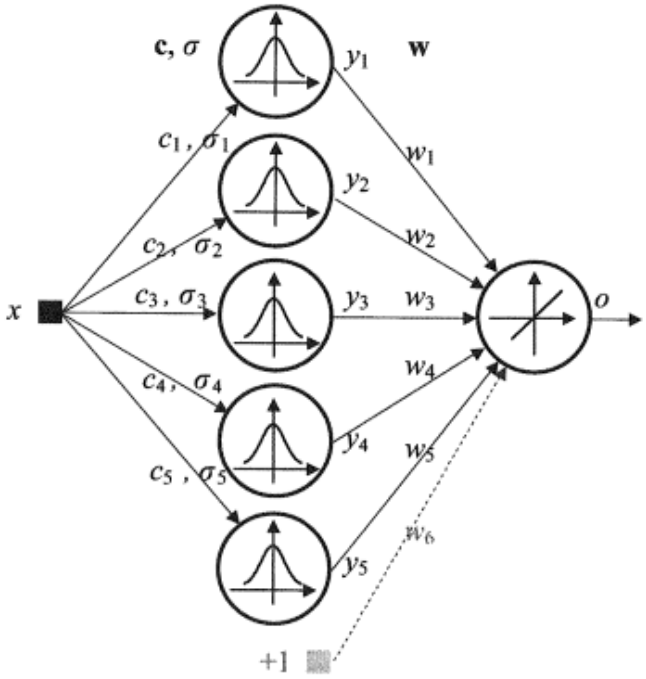

Рис. 21 Структурна схема RBF

Динамічні мережі навченого векторного квантування (Dynamic learning vector quantization, DLVQ networks) схожі на карти Кохонена (SOM), що самоорганизующиеся. На відміну від SOM, вона навчається з учителем, і в ній відсутні сусідські стосунки між прототипами (But they perform supervised learning and lack a neighborhood relationship between the prototypes). Векторне квантування є набагато загальнішою операцією, ніж кластеризація.

3.6 Глибинне навчання НМ

Глибинне навчання (англ. Deep learning) - набір алгоритмів машинного навчання, які намагаються моделювати високо рівневі абстракції в даних, використовуючи архітектури, що складаються з безлічі нелінійних трансформацій.

Під терміном «глибина» в даному випадку розуміється глибина графа обчислень моделі - максимальна довжина між вхідним і вихідним вузлами конкретної архітектури. У разі, наприклад, простий нейронної мережі прямого поширення глибина відповідає кількості шарів мережі. Термін глибинне навчання акцентує увагу на складності навчання внутрішніх (глибоких) шарів багатошарової мережі, які погано піддаються класичним методам навчання, таким як метод зворотного поширення помилки.

Глибинне навчання є частиною більш широкого сімейства методів машинного навчання, заснованих на репрезентаційних навчанні. Спостереження (наприклад, зображення) можуть бути представлені по-різному (вектор пікселів і т. Д.), Але деякі репрезентації дозволяють легше вирішувати поставлені завдання (наприклад, чи є це зображення особою?). Дослідження в цій області намагаються визначити які репрезентації корисніші і як створити моделі, які могли б навчитися створювати такі репрезентації.

Різні архітектури глибинного навчання, такі як глибокі нейромережі, згорткові нейронні мережі, глибокі мережі довіри, використовувалися в таких областях як комп'ютерний зір, розпізнавання мови, обробка природної мови, де в деяких завданнях вони показували кращі на даний момент результати.

Глибокі архітектури, засновані на штучних нейронних мережах, беруть свій початок з неокогнітрона, розробленого Куніхіко Фукусімою в 1980 році. Самі нейронні мережі з'явилися ще раніше. У 1989 році Яну Лекуну вдалося використати алгоритм зворотного поширення помилки для навчання глибоких нейромереж для вирішення задачі розпізнавання рукописних ZIP-кодів . Незважаючи на успішний досвід, для навчання моделі було потрібно три дні, що істотно обмежувало застосування цього методу. Низька швидкість навчання пов'язана з багатьма факторами, включаючи проблему зникаючого градієнта, яку в 1991 році аналізували Юрген Шмідхубер і Сеппо Хохрайтер. Через ці проблем нейронні мережі в 1990-х роках поступилися місцем методу опорних векторів.

