МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторної роботи

„ДОСЛІДЖЕННЯ ПРОЦЕСУ РОЗПІЗНАВАННЯ ДРУКАРСЬКИХ СИМВОЛІВ ЗА ДОПОМОГОЮ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ”

з дисципліни „Штучні нейронні мережі в задачах обробки даних”
для магістрів 2 курсу

Спеціальність: 122 Комп’ютерні науки
Освітньо-професійна програма: Комп’ютерні науки

ЗАТВЕРДЖЕНО
на засіданні групи забезпечення
спеціальності 122 Комп’ютерні науки
« 30 » 03 2024 року
протокол № 7
Голова групи (Кузніченко С.Д.)

ЗАТВЕРДЖЕНО
на засіданні кафедри АСМНСІ
« 4 » березня 2024 року
протокол № 8
Завідувач кафедри (Перелигін Б.В.)

ОДЕСА - 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

ДОСЛІДЖЕННЯ ПРОЦЕСУ РОЗПІЗНАВАННЯ ДРУКАРСЬКИХ СИМВОЛІВ ЗА ДОПОМОГОЮ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

МЕТОДИЧНІ ВКАЗІВКИ ДЛЯ МАГІСТРІВ 2 КУРСУ
ДО ЛАБОРАТОРНОЇ РОБОТИ З ДИСЦИПЛІНИ
„ШТУЧНІ НЕЙРОННІ МЕРЕЖІ В ЗАДАЧАХ ОБРОБКИ ДАНИХ”

ЗАТВЕРДЖЕНО
на засіданні групи забезпечення
спеціальності 122 Комп’ютерні науки
« 4 » березня 2024 року
протокол № 9

ОДЕСА – 2024

Дослідження процесу розпізнавання друкарських символів за допомогою штучних нейронних мереж. Методичні вказівки до лабораторного заняття з дисципліни „Штучні нейронні мережі в задачах обробки даних” для магістрів 2 курсу навчання за спеціальністю 122 Комп’ютерні науки освітньо-професійною програмою Комп’ютерні науки. / Перелигін Б.В., Ткач Т.Б. – Одеса, ОДЕКУ, 2024 р. – 52 с.

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 даних.

Мета дисципліни – підготовка майбутніх фахівців в області обробки даних.

Останнім часом обробка і аналіз подібної інформації стали особливо актуальними. Існуючі численні 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 наступного ухвалення рішень системами управління різного рівня. Фахівці з подібними знаннями потрібні 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, і вивчаються технічні і програмні засоби, які її реалізовують.

Тому завданням дисципліни є вивчення застосування нейромережних технологій обробки моніторингової інформації, що поступає від систем виміру параметрів стану довкілля, штучних супутників Землі, радіолокаційних станцій і інших систем моніторингу довкілля. В результаті її обробки можливе отримання значущих характеристик для ухвалення обґрунтованих рішень в системах управління різних ієрархічних рівнів.

Практична частина дисципліни включає лабораторні роботи по вивченню і дослідженню способів і методів обробки і аналізу моніторингової інформації.

Ця лабораторна робота присвячена дослідженню процедури розпізнавання за допомогою штучних нейронних мереж друкарських символів, а саме, букв і цифр.

В результаті підготовки і проведення лабораторної роботи "Дослідження процесу розпізнавання друкарських символів за допомогою штучних нейронних мереж" студенти повинні здобути:

знання:

- про представлення друкарської інформації в ЕОМ,
- про застосовування штучних нейронних мереж різних парадигм для розпізнавання друкарських символів.

уміння:

- застосовувати штучні нейронні мережі для вирішення завдань розпізнавання друкарських символів.

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

При виконанні лабораторної роботи кожен студент відповідає на теоретичні питання 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 доводиться 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 питання він отримує допуск до виконання лабораторної роботи, після чого практично виконує лабораторну роботу. Якщо у студента немає допуску, то і роботу він не виконує.

