МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторної роботи

ДОСЛІДЖЕННЯ ПЕРСЕПТРОНА

з дисципліни

ШТУЧНІ НЕЙРОННІ МЕРЕЖІ В ЗАДАЧАХ ОБРОБКИ ДАНИХ

Одеса – 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторної роботи

ДОСЛІДЖЕННЯ ПЕРСЕПТРОНА

з дисципліни

ШТУЧНІ НЕЙРОННІ МЕРЕЖІ В ЗАДАЧАХ ОБРОБКИ ДАНИХ

Узгоджено
на факультеті магістерської
підготовки

Одеса – 2019

Методичні вказівки до лабораторної роботи „Дослідження персептрона” з дисципліни „Штучні нейронні мережі в задачах обробки даних” для студентів 1 курсу магістратури, що навчаються за спеціальністю „Комп'ютерні науки” / Перелигін Б.В., Ткач Т.Б. – Одеса, ОДЕКУ, 2019. – 39 с.

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторної роботи
„Дослідження персептрона”
з дисципліни
„Штучні нейронні мережі в задачах обробки даних”

Укладачі: к.т.н., доц. Перелигін Б.В., к.ф.-м.н. Ткач Т.Б.

Підписано до друку Формат Папір
Ум. друкарських листів Тираж Замовлення №

Надруковано з готового оригінал-макету

Одеський державний екологічний університет
65016, Одеса, вул. Львівська, 15

ЗМІСТ

стор.

ВСТУП

Дисципліна „Штучні нейронні мережі в задачах обробки даних” є дисципліною підготовки магістрів за спеціальністю „Комп'ютерні науки”. Вона знайомить майбутніх фахівців з сучасними методами обробки і аналізу даних.

Мета дисципліни – підготовка майбутніх фахівців в області обробки і аналізу інформації, що одержується від систем моніторингу.

Останнім часом обробка і аналіз подібної інформації стала особливо актуальною. Існуючі багаточисельні системи метеорологічного, гідрологічного, океанологічного, екологічного моніторингу наземного і космічного базування, радіолокаційні системи дистанційного моніторингу надають велику кількість важливої інформації, яка потребує грамотної обробки для отримання її характеристик з метою подальшого прийняття рішень системами управління різного рівня. Фахівці з подібними знаннями необхідні і в державних структурах, і в наукових установах, і в комерційних фірмах.

У дисципліні „Штучні нейронні мережі в задачах обробки даних” розглядаються сучасні методи обробки і аналізу моніторингової інформації, засновані на теорії штучних нейронних мереж, та вивчаються технічні і програмні засоби, що реалізовують її.

Тому головною задачею дисципліни є ознайомлення студентів з застосуванням нейромережних технологій обробки моніторингової інформації, що надходить від систем вимірювання параметрів стану навколишнього середовища, штучних супутників Землі, радіолокаційних станцій і інших систем моніторингу. В результаті її обробки можливе отримання значущих характеристик для прийняття обґрунтованих рішень в системах управління різних ієрархічних рівнів.

Практична частина дисципліни включає лабораторні роботи з вивчення і дослідження способів і методів обробки і аналізу моніторингової інформації.

Ця лабораторна робота присвячена вивченню і дослідженню класу штучних нейронних мереж званих персептронами.

У результаті підготовки і проведення лабораторної роботи „Дослідження персептрона” студенти повинні набути:

знання:

- про архітектуру одношарового і багатошарового персептрона,
- про дельта–правило навчання одношарового персептрона і алгоритм зворотного поширення помилки,
- про можливості персептронів,

уміння:

- застосовувати персептрони для вирішення задач обробки моніторингової інформації.

У цих методичних вказівках наводяться теоретичні відомості, необхідні для виконання лабораторної роботи, а також мета, завдання і порядок виконання роботи. Наведені також вимоги до оформлення звіту про лабораторну роботу.

При виконанні лабораторної роботи кожен студент відповідає на теоретичні запитання і, потім, після отримання допуску, практично виконує роботу.

Оцінюється лабораторна робота в межах виділених на неї в робочій навчальній програмі балів, причому 50% цих балів припадає на оцінку готовності студента до лабораторної роботи з теоретичних питань і 50% – на оцінку практичного виконання роботи. При отриманні студентом позитивної оцінки за відповідь на теоретичні запитання він одержує допуск до виконання лабораторної роботи, після чого практично виконує лабораторну роботу. Якщо у студента немає допуску, то і роботу він не виконує.

Після демонстрації викладачу результатів виконання лабораторної роботи і отримання його дозволу студент оформляє звіт. Після оформлення захищає звіт у вигляді відповідей на питання викладача про хід виконання лабораторної роботи і про результати роботи.

Проводиться лабораторна робота в комп'ютерному класі на персональних електронно-обчислювальних машинах зі встановленою системою комп'ютерної математики.

Вимоги правил техніки безпеки при проведенні лабораторної роботи на персональних ЕОМ

1) Ввімкнути апаратуру комп'ютера вимикачами на корпусах в послідовності: стабілізатор напруги, відеодисплейний термінал, процесор.
2) Відрегулювати яскравість свічення екрану відеодисплейного терміналу, фокусування, контрастність. Не слід встановлювати велику яскравість свічення екрану, щоб уникнути стомлення очей. Її слід встановити так, щоб відношення яскравості екрану до яскравості навколишніх поверхонь в робочій зоні було не більше, ніж 3:1.
3) Під час роботи за клавіатурою сидіти прямо, не напружуватися.
4) Для зменшення несприятливого впливу на користувача пристрою управління маркером „миша” слід зайняти велику поверхню столу для переміщення „миші” і для зручного упору ліктьового суглоба.
5) Після закінчення роботи вимкнути апаратуру в порядку, зворотному включенню.
6) Під час лабораторної роботи не дозволяються сторонні розмови, створення дратівливих шумів.

При проведенні лабораторної роботи
ЗАБОРОНЯЄТЬСЯ:

1) Користуватися кабелями і дротами з пошкодженою ізоляцією.
2) Залишати під напругою кабелі і дроти з неізольованими провідниками.
3) Застосовувати саморобні подовжувачі, що не відповідають вимогам Правил побудови електроустановок.
4) Використовувати пошкоджені електричні розетки.
5) При необхідності перемикання мережних кабелів робити це тільки при вимкненому електричному живленні комп'ютера.
6) Класти будь-які предмети на апаратуру комп'ютера.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ ПО ЛАБОРАТОРНІЙ РОБОТІ

„Дослідження персептрона”

1.1 Загальні відомості

Штучні нейронні мережі засновані на простій біологічній моделі нервової системи, що складається з 10^{11} нейронів, кожний з яких приймає зважену суму вхідних сигналів і за певних умов передає сигнал іншим нейронам. Кількість зв'язків нейронів в системі досягає 10^{15}.

Теорія нейронних мереж виникла з досліджень мозку і пов'язана із спробами відтворення здатності нервових біологічних систем до навчання і виправлення помилок, моделюючи низькорівневу структуру мозку.

Ця теорія розвивалася з середини 20 століття і з кінця 90-х років знайшла широке практичне вживання. В космонавтиці і аеронавтиці – для імітації траєкторій польоту і побудови систем автоматичного пілотування. У військовій справі – для управління зброєю і стеженням за цілями. В електроніці – для розробки систем машинного зору і синтезу мови. В медицині – для діагностики захворювань і конструювання протезів. У виробництві – для управління технологічними процесами, роботами і т.д. Такий успіх нейронних мереж пояснюється тим, що була створена необхідна елементна база для реалізації нейронних мереж, а також розроблені потужні інструментальні засоби для їх моделювання у вигляді пакетів прикладних програм. До числа подібних пакетів відноситься пакет Neural Networks Toolbox (NNT) системи комп'ютерної математики (СКМ) MATLAB 6 фірми MathWorks.

Пакет прикладних програм NNT містить засоби для побудови нейронних мереж, що базуються на поведінці математичного аналога нейрона. Пакет забезпечує ефективну підтримку проектування, навчання, аналізу і моделювання багатьох відомих типів мереж – від базових моделей персептрона до асоціативних мереж і мереж, що самоорганізовуються. Для кожного типу архітектури і навчальних правил є функції ініціалізації, навчання, адаптації, створення, моделювання, відображення, оцінки і демонстрації, а також приклади вживання.

1.2 Одношаровий персептрон

Перше систематичне вивчення штучних нейронних мереж було зроблено Мак-Каллоком і Піттсом в 1943 р. Проста нейронна модель, показана на рис. 1.1, використовувалася в більшій частині їх роботи.

Їй відповідає модель штучного нейрона (рис. 1.2). Ці системи (і їм подібні) одержали назву персептронів.

В принципі описуються і складніші системи (рис. 1.3). Вони

складаються з одного шару штучних нейронів, з'єднаних за допомогою вагових коефіцієнтів з безліччю входів.

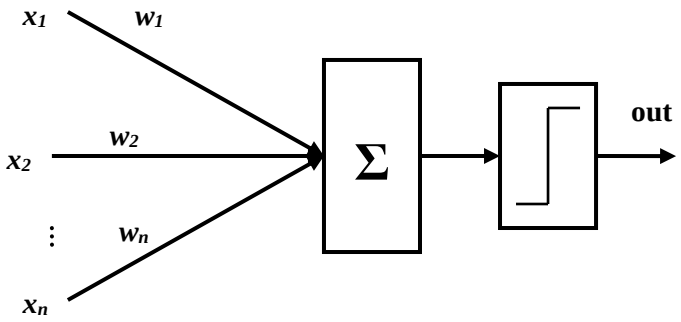


Рис. 1.1 – Персептронний нейрон Уоррена Мак-Каллока і Вальтера Піттса

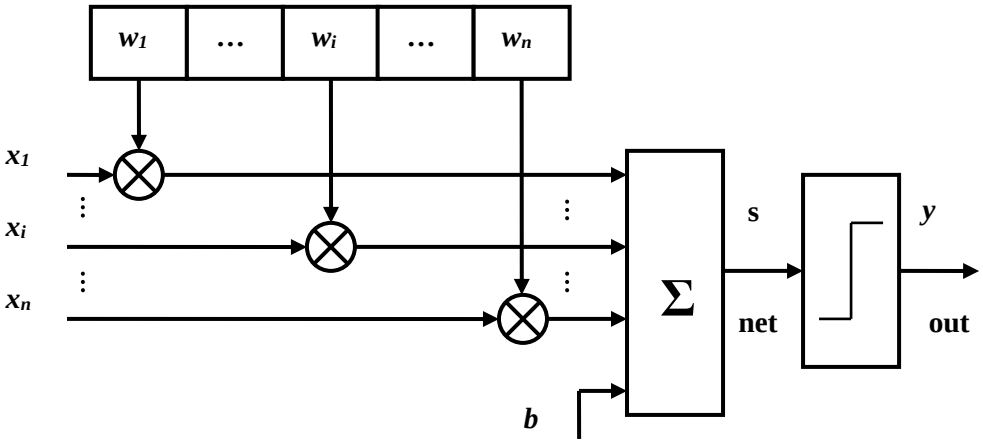


Рис. 1.2 – Структурна схема одношарового персептрона з одним виходом

У 60-і роки 20 століття персептрони викликали великий інтерес. Розенблатт довів теорему про навчання персептронів. Інші дослідники дали ряд переконливих демонстрацій систем персептронного типу, і дослідники у всьому світі прагнули вивчити можливості цих систем. Первинний оптимізм змінився розчаруванням, коли виявилося, що персептрони не здатні навчитися розв'язанню ряду простих задач.