Термін «глибинне навчання» придбав популярність після публікації Джеффрі Хинтона і Руслана Салахутдінова в середині 2000-х років, в якій вони показали, що можна ефективно попередньо навчити багатошарову нейронну мережу, якщо навчати кожен шар окремо за допомогою обмеженої машини Больцмана, а потім донавчати за допомогою методу зворотного поширення помилки.

3.7 Теми та алгоритми лабораторних робіт

Лабораторна робота № 5

З допомогою мережі Хопфілда відновити зашумлені графічні зображення(рис. 22б).

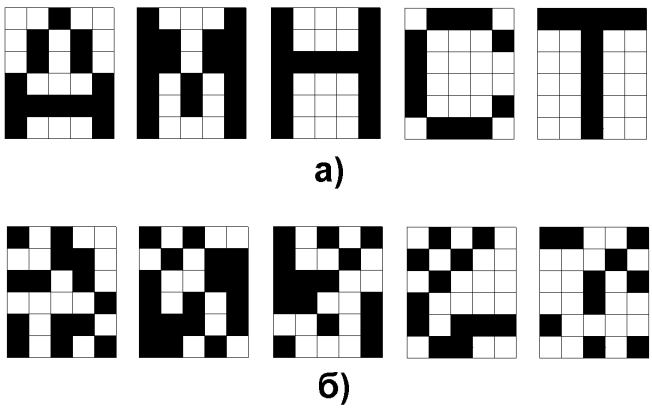

Рис. 22 Зразкові і тестові образи

Алгоритм лабораторної роботи № 5

І. Фаза навчання

1. Формуємо зразкові графічні образи навчальної вибірки.

2. Представляємо двовимірні зображення у вигляді одновимірних бінарних векторів скануючи зображення по рядках.

3. На стадії ініціалізації мережі вагові коефіцієнти синапсів встановлюються таким чином :

$$w_{ij} = \begin{cases} \sum_{k=0}^{m-1} x_i^k x_j^k, i \neq j \\ 0, \qquad i = j \end{cases}$$

Тут і і j - індекси, відповідно, предсинаптического і постсинаптичного нейронів; x_{ik}, x_{jk} - і -й і j -й елементи вектору k -го зразка навчальної вибірки.

II. Фаза функціонування

1. На входи мережі подається невідомий сигнал. Фактично його введення здійснюється безпосередньою установкою значень аксонів :

$$yi(0) = xi,\ i = 0..n - 1,$$

тому позначення на схемі мережі вхідних синапсів в явному виді носить чисто умовний характер. Нуль в дужці праворуч від уі означає нульову ітерацію в циклі роботи мережі.

2. Розраховується новий стан нейронів і їх виходів

$$s_j(t+1) = \sum_{i=0}^{n-1} w_{ij} y_i(t), \quad j=0..n - 1$$

і нові значення аксонів

$$y_j(t+1) = F\left[s_j(t+1)\right]$$

де F - активаційна функція у вигляді стрибка.

3. Перевірка, чи змінилися вихідні значення аксонів за останню ітерацію. Якщо так - перехід до пункту 2, інакше (якщо виходи стабілізувались) - кінець. При цьому вихідний вектор є зразком, що якнайкраще поєднується з вхідними даними.

Лабораторна робота № 6

Реалізувати програмну систему, що відновлює асоціативні пари графічних образів за неповним чи неточним представлення одного з них. Це можуть бути , наприклад, пари, що складаються з малюнків і надписів до них.