Після демонстрації викладачеві результатів виконання лабораторної роботи і отримання його дозволу студент оформлює 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 електронно-обчислювальних машинах

1) Включити апаратуру комп'ютера вимикачами на корпусах в послідовності: стабілізатор напруги, відеотермінал, процесор.
2) Відрегулювати яскравість свічення екрану відеотерміналу, фокусування, контрастність. Не слід встановлювати велику яскравість свічення екрану щоб уникнути стомлення очей. Її слід встановити так, щоб відношення яскравості екрану до яскравості поверхонь, що оточують його, в робочій зоні було не більш, 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 зручного упору ліктьового суглоба.
5) Після закінчення роботи вимкнути апаратуру в порядку, зворотному включенню.
6) Під час лабораторної роботи не дозволяються сторонні розмови, дратівливі шуми.

При проведенні лабораторної роботи ЗАБОРОНЯЄТЬСЯ:

1) Користуватися кабелями і проводами з пошкодженою ізоляцією
2) Залишати під напругою кабелі і дроти з неізольованими провідниками.
3) Застосовувати саморобні подовжувачі, що не відповідають вимогам Правил устрою 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 процесу розпізнавання друкарських символів за допомогою штучних нейронних мереж"

1.1 Загальні відомості

Штучні нейронні мережі засновані на простій біологічній моделі нервової системи, що складається з 10^{11} нейронів, кожен 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 з спробами відтворення здатності нервових біологічних систем до навчання і виправлення помилок, а також з моделювання низькорівневої структури мозку.

Ця теорія розвивалася з середини 20 століття і з кінця 90-х років знайшла широке практичне застосування: в космонавтиці і аеронавтиці – для імітації траєкторій польоту і побудови систем автоматичного пілотування; у військовій справі – для управління зброєю і стеженням за цілями; у електроніці – для розробки систем машинного зору і синтезу мови; у медицині – для діагностики захворювань і конструювання протезів; у виробництві – для управління технологічними процесами, роботами і так далі. 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.

Пакети прикладних програм містять засоби для побудови нейронних мереж, які базуються на поведінці математичного аналога нейрона. Пакети забезпечують ефективну підтримку проектування, навчання, аналізу і моделювання безлічі відомих типів мереж – від базових моделей персептрона до асоціативних мереж, які самоорганізуються. Для кожного типу архітектури і повчальних правил є функції ініціалізації, навчання, адаптації, створення, моделювання, відображення, оцінки і демонстрації, а також приклади застосування.

1.2 Штучна нейронна мережа прямого поширення

1.2.1 Архітектура

Прикладом штучної нейронної мережі прямого поширення є багатошаровий персептрон, при навчанні якого використовується алгоритм зворотного поширення помилки в різних модифікаціях. Особливістю мережі прямого поширення є наявність більш ніж одного навченого шару.

Архітектура мережі прямого поширення представлена на рис. 1.1.

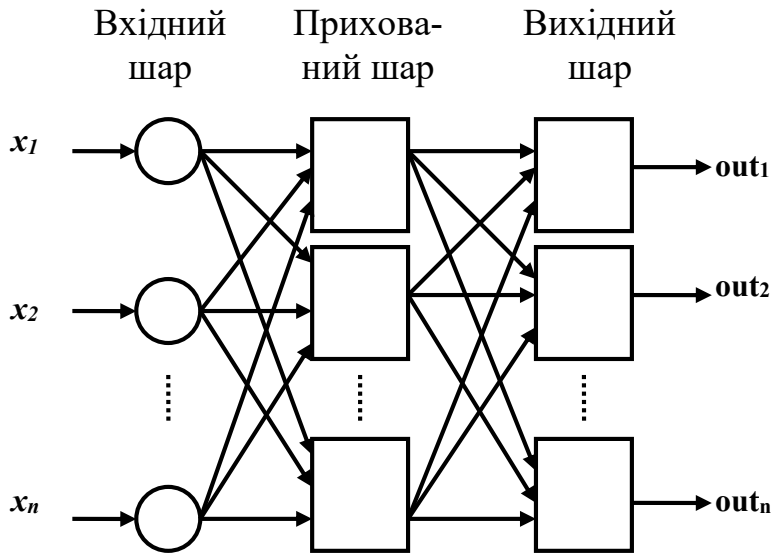


Рис. 1.1 – Архітектура багатошарової (двошарової) штучної нейронної мережі прямого поширення

1.2.2 Навчання

Типовим, найбільш відомим і найпоширенішим представником алгоритмів навчання мереж прямого поширення є алгоритм зворотного поширення помилки – це детерміністський ітераційний градієнтний алгоритм навчання з вчителем.