Мінський і Пайперт наприкінці 1960-х років в першому виданні книги „Персептрони" математично строго проаналізували цю проблему і показали, що є жорсткі обмеження на те, що можуть виконувати одношарові персептрони, і, отже, на те, чому вони можуть навчатися. Оскільки у той час методи навчання багатошарових мереж не були відомі, остільки дослідження в області нейронних мереж прийшли в спадок приблизно на 15 років. Розробка методів навчання багатошарових мереж в

значній мірі вплинула на відродження інтересу до мереж.

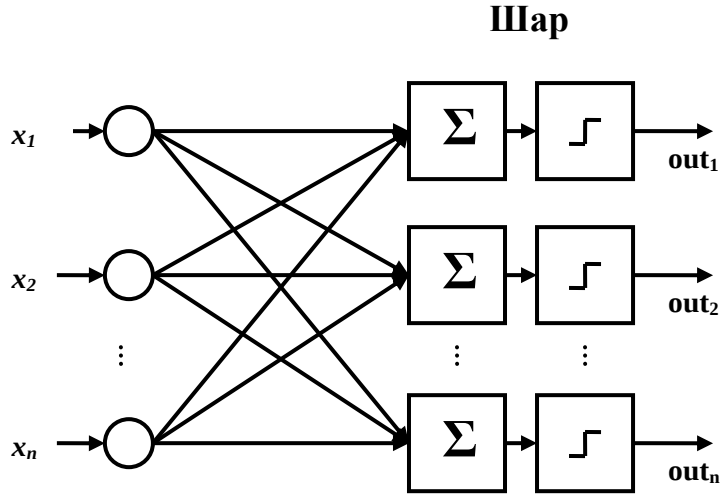


Рис. 1.3 – Архітектура одношарового персептрона з багатьма виходами

Не дивлячись на свої обмеження, одношарові персептрони широко вивчалися. Теорія персептронів є основою для багатьох інших типів штучних нейронних мереж. Через ці причини вони є вихідним матеріалом для вивчення штучних нейронних мереж.

Навчання одношарового персептрона.

Здатність штучних нейронних мереж навчатися є їх специфічною властивістю. Подібно біологічним системам, які вони моделюють, ці нейронні мережі самі моделюють себе в результаті спроб досягти кращої моделі поведінки. Персептрон навчають, подаючи безліч образів поодинці на його вхід і підстроюючи ваги до тих пір, поки для всіх образів не буде досягнутий необхідний вихід.

Таким чином, навчання мережі полягає в підстроюванні вагових коефіцієнтів кожного нейрона.

Хай є набір пар векторів (x^k, y^k), k = 1...N, званий *навчальною вибіркою*. Називатимемо нейронну мережу навченою на даній повчальній вибірці, якщо при подачі на входи мережі кожного вектора x^k на виходах всякий раз виходить відповідний вектор y^k.

Запропонований Ф.Розенблаттом метод навчання полягає в ітераційному підстроюванні матриці ваг, що послідовно зменшує помилку у вихідних векторах.

Алгоритм включає декілька кроків:

0) Початкові значення ваг всіх нейронів $W(t=0)$ вважаються випадковими.
1) Мережі пред'являється вхідний образ x^k, в результаті формується вихідний образ $\tilde{y}^k = y^k$.

2) Обчислюється вектор помилки $\delta^k = \left(y^k - \tilde{y}^k\right)$, що робиться мережею на виході. Подальша ідея полягає в тому, що зміна вектора вагових коефіцієнтів в області малих помилок повинна бути пропорційна помилці на виході, і дорівнює нулю, якщо помилка дорівнює нулю.
3) Вектор ваг модифікується по формулі $W(t+\Delta t)=W(t)+hx^k \cdot \left(\delta^k\right)^T$. Тут 0<h<1 – темп навчання.
4) Кроки 1 – 3 повторюються для всіх навчальних векторів. Один цикл послідовного пред'явлення всієї вибірки називається *епохою.*

Навчання завершується після закінчення декількох епох:

а) коли ітерації зійдуться, тобто вектор ваг перестає змінюватися,
б) коли повна підсумована по всіх векторах абсолютна помилка стане менше деякого малого значення.

Формула, що використовується на 3-му кроці, враховує наступні обставини:

а) модифікуються тільки компоненти матриці ваг, що відповідають ненульовим значенням входів;
б) знак приросту ваги відповідає знаку помилки, тобто позитивна помилка (δ > 0, значення виходу менше вимагається) проводить до посилення зв'язку;
в) навчання кожного нейрона відбувається незалежно від навчання решти нейронів, що відповідає важливому з біологічної точки зору, принципу *локальності навчання.*

Даний метод навчання називається „*методом корекції із зворотною передачею сигналу помилки*" або „*δ-правило*". Дельта-правило модифікує вагу відповідно з вимагаємим і дійсним значеннями виходу кожної полярності, як для безперервних, так і для бінарних входів і виходів. Представлений алгоритм відноситься до класу алгоритмів навчання з вчителем, оскільки відомі як вхідні вектори, так і необхідні значення вихідних векторів.

Труднощі з алгоритмом навчання персептрона:

- може виявитися скрутним визначити, чи виконана умова роздільності для конкретної навчальної множини,
- у багатьох ситуаціях, що зустрічаються на практиці, входи часто міняються в часі і можуть бути роздільні в один момент часу і нероздільні в іншій,
- у доказі алгоритму навчання персептрона нічого не говориться про те, скільки кроків потрібно для навчання мережі,
- не доведено, що персептронний алгоритм навчання більш швидкий в порівнянні з простим перебором всіх можливих значень ваг, і в деяких випадках цей підхід може виявитися кращим.

Можливості одношарового персептрона.

Доведення теореми навчання персептрона Розенблаттом показало, що персептрон здатний навчитися всьому, що він здатний уявити. Важливо при цьому уміти розрізняти уявність і навченість. Поняття уявності відноситься до здатності персептрона (або іншої мережі) моделювати певну функцію. Навченість вимагає наявності систематичної процедури підстроювання вагів мережі для реалізації цієї функції.

Кожний нейрон персептрона є формальним пороговим елементом, що приймає одиничні значення у випадку, якщо сумарний зважений вхід більше деякого порогового значення:

$$y_j = \begin{cases} 1, \sum_j W_{ij} x_j > \theta_j \\ 0, \sum_j W_{ij} x_j \le \theta_j \end{cases}.$$

Таким чином, при заданих значеннях ваг і порогів, нейрон має певне значення вихідної активності для кожного можливого вектора входів. Множина вхідних векторів, при яких нейрон активний (*y=1*), відокремлена від множини векторів, на яких нейрон пасивний (*y=0*) *гіперплощиною,* рівняння якої*:*

$$\sum_j W_{ij} x_j - \theta_j = 0.$$

Отже, нейрон здатний *відділити* (мати різний вихід) тільки такі дві множини векторів входів, для яких існує гіперплощина, що відсікає одну множину від іншої. Такі множини називають *лінійно роздільними*.

Проілюструємо це поняття на прикладі. Хай є нейрон, для якого вхідний вектор містить тільки дві булеві компоненти (*x1*, *x2*), що визначають площину. На даній площині можливі значення векторів відповідають вершинам одиничного квадрата. В кожній вершині визначено необхідне значення активності нейрона 0 (на рис. 1.4 – біла точка) або 1 (чорна точка). Вимагається визначити, чи існує такий набір вагів і порогів нейрона, при якому цей нейрон зможе відділити точки різного кольору?

На рис. 1.4 представлена одна з ситуацій, коли цього зробити не можна внаслідок лінійної нероздільності множини білих і чорних точок.

Необхідна активність нейрона для цього рисунка визначається таблицею істинності (наведена нижче), в якій неважко впізнати логічну функцію „що виключає АБО”:

X_1	X_2	Y
0	0	0
1	0	1
0	1	1
1	1	0

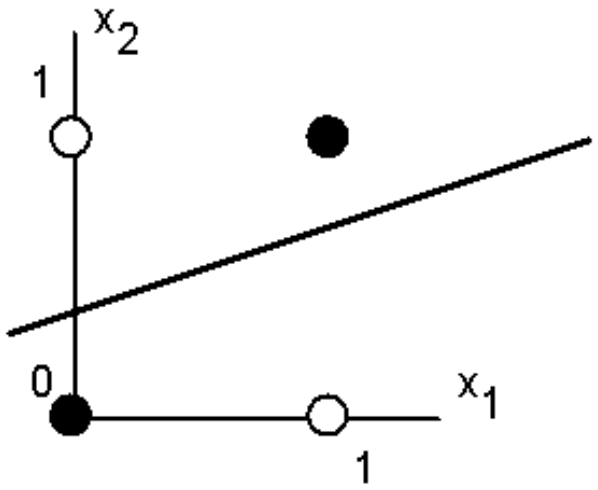


Рис. 1.4 – Білі точки не можуть бути відокремлені однією прямою від чорних (проблема „що виключає АБО” одношарового персептрона)

Лінійна нероздільність множини аргументів, які відповідають різним значенням функції, означає, що функція „що виключає АБО”, не може бути представлена формальним нейроном.

Розглянемо цю задачу з другого боку. Вважатимемо вихід мережі NET як поверхню над площиною x–y. Кожна точка цієї поверхні знаходиться над відповідною точкою площини x–y на відстані, рівному значенню NET в цій точці. Можна показати, що нахил цієї NET-поверхні однаковий для всієї поверхні x–y. Всі точки, в яких значення NET рівне величині порогу, проектуються на лінію рівня площини NET (рис. 1.5).

Ясно, що всі точки по одну сторону порогової прямої проектуються в значення NET, яке більше від порогу, а точки по іншу сторону дадуть менші значення NET. Таким чином, порогова пряма розбиває площину x–y на дві області. У всіх точках по одну сторону порогової прямої значення OUT дорівнює одиниці, по іншу сторону – нулю.

Отже, неможливо накреслити пряму лінію, що розділяє площину x–y так, щоб реалізовувалася функція „що виключає АБО”. Цей приклад не єдиний. Є великий клас функцій, що не реалізуються одношаровою мережею. Про ці функції говорять, що вони є лінійно нероздільними, і вони накладають певні обмеження на можливості одношарових мереж.