Алгоритм лабораторної роботи № 6

I. Фаза навчання

4. Будуємо два графічних вікна для введення асоційованих пар образів у вигляді чорних і білих мегапікселів на сітках. Ініціюємо нульову матрицю **W**=0 розміром n*m.
5. Вносимо асоційовану k –ту пару образів з навчальну вибірки у графічні вікна.
6. Перетворюємо графічні образи з кожного вікна у одновимірні бінарні вектори $\mathbf{X_k[n]}$ і $\mathbf{Y_k[m]}$, скануючи графічні вікна, наприклад, по рядках (чорний піксель - (+1), білий - (-1)).
7. Знаходимо матричний добуток векторів $\mathbf{X}_k^T\ \mathbf{Y}_k$ і додаємо його до **W.**
8. Повторюємо пункт 2. до вичерпання всіх навчальних пар. Отримуємо матрицю ДАП:

$$\mathbf{W} = \sum_{k=1}^{M} \mathbf{X}_k^T\ \mathbf{Y}_k$$

II. Фаза функціонування

1. У одне з графічних вікон вносимо графічний образ (можливо зашумлений).
2. Перетворюємо графічні образи з кожного вікна у одновимірні бінарні вектори Xk[n] чи Yk[m], скануючи графічні вікна, наприклад, по рядках (чорний піксель - (+1), білий - (-1)).
3. Застосовуємо по-черзі формули

$$\mathbf{Y}=F(\mathbf{X}\ \mathbf{W}),$$

$$\mathbf{X}=F(\mathbf{Y}\ \mathbf{W}^T),$$

4. починаючи з тієї, у праву частину якої, входить заданий вектор (якщо заданий вектор X, то починаємо з першої формули, якщо Y, то – з другої).
5. Пункт 3. повторюємо до стабілізації, тобто поки різниця між значеннями двох ітерацій не будуть збігатися:

$$|\mathbf{X}^{(t+1)}-\mathbf{X}^{(t)}|< \varepsilon$$

$$|\mathbf{Y}^{(t+1)}-\mathbf{Y}^{(t)}|< \varepsilon$$

6. Відображаємо відновлені вектори у відповідних графічних вікнах (рис 23).

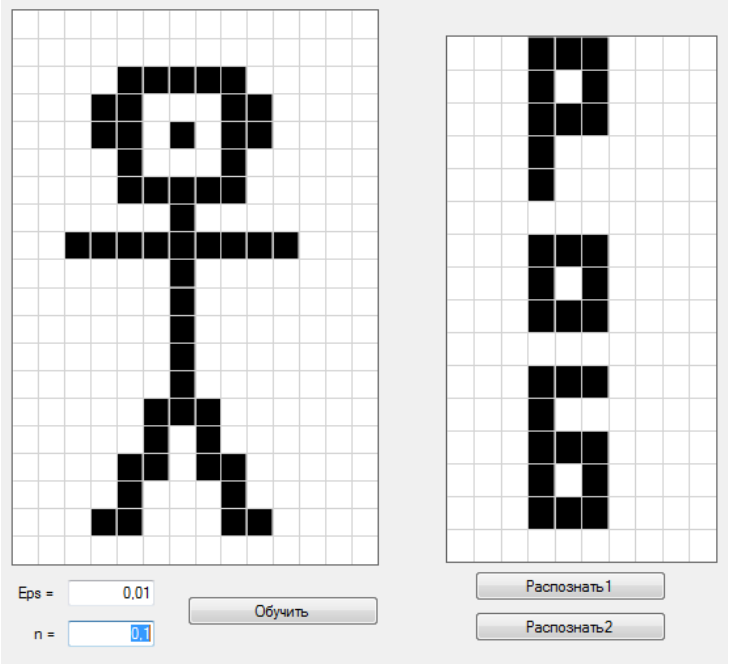

Рис. 23 Приклад графічних вікон з образами

ВИСНОВКИ

Цілком очевидно, що свою силу нейронні мережі черпають, по-перше, з розпаралелювання обробки інформації і, по-друге, у здібності до самонавчання, тобто створювати узагальнення. Під терміном узагальнення розуміється здатність отримувати обґрунтований результат на підставі даних, які не зустрічались у процесі навчання. Ці властивості дозволяють нейронним мережам вирішувати складні (масштабні) завдання, які на сьогоднішній день вважаються

важко вирішуваними. Однак на практиці при автономній роботі нейронні мережі не можуть забезпечити готові рішення. Їх необхідно інтегрувати в складні системи. Зокрема, комплексне завдання можна розбити на послідовність простих, частина з яких може вирішуватися нейронними мережами.