Лінійна роздільність обмежує одношарові мережі задачами

класифікації, в яких множина точок (відповідних вхідним значенням) може розділятись геометрично.

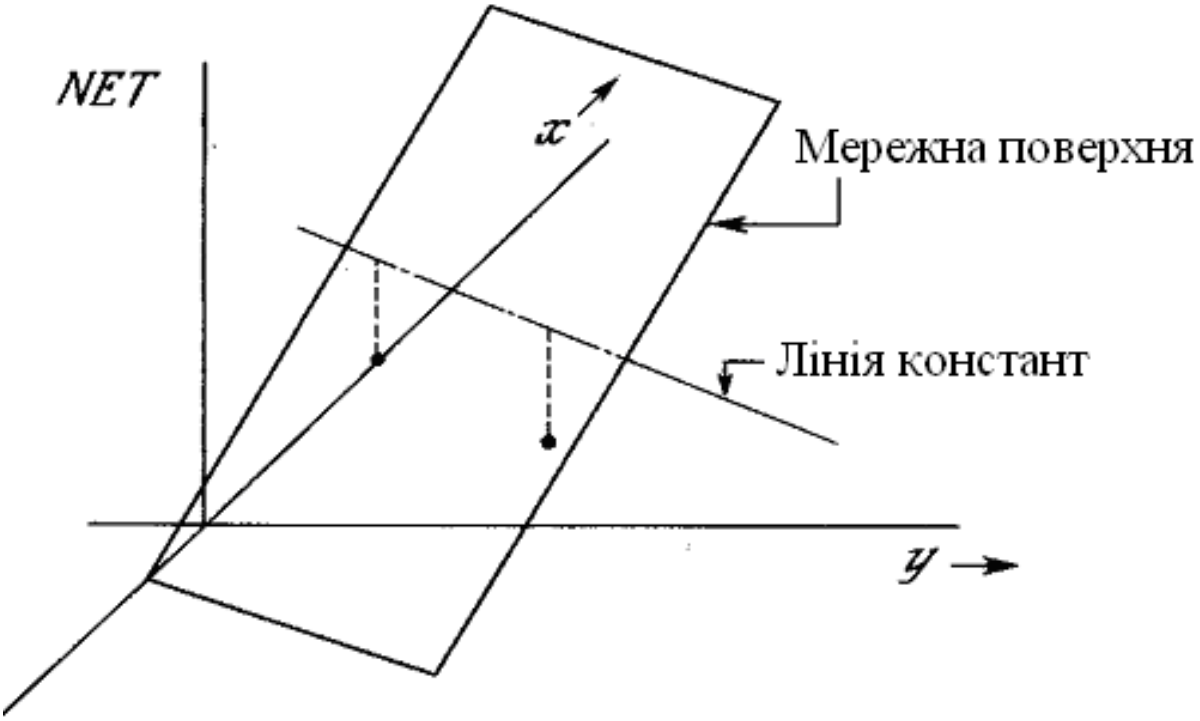


Рис. 1.5 – Персептронна NET-площина

Для випадку з двома входами роздільник є прямою лінією. У разі трьох входів розділення здійснюється площиною, що розтинає тривимірний простір. Для чотирьох або більш входів візуалізація неможлива, і необхідно уявити n-мірний простір, що розтинається „гіперплощиною” – геометричним об'єктом, який розтинає простір чотирьох або більшого числа вимірювань.

Оскільки лінійна роздільність обмежує можливості персептронного уявлення, то важливим є знання про те, що функція роздільна. На жаль, не існує простого способу визначити це, якщо число змінних велике.

Нейрон з n двійковими входами може мати 2^n різних вхідних образів, що складаються з нулів і одиниць. Оскільки кожний вхідний образ може відповідати двом різним бінарним виходам (одиниця і нуль), то всього є 2^{2^n} функцій від n змінних.

Як видно з таблиці 1.1, ймовірність того, що випадково вибрана функція виявиться лінійно роздільною, дуже мала навіть для помірного числа змінних. З цієї причини вживання одношарових персептронів на практиці обмежено простими задачами.

Таблиця 1.1 – Лінійно роздільні функції

n	2^{2^n}	Число лінійно роздільних функцій
1	4	4
2	16	14
3	256	104
4	65536	1882
5	$4{,}3\times10^{9}$	94572
6	$1{,}8\times10^{19}$	15 028 134

1.3 Багатошаровий персептрон

Багатошаровий персептрон – окремий випадок персептрона Розенблатта, в якому один алгоритм зворотного поширення помилки навчає всі шари. Назва з історичних причин не відображає особливості даного виду персептрона, тобто не зв'язана з тим, що в ньому є декілька шарів (оскільки декілька шарів було і у персептрона Розенблатта). *Особливістю багатошарового персептрона є наявність більш ніж одного навчаємого шару*. Необхідність у великій кількості навчаємих шарів відпадає, оскільки теоретично єдиного прихованого шару достатньо, щоб перекодувати вхідне уявлення так, щоб одержати лінійну роздільність для вихідного уявлення. Існує припущення, що, використовуючи більшу кількість шарів, можна зменшити число елементів в них, тобто сумарне число елементів в шарах буде менше ніж при використанні одного прихованого шару.

Всі види персептронів, запропоновані Розенблаттом починаючи з 1958 року, є за сучасною класифікацією багатошаровими. Проте в 1970-е роки інтерес до персептронів знизився, і в 1986 році Руммельхарт сконструював багатошаровий персептрон наново. Тому виникло уявлення про те, що первинний персептрон Розенблатта був примітивним і одношаровим, і лише Руммельхарт обґрунтував необхідність введення прихованих шарів.

У своїх дослідженнях Розенблатт використовував переважно елементарний персептрон з трьох шарів, причому вага першого шару вибиралася випадковим чином, а потім фіксувалася. Поєднання випадковості і великого числа нейронів в першому шарі забезпечували високу ймовірність попадання в такий гіперпростір, в якому була лінійна роздільність і гарантувалася збіжність процесу навчання, тобто Розенблатт зміг уникнути навчання першого шару, використовуючи його випадкову проекцію на багатовимірний простір.

Проте заслуга Руммельхарта, а одночасно з ним Вербоса П., Галушкіна А.І., Барцева С.І., Охоніна В.А. і ін. полягала в практичній модифікації багатошарової архітектури і розробці принципово нового

методу навчання – методу зворотного поширення помилки, але саме робота Руммельхарта відродила практичний інтерес до персептронів.

Відмінності багатошарового персептрона від персептрона Розенблатта. В 1988 році Мінський перевидав книгу „Персептрони”, в яку включив нові розділи. В них він показав, що якісних відмінностей при навчанні персептрона методом корекції помилки і навчанням багатошарового персептрона методом зворотного поширення помилки немає, обидва способи вирішують зіставні задачі і з тією ж ефективністю і обмеженнями. Різниця лише в способі досягнення мети.

Серед відмінностей багатошарового персептрона Румельхарта від персептрона Розенблатта можна виділити наступні:

- використання нелінійної функції активації, як правило, сигмоїдної,
- число навчаємих шарів більше одного. Частіше за все в додатках використовуються не більше трьох,
- сигнали, що надходять на вхід, і одержувані з виходу не бінарні, а можуть кодуватися десятковими числами, які потрібно нормалізувати, так щоб значення були на відрізку від 0 до 1 (нормалізація необхідна як мінімум для вихідних даних, відповідно до функції активації – сигмоїдної),
- допускається довільна архітектура зв'язків (у тому числі і повнозв'язні мережі),
- помилка мережі обчислюється не як число неправильних образів після ітерації навчання, а як деяка статистична міра нев'язності між потрібним і одержуваним значенням,
- навчання проводиться не до відсутності помилок після навчання, а до стабілізації вагових коефіцієнтів при навчанні або уривається раніше, щоб уникнути перенавчання.

Багатошаровий персептрон володіє функціональними перевагами в порівнянні з персептроном Розенблатта тільки в тому випадку, якщо у відповідь на дію не просто буде виконана якась реакція (оскільки вже в персептроні може бути одержана реакція будь-якого типу), а виразиться в підвищенні ефективності вироблення таких реакцій. Наприклад, покращає здібність до узагальнення, тобто до правильних реакцій на дії яким персептрон не навчався.

Архітектура багатошарового персептрона представлена на рис. 1.6.

Навчання багатошарового персептрона.

Типовим, найвідомішим і найпоширенішим представником алгоритмів навчання багатошарового персептрона є алгоритм зворотного поширення помилки – це детерміністський ітераційний градієнтний алгоритм навчання з вчителем.

Алгоритм зворотного поширення помилки використовується для навчання багатошарових нейронних мереж з послідовними зв'язками або,

як прийнято говорити, мереж прямого поширення типу багатошарового персептрона (рис. 1.6).

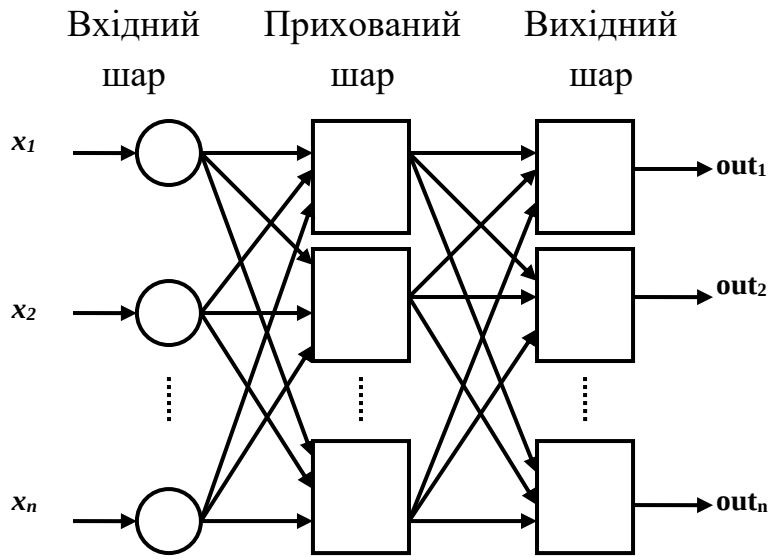


Рис. 1.6 – Архітектура багатошарового (двошарового) персептрона

У багатошарових мережах оптимальні вихідні значення нейронів всіх шарів, окрім останнього шару, як правило, невідомі. Тому персептрон з трьома і більше шарами вже неможливо навчити, керуючись тільки величинами помилок на виходах штучної нейронної мережі (ШНМ). Найприйнятнішим варіантом навчання в таких умовах виявляється градієнтний метод пошуку мінімуму функції помилки з розглядом сигналів помилки від виходів ШНМ до її входів, тобто в напрямі, зворотному прямому поширенню сигналів в звичному режимі роботи. Цей алгоритм навчання ШНМ одержав назву *процедури зворотного поширення помилки*.

Основна *ідея зворотного поширення* полягає в тому, як одержати оцінку помилки для нейронів прихованих шарів. Помітимо, що *відомі* помилки, що робляться нейронами вихідного шару, виникають внаслідок *невідомих* поки помилок нейронів прихованих шарів. Чим більше значення синаптичного зв'язку між нейроном прихованого шару і вихідним нейроном, тим сильніше помилка першого впливає на помилку другого. Отже, оцінку помилки елементів прихованих шарів можна одержати, як зважену суму помилок подальших шарів. При навчанні інформація поширюється від низьких шарів ієрархії до вищих шарів, а оцінки помилок, що робляться мережею – у зворотному напрямі, що і відображено в назві методу.

Розглянемо цей алгоритм.

Для спрощення розгляду обмежимося випадком, коли мережа має тільки один прихований шар (рис. 1.7).

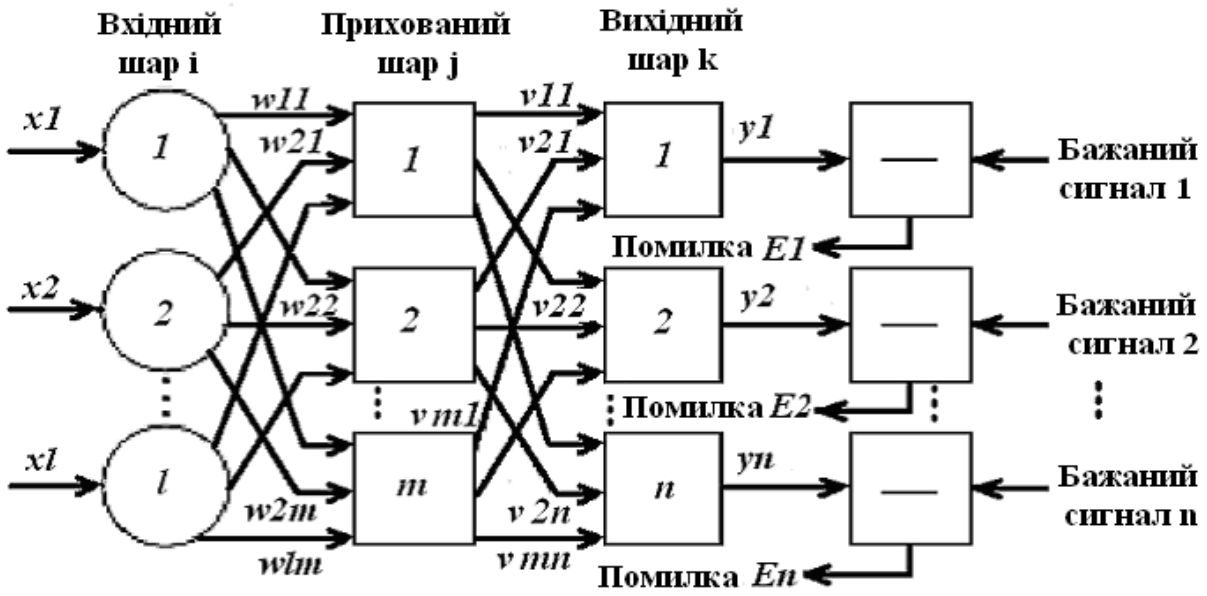


Рис. 1.7 – Процедура навчання багатошарового персептрона

Матрицю вагових коефіцієнтів від входів до прихованого шару позначимо *W*, а матрицю ваг, що сполучають прихований і вихідний шар – *V*. Для індексів візьмемо такі позначення: входи нумеруватимемо тільки індексом *i* ($\overline{1...l}$), елементи прихованого шару – індексом *j* ($\overline{1...m}$), а виходи, відповідно, індексом *k* ($\overline{1...n}$).

Хай мережа навчається на вибірці (X^r,Y^r), r = 1...N. Активності нейронів позначатимемо малими буквами y з відповідним індексом, а сумарні зважені входи нейронів – малими буквами x.

Структура алгоритму:

– Крок 0. Початкові значення ваг всіх нейронів всіх шарів $V(t=0)$ і $W(t=0)$ вважаються випадковими числами;

– Крок 1. Мережі пред'являється вхідний образ X^{α}, в результаті формується вихідний образ $y \neq Y^{\alpha}$. При цьому нейрони послідовно від шару до шару функціонують по наступних формулах:

$$\text{прихований шар} \quad x_j = \sum_i W_{ij} x_i^{\alpha};\ \ y_j = f\left(x_j\right),$$

$$\text{вихідний шар} \quad x_k = \sum_j V_{jk} y_j;\ \ y_k = f\left(x_k\right),$$

де $f(x)$ – сигмоїдна функція;

– Крок 2. Хай функціонал квадратичної помилки мережі для даного вхідного образу має вигляд:

$$E = \sum_k \left(y_k - Y_k^{\alpha} \right)^2 .$$

Даний функціонал підлягає мінімізації. Класичний градієнтний метод оптимізації полягає в ітераційному уточненні аргументу згідно з формулою:

$$V_{jk}(t+1) = V_{jk}(t) - h \cdot \frac{\partial E}{\partial V_{jk}} .$$

Функція помилки в явному вигляді не містить залежності від ваги V_{jk}, тому скористаємося формулами неявного диференціювання складної функції:

$$\frac{\partial E}{\partial y_k} = \delta_k = \left(y_k - Y_k^{\alpha} \right),$$

$$\frac{\partial E}{\partial x_k} = \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial x_k} = \delta_k \cdot y_k (1 - y_k),$$

$$\frac{\partial E}{\partial V_{jk}} = \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial x_k} \cdot \frac{\partial x_k}{\partial V_{jk}} = \delta_k \cdot y_k (1 - y_k) \cdot y_j .$$

Тут врахована корисна властивість сигмоїдної функції *f(x):* її похідна виражається тільки через саме значення функції, $f'(x)=f(1–f)$. Таким чином, всі необхідні величини для підстроювання ваг вихідного шару V одержані (рис. 1.8).

– Крок 3. На цьому кроці виконується підстроювання ваг прихованого шару (рис. 1.9). Градієнтний метод дає:

$$W_{ij}(t+1) = W_{ij}(t) - h \cdot \frac{\partial E}{\partial W_{ij}} .$$

Обчислення похідних виконуються по тих же формулах, за винятком деякого ускладнення формули для помилки δ_j.

$$\frac{\partial E}{\partial x_k} = \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial x_k} = \delta_k \cdot y_k (1 - y_k),$$

$$\frac{\partial E}{\partial y_j} = \delta_j = \sum_k \frac{\partial E}{\partial x_k} \cdot \frac{\partial x_k}{\partial y_j} = \sum_k \delta_k \cdot y_k (1 - y_k) \cdot V_{jk},$$

$$\frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_j} \cdot \frac{\partial x_j}{\partial W_{ij}} = \delta_j \cdot y_j (1 - y_j) \cdot X_i^{\alpha} =$$

$$= \left[\sum_k \delta_k \cdot y_k (1 - y_k) \cdot V_{jk} \right] \cdot \left[y_j (1 - y_j) \cdot X_i^{\alpha} \right]$$

При обчисленні δ_j тут і був застосований принцип зворотного поширення помилки: часткові похідні беруться тільки по змінних *подальшого* шару. За одержаними формулами модифікується вага нейронів прихованого шару. Якщо в нейронній мережі є декілька прихованих шарів, процедура зворотного поширення застосовується послідовно для кожного з них, починаючи з шару, передуючого вихідному, і далі до шару, наступного за вхідним шаром. При цьому формули зберігають свій вигляд із заміною елементів вихідного шару на елементи відповідного прихованого шару.

– Крок 4. Кроки 1–3 повторюються для всіх навчальних векторів. Навчання завершується після досягнення малої повної помилки або максимально допустимого числа ітерацій.

Як видно з опису кроків 2–3, навчання зводиться до рішення задачі оптимізації функціонала помилки градієнтним методом. Сутність зворотного поширення помилки полягає в тому, що для її оцінки для нейронів прихованих шарів можна прийняти зважену суму помилок подальшого шару.

Параметр *h* має сенс темпу навчання і вибирається достатньо малим для забезпечення збіжності методу. Необхідно відзначити, що:

– по-перше, збіжність методу зворотного поширення дуже повільна. Невисокий темп збіжності є особливістю всіх градієнтних методів, оскільки локальний напрям градієнта зовсім не співпадає з напрямом до мінімуму,

– по-друге, підстроювання ваг виконується незалежно для кожної пари образів навчальної вибірки. При цьому покращення функціонування на деякій заданій парі може, взагалі кажучи, приводити до погіршення роботи на попередніх образах. В цьому значенні, немає достовірних (окрім практики вживання методу) гарантій збіжності.