Наведемо деякі переваги нейронних мереж перед традиційними обчислювальними системами.

Рішення задач при невідомих закономірностях. Використовуючи здатність навчання на множині прикладів, нейронна мережа здатна вирішувати завдання, в яких невідомі закономірності розвитку ситуації і залежності між вхідними та вихідними даними. Традиційні математичні методи та експертні системи в таких випадках пасують.

Стійкість до шумів у вхідних даних. Можливість роботи при наявності великої кількості неінформативних, шумових вхідних сигналів. Немає необхідності робити їх попередній відсів, нейронна мережа сама визначить їх малопридатними для вирішення завдання і відкине їх.

Адаптація до змін навколишнього середовища. Нейронні мережі мають здатність адаптуватися до змін навколишнього середовища. Зокрема, нейронні мережі, навчені діяти в певному середовищі, можуть легко перевчити для роботи в умовах незначних коливань параметрів середовища. Більш того, для роботи в нестаціонарному середовищі (де статистика змінюється з плином часу) можуть бути створені нейронні мережі, які можна перенавчати в реальному часі. Чим вище адаптивні здібності системи, тим більш стійкою буде її робота в нестаціонарному середовищі. Для того щоб використовувати всі переваги адаптивності, основні параметри системи повинні бути досить стабільними, щоб можна було не враховувати зовнішні перешкоди, і досить гнучкими, щоб забезпечити реакцію на істотні зміни середовища.

Потенційно надвисока швидкодія. Нейронні мережі мають потенційно надвисоку швидкодію за рахунок використання масового паралелізму обробки інформації;

Відмовостійкість при апаратній реалізації нейронної мережі. Нейронні мережі потенційно відмовостійкі. Це означає, що при несприятливих умовах їх продуктивність падає незначно. Однак, беручи до уваги розподілений характер зберігання інформації в нейронній мережі, можна стверджувати, що тільки серйозні пошкодження структури нейронної мережі істотно вплинуть на її працездатність.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Хайкин С. Нейронные сети: полный курс, 2-е изд. — М.: Вильямс, 2006. — 1104 с.
2. Осовский С.,Нейронные сети для обработки информации, — М.: Финансы и статистика, 2002. — 344 с.: ил.
3. Короткий С., Нейронные сети: основние положения. Електронный ресурс
4. Короткий С. , "Нейронные сети: Алгоритм обратного распространения" Електронный ресурс
5. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика . М.: Мир, 1992. — 240 с.
6. Горбань А.Н., Россиев Д.А. Нейронные сети на персональном компьютере — Новосибирск: Наука, 1996. — 276 с.
7. Терехов В. А., Ефимов Д. В., Тюкин И. Ю. Нейросетевые системы управления. — М.: Высшая школа, 2002. — 184 с.
8. Sankar K. Pal, Sushmita Mitra, Multilayer Perceptron, Fuzzy Sets, and Classification //IEEE Transactions on Neural Networks, Vol.3, N5, 1992, pp.683-696.
9. Bernard Widrow, Michael A. Lehr, 30 Years of Adaptive NeuralNetworks : Perceptron, Madaline, and Backpropagation //Artificial Neural Networks : Concepts and Theory, IEEE Computer Society Press, 1992, pp.327-354.
10. JOHN MARKOFFNOV, Scientists See Promise in Deep-Learning Programs . 23, 2012, Електронний ресурс: http://www.nytimes.com/2012/11/24/science/scientists-see-advances-in-deep-learning-a-part-of-artificial-intelligence.html?_r=0
11. D. Hinton, Нейронные сети в обучении машин , Електронний ресурс: https://www.coursera.org/course/neuralnets