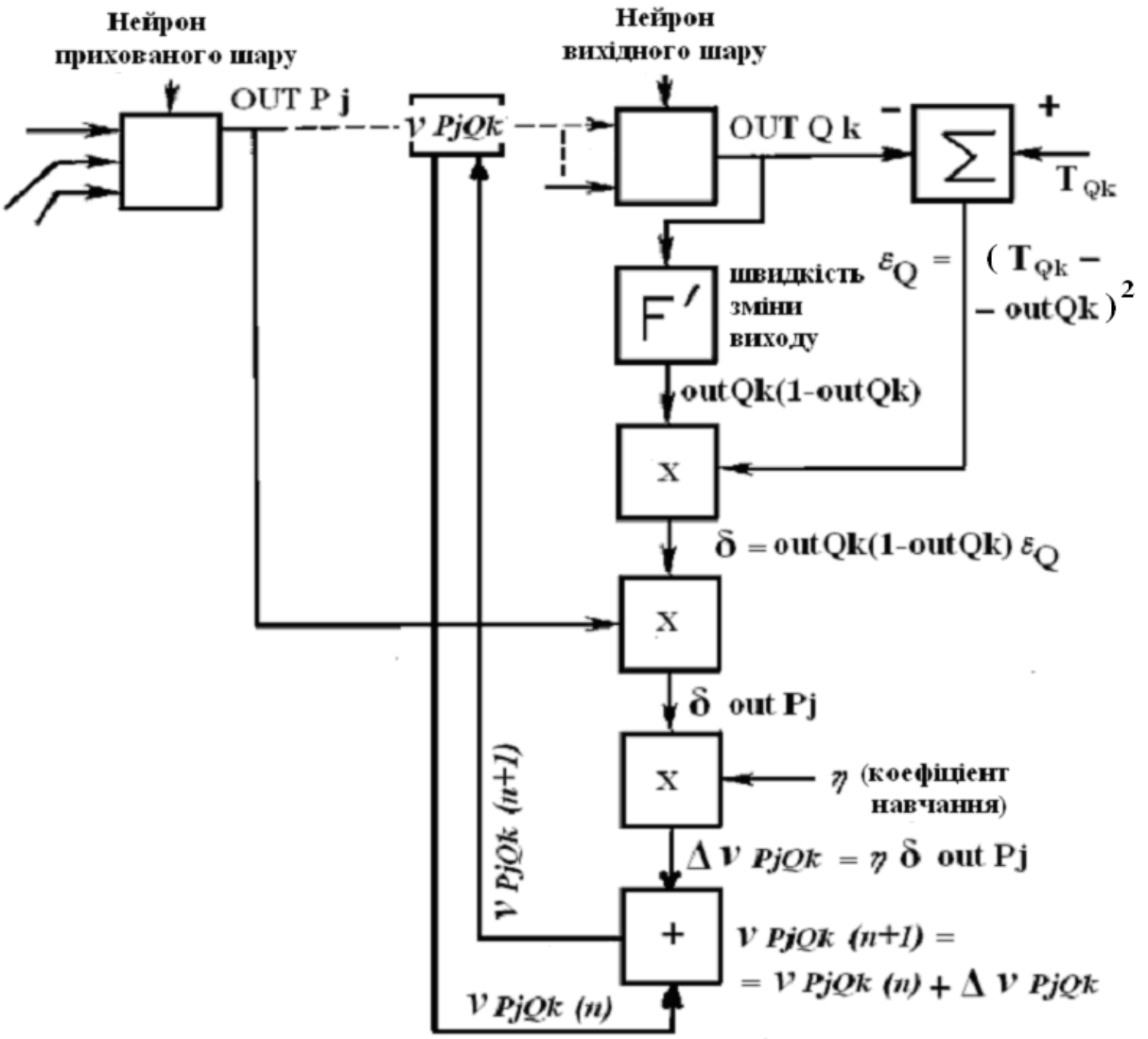


Рис. 1.8 – Підстроювання ваг вихідного шару

Дослідження показують, що для представлення довільного функціонального відображення, яке задається навчальною вибіркою, достатні всього *два шари* нейронів. Проте на практиці, у разі складних функцій, використовування більш ніж одного прихованого шару може давати економію повного числа нейронів.

Пороги нейронів. Легко помітити, що поріг нейрона може бути зроблений еквівалентним додатковій вазі, сполученій з фіктивним входом, рівним –1. Дійсно, вибираючи $W_0=\theta$, $x_0= -1$ і починаючи підсумовування з нуля, можна розглядати нейрон з нульовим порогом і одним додатковим входом:

$$y = f\left(\sum_{i=1}^{n} W_i x_i - \theta\right) = f\left(\sum_{i=1}^{n} W_i x_i + (-1)\cdot(\theta)\right) =$$

$$= f\left(\sum_{i=1}^{n} W_i x_i + W_0 x_0\right) = f\left(\sum_{i=0}^{n} W_i x_i\right) \quad .$$

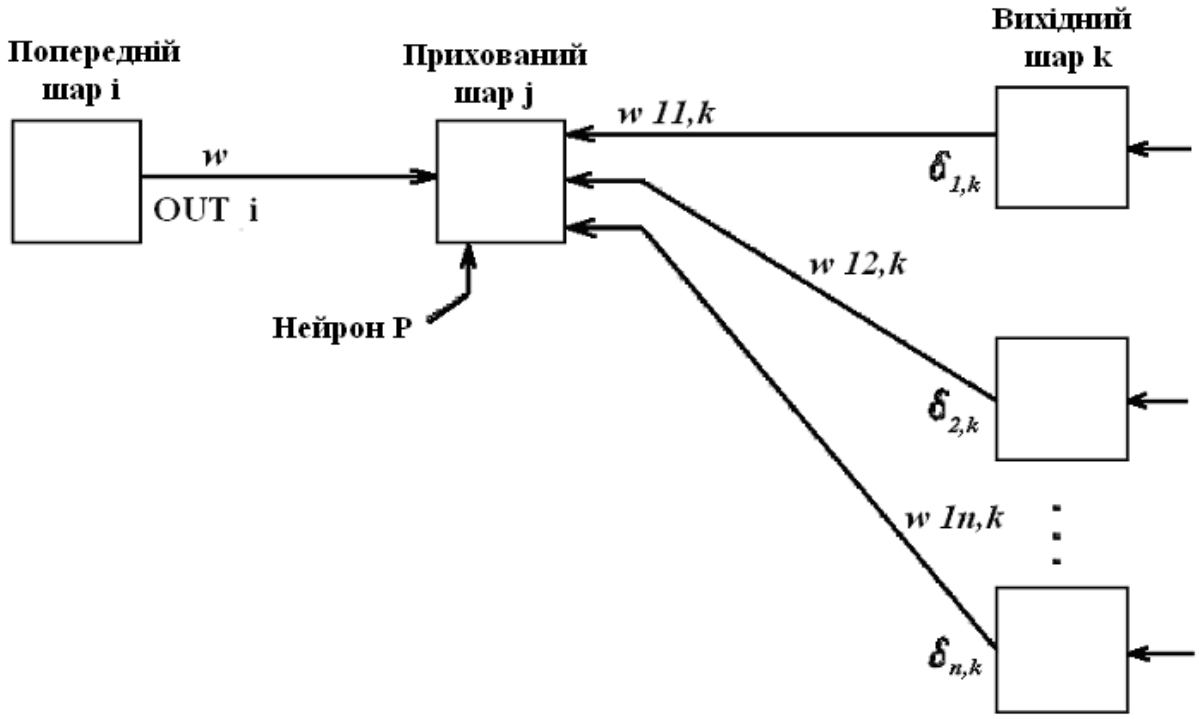


Рис. 1.9 – Підстроювання ваг прихованого шару

Можна на додаткові входи нейронів подавати відповідні пороги (зсуви) і тоді всі викладені в алгоритмі зворотного поширення формули підсумовування по входах починаються з нульового індексу.

Можливості багатошарового персептрона.

До кінця 60-х років проблема лінійної роздільності була добре зрозуміла. До того ж було відомо, що це серйозне обмеження уявності одношаровими мережами можна подолати, додавши додаткові шари. Наприклад, двошарові мережі можна одержати каскадним з'єднанням двох одношарових мереж. Вони здатні виконувати більш загальні класифікації, відділяючи ті точки, які містяться в опуклих обмежених або необмежених областях. Область називається опуклою, якщо для будь-яких двох її точок відрізок, що їх сполучає цілком лежить в області. Область називається обмеженою, якщо її можна укласти в деякий круг. Необмежену область неможливо укласти всередину круга (наприклад, область між двома паралельними лініями). Приклади опуклих обмежених і необмежених

областей представлені на рис. 1.10.

Щоб уточнити вимогу опуклості, розглянемо просту двошарову мережу з двома входами, підведеними до двох нейронів першого шару, сполученими з єдиним нейроном в шарі 2 (рис. 1.11).

Хай поріг вихідного нейрона рівний 0,75, а обидві його ваги дорівнюють 0,5. В цьому випадку для того, щоб поріг був перевищений, і на виході з'явилася одиниця, вимагається, щоб обидва нейрони першого рівня на виході мали одиницю. Таким чином, вихідний нейрон реалізує логічну функцію І.

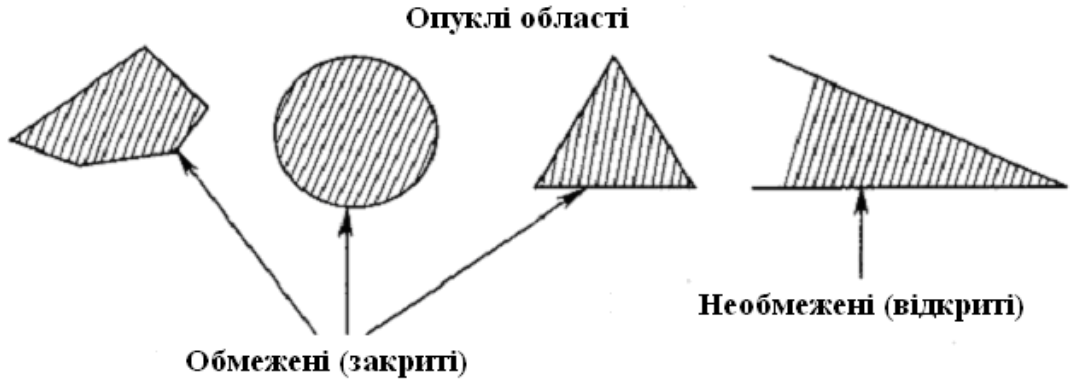


Рис. 1.10 – Опуклі обмежені і необмежені області

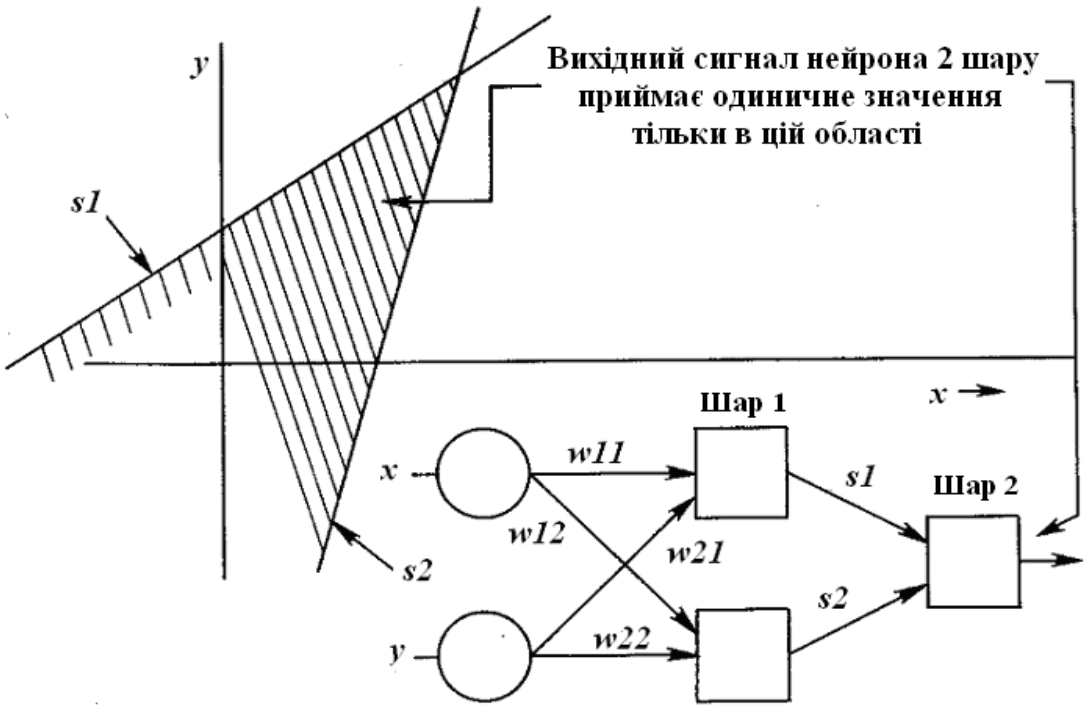


Рис. 1.11 – Опукла область рішень, що задається двошаровою мережею

На рис. 1.11 кожен нейрон шару 1 розбиває площину x–y на дві напівплощини, один забезпечує одиничний вихід для входів нижче верхньої лінії, інший – для входів вище за нижню лінію. Тут же показаний результат такого подвійного розбиття, де вихідний сигнал нейрона другого шару рівний одиниці тільки усередині V-подібної області. Аналогічно в другому шарі може бути використано три нейрони з подальшим розбиттям площини і створенням області трикутної форми. Включенням достатнього числа нейронів у вхідний шар можна утворити опуклий багатокутник будь-якої бажаної форми. Оскільки вони утворені за допомогою операції І над областями, лініями, що задаються, то всі такі багатогранники опуклі, отже, тільки опуклі області і виникають. Точки, що не становлять опуклої області, не можуть бути відокремлені від інших точок площини двошаровою мережею.

Нейрон другого шару не обмежений функцією І. Він може реалізовувати багато інших функції при відповідному виборі вагів і порогу. Наприклад, можна зробити так, щоб одиничний вихід будь-якого з нейронів першого шару приводив до появи одиниці на виході нейрона другого шару, реалізувавши тим самим логічне АБО. Є 16 двійкових функцій від двох змінних. Якщо вибирати відповідним чином ваги і поріг, то можна відтворити 14 з них (всі, окрім „що виключає АБО” і „що виключає НІ”).

Входи не обов'язково повинні бути двійковими. Вектор безперервних входів може бути довільною точкою на площині x–y. В цьому випадку виявляється здатність мережі розбивати площину на безперервні області, а не з розділенням дискретної безлічі точок. Для всіх цих функцій, проте, лінійна роздільність показує, що вихід нейрона другого шару дорівнює одиниці тільки в частині площини x–y, обмеженою багатокутною областю. Тому для розділення площин P і Q необхідне, щоб всі P лежали всередині опуклої багатокутної області, що не містить точок Q (або навпаки).

Тришарова мережа, проте, є більш загальною. Її класифікувальні можливості обмежені лише числом штучних нейронів і ваг. Обмеження на опуклість відсутні. Тепер нейрон третього шару приймає як вхід набір опуклих багатокутників, і їх логічна комбінація може бути неопуклою.

На рис. 1.12 ілюструється випадок, коли два трикутники A і B, скомбіновані за допомогою функцій „A і НЕ B”, задають неопуклу область. При додаванні нейронів і ваг число сторін багатокутників може необмежено зростати. Це дозволяє апроксимувати область будь-якої форми з будь-якою точністю. Додатково не всі вихідні області другого шару повинні перетинатися. Можливо об'єднувати різні області, опуклі і неопуклі, видаючи на виході одиницю, всякий раз, коли вхідний вектор належить одній з них.

Не дивлячись на те, що можливості багатошарових мереж були відомі давно, протягом багатьох років не було теоретично обґрунтованого

алгоритму для настроювання їх ваг. В подальші роки були розроблені багатошарові навчальні алгоритми.

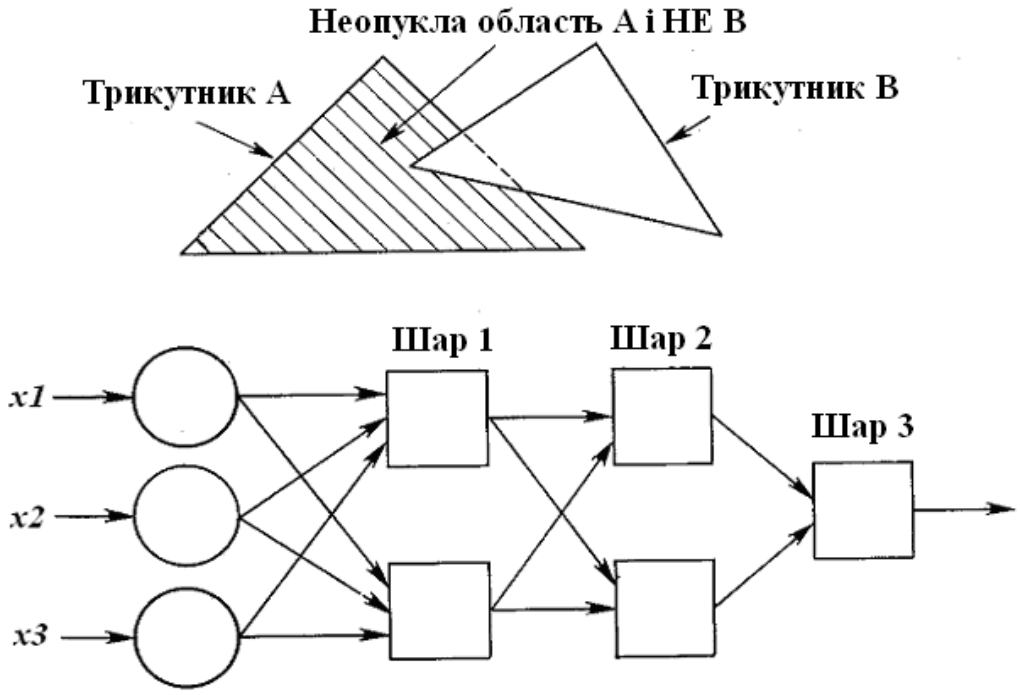


Рис. 1.12 – Увігнута область рішень, що задається тришаровою мережею

2 ПРАКТИЧНА ЧАСТИНА ЛАБОРАТОРНОЇ РОБОТИ

„Дослідження персептрона”

2.1 Мета лабораторної роботи

Вивчення архітектури персептронів і функцій для їх створення; вивчення алгоритмів навчання персептронів; вивчення можливостей персептронів. Придбання практичних навиків побудови і навчання персептронів для вирішення задач обробки даних.

2.2 Технічне забезпечення лабораторної роботи

1) персональний комп'ютер,
2) програмне забезпечення – система комп'ютерної математики.

2.3 Порядок виконання лабораторної роботи

Час, що відводиться на проведення лабораторної роботи в комп'ютерному класі – 6 годин, на самостійну роботу з підготовки до проведення лабораторної роботи і до захисту звіту – 6 годин.

Напередодні лабораторної роботи:
1) вивчити завдання і порядок виконання лабораторної роботи,
2) вивчити теоретичні відомості до лабораторної роботи.

Під час лабораторної роботи:
1) одержати допуск до проведення лабораторної роботи, відповівши на питання викладача з теоретичної частини досліджень при проведенні лабораторної роботи,
2) написати програмний код по кожному пункту завдання і проаналізувати результати, одержані при виконанні програмних кодів,
3) зробити висновки по лабораторній роботі.

Після лабораторної роботи:
1) підготувати звіт про лабораторну роботу відповідно до наведених в цьому методичному посібнику вимог,
2) захистити звіт перед викладачем, відповівши на його запитання щодо практичного проведення досліджень при виконанні лабораторної роботи.

2.4 Функції, які застосовуються в лабораторній роботі і їх опис

Персептрон – це одношарова нейронна мережа з **S** нейронами і **R** входами, кожний з яких може складатися з декількох елементів. Передавальною функцією кожного нейрона є східчаста функція типу **hardlim** або **hardlims**. Крім основних входів, нейрони персептрона мають вхід для постійного зсуву, рівного одиниці. Елементи входів і зсуву зважуються за допомогою функції скалярного добутку **dotprod** і підсумовуються за допомогою функції накопичення **netsum.**

Створення персептрона проводиться наступною функцією:

net = newp(PR, S, tf, lf)

де **net** – об'єкт класу **network; PR** – масив розміру **R**x**2** мінімальних і максимальних значень для **R** векторів входу**; S** – число нейронів персептрона**; tf** – передавальна функція із списку **{hardlim, hardlims},** причому за умовчанням задається **hardlim; lf** – навчальна функція із списку **{learnp, learnpn**}, причому за умовчанням – **learnp.**

При створенні персептрона, матриця ваг і вектор зсувів ініціалізуються нулями за допомогою функцій **initzero**.

Навчання персептрона проводиться за допомогою функції адаптації **adapt**, яка коректує вагу і зсуви за наслідками обробки кожної пари вхідних і вихідних значень (навчання з вчителем). Вживання функції **adapt** гарантує, що будь-яка задача класифікації з лінійно роздільними векторами буде вирішена за кінцеве число циклів настроювання.

Із застосуванням функції навчання **train** настройка параметрів мережі виконується не після кожного проходу, а в результаті всіх проходів навчальної множини. Проте у ряді випадків її вживання не забезпечує збіжності процесу настроювання, тому функція навчання **train** рідше використовується для навчання персептрона.

Настройка ваг і зсувів, реалізується функціями **learnp** і **learnpn**, проводиться за наступними правилами:

а) для вхідних сигналів обчислюються вихідні;
б) визначаються помилки як різниця між цільовим виходом і відповідним вихідним сигналом;
в) проводиться коректування ваг і зсувів шляхом складання старих значень з приростами, кожне з яких дорівнює добутку відповідного сигналу на помилку того нейрона, для якого коректується параметр.

Для того, щоб зробити час навчання нечутливим до великих або малих викидів векторів входу, проводять нормування вхідних даних при обчисленні приростів ваг і зсувів:

$$pn_i = p_i\ (sqrt(1+ p_1^2 +\ldots+ p_r^2)).$$

Таке нормування забезпечується вживанням функції **learnp** для настроювання, як ваг, так і зсувів. Також автоматично властивості **net.adaptFcn** задається значення **adaptwb**, що дозволяє використовувати будь-які функції для настройки ваг і зсувів, а властивості **net.adaptParam** – набір параметрів за умовчанням.

Адаптація персептрона проводиться функцією-методом **adapt(net,P,T)**, де **P** – вхідні вектори; **T** – цільові значення. Процес адаптації продовжується до тих пір, поки не буде досягнуте необхідне значення критерію якості навчання у вигляді середньої абсолютної помилки, обчислюваною функцією **mae**.

2.5 Завдання на лабораторну роботу

Завдання 1

Виконати моделювання одношарового персептрона і дельта–правила його навчання для класифікації двох не перетинних класів об'єктів.

Навчальні вектори зведені в таблицю 2.1.

Таблиця 2.1 – Таблиця еталонів (навчальна множина)

Класи	Еталони					
		1	2	3	4	5
0	X1	0,10	0,25	0,18	0,23	0,30
	Y1	0,20	0,80	0,45	0,70	1,40
1	X2	0,07	0,06	0,02	0,06	0,13
	Y2	0,60	0,68	0,40	0,40	0,90

Вектори для класифікації зведені в таблицю 2.2.

Таблиця 2.2 – Таблиця об'єктів

	Об'єкти	
	1	2
X1	0,11	0,10
Y1	0,40	0,55

Розробка алгоритму навчання персептрона відповідно до правила Розенблатта.

Підстроювання параметрів (навчання) персептрона здійснюється з використанням навчальної множини.

Позначимо **p** – вектор входів персептрона, **t** – цільовий вектор, **a** – вектор виходу нейрона. Мета навчання – зменшити помилку **e = t–a** між

вектором цілі і реакцією нейрона.

Навчання персептрона (крок підстроювання ваги) повинне залежати від величини помилки **e**. Вектор цілі може мати значення тільки **0** і **1**, оскільки персептрон з функцією активації **hardlim** може видавати на виході тільки такі значення.

Розрахуємо зміну вектора ваг Δw з похибкою **e = t–a**:

$$\Delta w = (t-a)p = ep.$$

Аналогічно одержимо вираз для зміни зсуву з урахуванням того, що зсув можна розглядати як вагу для одиничного входу:

$$\Delta b = (t-a)1 = e.$$

У разі декількох нейронів ці співвідношення можна представити у векторній формі:

$$\Delta W = (t-a)pT = epT$$
$$\Delta b = (t-a)1 = e.$$

Остаточно правило навчання (підстроювання вагів і зсувів) персептрона можна записати в наступному вигляді:

$$W^{нов} = W^{стар} + ep^T$$
$$b^{нов} = b^{стар} + e.$$

Реалізуємо алгоритм програмно:

```
P=[0.10 0.25 0.18 0.23 0.30 0.07 0.06 0.02 0.06 0.13;      % – навчальна
    0.20 0.80 0.45 0.70 1.40 0.60 0.68 0.40 0.40 0.90]     % – множина;
T=[0 0 0 0 0 1 1 1 1 1]       % – цільова функція;
len=length(P);                % – кількість навчальних векторів;
W=[0 0];                      % – початкові значення ваг;
b=0;                          % – початкове значення зсуву;
c=1;                          % – логічна умова;
n=0;                          % – лічильник епох;
while c==1                    % – цикл з невизначеним числом кроків;
   n=n+1;
   N=W*P+b;                   % – вихід суматора нейрона;
   a=hardlim(N);              % – вихід нейрона;
   e=T-a;                     % – вектор помилок за одну епоху;
   E=0;                       % – сумарна помилка за епоху;
   for k=1:len
      E=E+e(k);               % – обчислення сумарної помилки за епоху;
   end;
```

```
   flag=0;       % – прапорець ненульового значення у векторі помилок;
   for m=1:len
      if e(m)~=0       % – перевірка ненульового значення помилки;
         flag=1;
      end;
   end;
   if flag==1
      W=W+e*P';       % – навчання (підстроювання ваг) нейрона;
      b=b+E;           % – навчання (підстроювання зсуву) нейрона;
   else
      c=0;
   end;
end;
a
e
b
W
n
```

У результаті персептрон побудований і навчений.

Проведемо класифікацію 1-го об'єкта:

```
U=[0.11; 0.40];
NU=W*U+b;
aU=hardlim(NU)
```

Проведемо класифікацію 2-го об'єкта:

```
U1=[0.10; 0.55];
NU1=W*U1+b;
aU1=hardlim(NU1)
```

Відобразимо на графіку навчальний масив і об'єкти, що підлягають класифікації:

```
figure;
plot(P(1,1:5),P(2,1:5),'vk',P(1,6:10),P(2,6:10),'+k',U(1,:),U(2,:),'ok',U1(1,:),
U1(2,:),'ok');
```

Побудуємо на графіку лінію, що розділяє два класи об'єктів. В нашому двовимірному випадку лінія має вигляд:

$$W(1)\ P(1)+W(2)P(2)+b=0.$$

де W(1) – вага 1-го входу нейрона

W(2) – вага 2-го входу нейрона

P(1) – масив координат абсцис еталонів і об'єктів

P(2) – масив координат ординат еталонів і об'єктів

b – зсув нейрона.

Перетворивши вираз відповідно до згаданих особливостей, одержимо:

$$W(1)\,x+W(2)\,y+b=0.$$

Вирішимо його відносно y:

$$y=[-W(1)\,/W(2)]\cdot x - b/W(2).$$

Підставимо назад замість x масив P(1) і одержимо остаточно:

$$y=[-W(1)\,/W(2)]\cdot P(1) - b/W(2).$$

```
y=(-W(1,1) /W(1,2))*P(1,:)-b/W(1,2);
hold on;
plot(P(1,:),y);
```

Висновки за завданням 1.

Завдання 2

Створити персептрон з одним нейроном і одноелементним входом, діапазон значень якого від **0** до **1**, і проаналізувати значення параметрів його обчислювальної моделі, виконавши наступні дії по створенню і ініціалізації персептрона:

```
netP = newp([0 1], 1)              % – об'єкт – персептрон;
inf1 = netP.inputWeights{1,1} % – характеристика об'єкта;
inf2 = netP.biases{1}              % – характеристика об'єкта;
inf3 = netP.IW{1,1}                % – значення ваги;
inf4 = netP.b{1}                   % – значення зсуву;
netP.IW{1,1}=[10];                 % – завдання ваги;
netP.b{1}=[-7];                    % – завдання зсуву;
inf5 = netP.IW{1,1}                % – нове значення ваги;
inf6 = netP.b{1}                   % – нове значення зсуву;
netP = init(netP);                 % – ініціалізація нулями;
inf7 = netP.IW{1,1}                % – нове нульове значення ваги;
inf8 = netP.b{1}                   % – нове нульове значення зсуву;
netP.inputWeights{1,1}.initFcn='rands';
netP.biases{1}.initFcn='rands';
netP = init(netP);             % – ініціалізація випадковими значеннями;
inf9 = netP.IW{1,1}        % – нове випадкове  значення ваги;
inf10 = netP.b{1}          % – нове випадкове значення зсуву;
p = {[2] [4] [1.5]};           % – послідовність входів;
a = sim(netP, p)               % – моделювання мережі;
```

Висновки за завданням 2.

Завдання 3

На навчальній множині і цільовій функції Завдання 1 побудувати, навчити і перевірити роботу персептрона з використанням вбудованих функцій СКМ і провести класифікацію двох не перетинних класів об'єктів:

```
P=[0.10 0.25 0.18 0.23 0.30 0.07 0.06 0.02 0.06 0.13;      % – навчальна
    0.20 0.80 0.45 0.70 1.40 0.60 0.68 0.40 0.40 0.90];     % – множина;
T=[0 0 0 0 0 1 1 1 1 1]                                    % – цільова функція;
netP=newp([0.02 1.40;0 1],1);          % – об'єкт – персептрон;
netP.trainParam.epochs=100;        % – завдання параметрів навчання;
netP.trainParam.goal=0.01;
netP.trainParam.show=5;
netP = train(netP,P,T);                % – навчання персептрона;
inf1=netP.IW{1,1}                    % – виведення результатів навчання;
inf2=netP.b{1}
U=[0.11; 0.40];                        % – об'єкти для класифікації;
U1=[0.10; 0.55];
UU=[0.11 0.10; 0.40 0.55];
Y=sim(netP,UU)                         % – робота персептрона;
figure;                % – відображення результатів роботи персептрона;
plot(P(1,1:5),P(2,1:5),'vk',P(1,6:10),P(2,6:10),'+k',UU(1,:),UU(2,:),'ok');
```

Розрахуємо коефіцієнти і побудуємо лінію, що розділяє множини:

```
k1=netP.IW{1,1}
k2=netP.b{1}
y=(-k1(1,1) /k1(1,2))*P(1,:)-k2/k1(1,2);
hold on;
plot(P(1,:),y);
```

Висновки за завданням 3.

Завдання 4

На навчальній множині і цільовій функції Завдання 3 побудувати, навчити і перевірити роботу персептрона з використанням графічного інтерфейсу користувача (GUI) СКМ і провести класифікацію двох не перетинних класів об'єктів з характеристиками, вказаними в Завданні 3.

Висновки за завданням 4.

Завдання 5

Створити персептрон з одним нейроном і двійковим входом і навчити його спочатку для виконання логічної функції **І** на основі алгоритму адаптації, а потім для виконання логічної функції **АБО** на основі алгоритму навчання, виконавши наступні дії:

```
netP = newp([0 1;0 1], 1);          % – об'єкт – персептрон;
p={[0;0] [0;1] [1;0] [1;1]};        % – підготовка навчальних
p1=cat(2, p{:});                    % – послідовностей;
T1=num2cell(p1(1,:)& p1(2,:))       % – функція І;
T2=num2cell(p1(1,:)| p1(2,:))       % – функція АБО.
netP.adaptParam.passes = 20;        % – число проходів адаптації;
[netP,Y,E,Pf]=adapt(netP,p,T1);     % – настроювання на І;
inf1=netP.IW{1,1}                   % – дані про ваги і зсув;
inf2=netP.b{1}
Y =sim(netP,p)                      % – моделювання І.
netP = init(netP);                  % – ініціалізація;
netP = train(netP,p,T2);            % – настроювання на АБО;
netP.trainParam.epochs=20;          % – кількість епох навчання;
inf3=netP.IW{1,1}                   % – дані про ваги і зсув;
inf4=netP.b{1}
Y = sim(netP,p)                     % – моделювання АБО.
```

Висновки за завданням 5.

Завдання 6

Виконати моделювання двошарового з двома входами і одним виходом персептрона і алгоритму його навчання на основі зворотного поширення помилки.

Як приклад взяти персептрон на рис. 1.7.

Процес його навчання відображений на рис. 1.8 і 1.9.

Модель проходу вперед виглядає наступним чином.

Вихід суматора нейрона виразиться співвідношенням:

$$NET_1 = X \cdot W = (x_1, x_2, \ldots, x_l) \cdot \begin{pmatrix} w_{11} w_{12} \ldots w_{1m} \\ w_{21} w_{22} \ldots w_{2m} \\ \ldots\ldots\ldots\ldots \\ w_{l1} w_{l2} \ldots w_{lm} \end{pmatrix} =$$

$$= (x_1 w_{11} + x_2 w_{21} + \ldots + x_l w_{l1}, x_1 w_{12} + x_2 w_{22} + \ldots + x_l w_{l2}, \ldots, x_1 w_{1m} + x_2 w_{2m} + \ldots + x_l w_{lm})$$

Вихід першого прихованого шару виразиться співвідношенням:

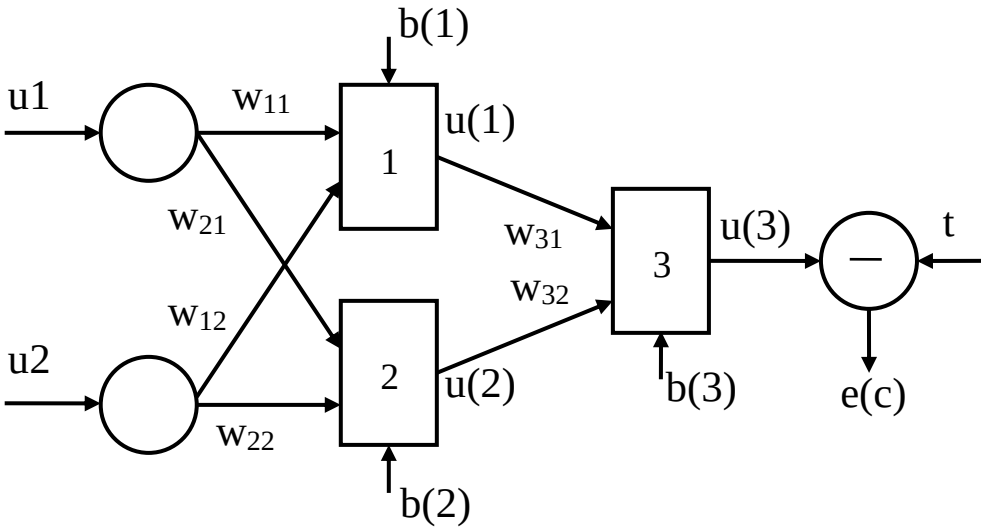
$$OUT_1 = F(NET_1) = F(X \cdot W).$$

Вихід вихідного шару аналогічно виразиться співвідношенням:

$$OUT_2 = F(OUT_1 \cdot V) = F(F(X \cdot W) \cdot V).$$

Зворотний прохід для підстроювання ваг вихідного і прихованого шару детально відображений на рис. 1.8 і 1.9.

Для кращого розуміння програмного коду є сенс розглянути структурну схему модельованого персептрона (рис. 2.1).



$u(1)=F(w_{11}\cdot u1+w_{12}\cdot u2+b1)$
$u(2)=F(w_{21}\cdot u1+w_{22}\cdot u2+b2)$
$u(3)=F(w_{31}\cdot u(1)+w_{32}\cdot u(2)+b3)$
$e(c)=(t-u(3))^2$

Рис. 2.1 – Структурна схема модельованого персептрона

При моделюванні виконати наступні дії:

```
clc;
clear all;
close all;
tic

lr=0.5; %  0.05  0.1  0.15  0.2  1  1.5  2  5  – коефіцієнт навчання
t=1;                    % – цільовий вектор
e=1;                    % – початкова помилка
c=0;                    % – лічильник епох
u1=1; u2=0; %  1        – початкові входи мережі
u=[0 0 0]; %  1 1 1     – початкові виходи нейронів мережі
b=[1 1 1];              % – початкові значення зсувів нейронів
w=[5 3.5;-1.5 -4;-1 1]; % – початкові значення ваг нейронів; замість -1
                        %   і 1 задати -2 і 2
```

```
while e>0.001              % – цикл з невизначеним числом кроків
  c=c+1;                   % – поточний номер епохи навчання
  c1(c)=c;                 % – накопичуваний масив кількості епох
  for i=1:3                % – цикл по нейронах – прохід вперед
    if i==3
      u(i)=w(i,1)*u(1)+w(i,2)*u(2)+b(i);% – розрахунок реакції нейронів
      u(i)=1/(1-exp(u(i)));
    else
      u(i)=w(i,1)*u1+w(i,2)*u2+b(i);
      u(i)=1/(1-exp(u(i)));
    end;
  end;

    e=(t-u(3))^2;        % – розрахунок помилок
    E(c)=e;              % – формування масиву помилок

  delta(3)=u(3)*(1-u(3))*e;    % – розрахунок кроків для підстроювання
  dw(3,2)=lr*delta(3)*u(2);    % – ваг і зсувів нейронів
  dw(3,1)=lr*delta(3)*u(1);
  db(3)=lr*delta(3)*b(3);

  delta(1)=u(1)*(1-u(1));
  dw(1,1)=lr*delta(1)*u1;
  dw(1,2)=lr*delta(1)*u2;
  db(1)=lr*delta(1)*b(1);

  delta(2)=u(2)*(1-u(2));
  dw(2,1)=lr*delta(2)*u1;
  dw(2,2)=lr*delta(2)*u2;
  db(2)=lr*delta(2)*b(2);

  for i=3:-1:1               % – цикл по номерах нейронів
    b(i)=b(i)+db(i);         % – розрахунок нових значень зсувів нейронів
    B(c,i)=b(i);             % – формування масиву зсувів нейронів
    for j=2:-1:1             % – цикл по входах нейронів
     w(i,j)=w(i,j)+dw(i,j);% – розрахунок нових значень ваг нейронів
     W(c,i,j)=w(i,j);        % – формування масиву ваг нейронів
    end;
  end;
  if c==10000                % – обмежувач кількості епох
    e=0.0001;
  end;
end;
```

```
figure; % побудова графіків залежності помилки, зсувів, ваг від епох;
plot(c1,E,'-k');

figure;
subplot(3,1,1); plot(c1,B(:,1),'-k'); title('Znachenija smewenija 1 neyrona');
subplot(3,1,2); plot(c1,B(:,2),'-k'); title('Znachenija smewenija 2 neyrona');
subplot(3,1,3); plot(c1,B(:,3),'-k'); title('Znachenija smewenija 3 neyrona');

figure;
subplot(3,2,1); plot(c1,W(:,1,1),'-k'); title('Znachenija 1 wesa 1 neyrona');
subplot(3,2,2); plot(c1,W(:,1,2),'-k'); title('Znachenija 2 wesa 1 neyrona');
subplot(3,2,3); plot(c1,W(:,2,1),'-k'); title('Znachenija 1 wesa 2 neyrona');
subplot(3,2,4); plot(c1,W(:,2,2),'-k'); title('Znachenija 2 wesa 2 neyrona');
subplot(3,2,5); plot(c1,W(:,3,1),'-k'); title('Znachenija 1 wesa 3 neyrona');
subplot(3,2,6); plot(c1,W(:,3,2),'-k'); title('Znachenija 2 wesa 3 neyrona');

toc
```

Числа, що стоять за позначкою коментарів, призначені для використання при дослідженні роботи персептрона.

Висновки за завданням 6.

2.6 Зміст звіту про лабораторну роботу

1) Титульний аркуш.
2) Розділ: Мета лабораторної роботи і Завдання на лабораторну роботу.
3) Розділ: Хід лабораторної роботи - програмні коди до всіх пунктів завдання з додатком результатів виконання всіх операцій і команд.
4) Розділ: Висновки по лабораторній роботі.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Яка модель лежить в основі штучних нейронних мереж?
2. Які витоки виникнення теорії нейронних мереж?
3. Яка область практичного вживання теорії нейронних мереж і чим це пояснюється?
4. Особливості архітектури персептрона?
5. У чому суть навчання одношарового персептрона?
6. Які можливості одношарового персептрона?
7. Що називається епохою?
8. До якого класу алгоритмів навчання відноситься дельта-правило?
9. Які труднощі можуть виникнути з алгоритмом навчання персептрона?
10. У чому відмінність понять уявності і навчаємості нейронної мережі?
11. Що є нейроном персептрона?
12. Яку множину називають лінійно роздільною?
13. У чому особливість багатошарового персептрона?
14. У чому полягає відмінність персептрона Румельхарта від персептрона Розенблатта?
15. За допомогою якого алгоритму відбувається навчання багатошарового персептрона?
16. Яка основна ідея процедури зворотного розповсюдження помилки?
17. Що називають опуклою областю?
18. Що називають обмеженою областю?
19. Чим обмежені класифікаційні можливості тришарової мережі?

ЛІТЕРАТУРА

Основна:

1. Перелигін Б.В., Ткач Т.Б. Застосування штучних нейронних мереж для обробки інформації в технічних системах моніторингу навколишнього середовища: Навчальний посібник. – Одеса: ТЕС, 2014. – 218 с.
2. Руденко О.Г., Бодянский Е.В. Искусственные нейронные сети: Учебное пособие. – Харьков: ООО „Компания СМИТ”, 2005. – 408 с.

Додаткова:

3. Медведев В.В., Потёмкин В.Г. Нейронные сети. MATLAB 6 / Под общей редакцией В.Г. Потёмкина. – М.: ДИАЛОГ-МИФИ, 2002. – 496с. (Пакеты прикладных программ; Кн. 4).
4. Галушкин А.И. Теория нейронных сетей. Кн. 1: Учебное пособие для вузов / Под общей редакцией А.И. Галушкина. – М.: ИПРЖР, 2000. – 416 с. (Нейрокомпьютеры и их применение).
5. Каллан Р. Основные концепции нейронных сетей.: Пер. с англ. – М.: Издательский дом „Вильямс”, 2001. – 27 с.
6. Круглов В.В., Дли М.И., Голунов Р.Ю. Нечёткая логика и искусственные нейронные сети: Учебное пособие для вузов. – М.: Издательство физико-математической литературы, 2001. – 224 с.
7. Круглов В.В., Борисов В.В. Искусственные нейронные сети. Теория и практика, 2-е изд., стереотип. – М.: Горячая линия-Телеком, 2002. – 382 с.
8. Хайкин С. Нейронные сети: полный курс, 2-е изд.: Пер. с англ. – М.: Издательский дом „Вильямс”, 2006. – 1104 с.

ДОДАТОК А

Вимоги до оформлення і форма титульного аркуша звіту про лабораторну роботу

1) Звіт виконується на листах формату 11 (А4) машинописним способом в будь-якому текстовому редакторі.
2) Колір шрифту – чорний, гарнітура – Таймс, кегль – 14, поля з усіх боків – 20 мм.
3) Мета лабораторної роботи і Завдання на лабораторну роботу оформляються в одному розділі з нового аркуша.
4) Хід лабораторної роботи оформляється в одному розділі з нового аркуша.
5) Висновки по лабораторній роботі оформляються в одному розділі з нового аркуша.
6) Допускається для розділу Хід лабораторної роботи, з метою економії паперу, виконання тексту і програмних кодів кеглем 12, при необхідності малюнки можуть виконуватися в кольорі.

Форма титульного аркуша звіту

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК, УПРАВЛІННЯ ТА АДМІНІСТРУВАННЯ
Кафедра автоматизованих систем моніторингу навколишнього середовища

ЗВІТ
про лабораторну роботу

ДОСЛІДЖЕННЯ ПЕРСЕПТРОНА

з дисципліни
ШТУЧНІ НЕЙРОННІ МЕРЕЖІ В ЗАДАЧАХ ОБРОБКИ ДАНИХ

Виконав(ла) студент(ка) групи МК-51
Іванов Петро Сидорович

_____
(підпис студента)

Перевірив Перелигін Б.В.

Оцінка за підготовку до лабораторної роботи	Оцінка за виконання лабораторної роботи	Загальна оцінка

Одеса – 2019