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

Основна ідея зворотного поширення полягає в тому, як отримати оцінку помилки для нейронів прихованих шарів. Помітимо, що відомі помилки, які утворюються нейронами вихідного шару, виникають внаслідок невідомих 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 отримати, як зважену суму помилок наступних шарів. При навчанні інформація поширюється від нижчих шарів ієрархії до вищих шарів, а оцінки помилок, яка утворюється мережею, – у зворотному напрямі, що і відбите в назві методу.

Розглянемо цей алгоритм.

Для спрощення розгляду обмежимося випадком, коли мережа має лише один прихований шар (рис. 1.2).

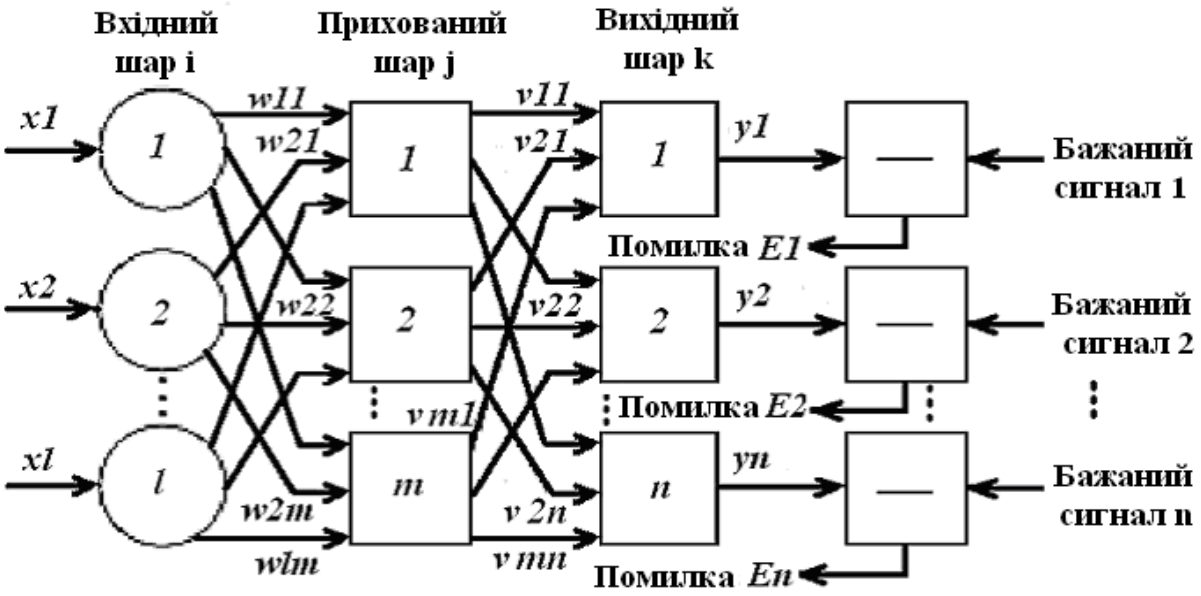


Рис. 1.2 – Процедура навчання мережі прямого поширення

Матрицю вагових коефіцієнтів від входів до прихованого шару позначимо W, а матрицю вагів, які сполучають прихований і вихідний шар, – V. Для індексів приймемо наступні позначки: входи нумеруватимемо лише індексом i ($\overline{1...l}$), елементи прихованого шару – індексом j ($\overline{1...m}$), а виходи, відповідно, індексом k ($\overline{1...n}$).

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

Структура алгоритму :

– *Крок 0.* Початкові значення вагів усіх нейронів усіх шарів $V(t=0)$ і $V(t=0)$ 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 за наступними формулами:

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

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

де $f(x)$ – сигмоїдна функція;

– *Крок 2.* Нехай функціонал квадратичної помилки мережі для цього вхідного образу має вигляд:

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

Цей 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 виді 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 лише через саме значення функції, $f'(x)=f(1-f)$. Таким чином, усі необхідні величини для налаштування вагів вихідного шару V отримані (рис. 1.3).

– *Крок 3. На* цьому кроці виконується налаштування вагів прихованого шару (рис. 1.4). Градієнтний метод як і раніше дає:

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

Обчислення похідних виконуються за тими ж формулами, за винятком деякого ускладнення формули для помилки δ_j.

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

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

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

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

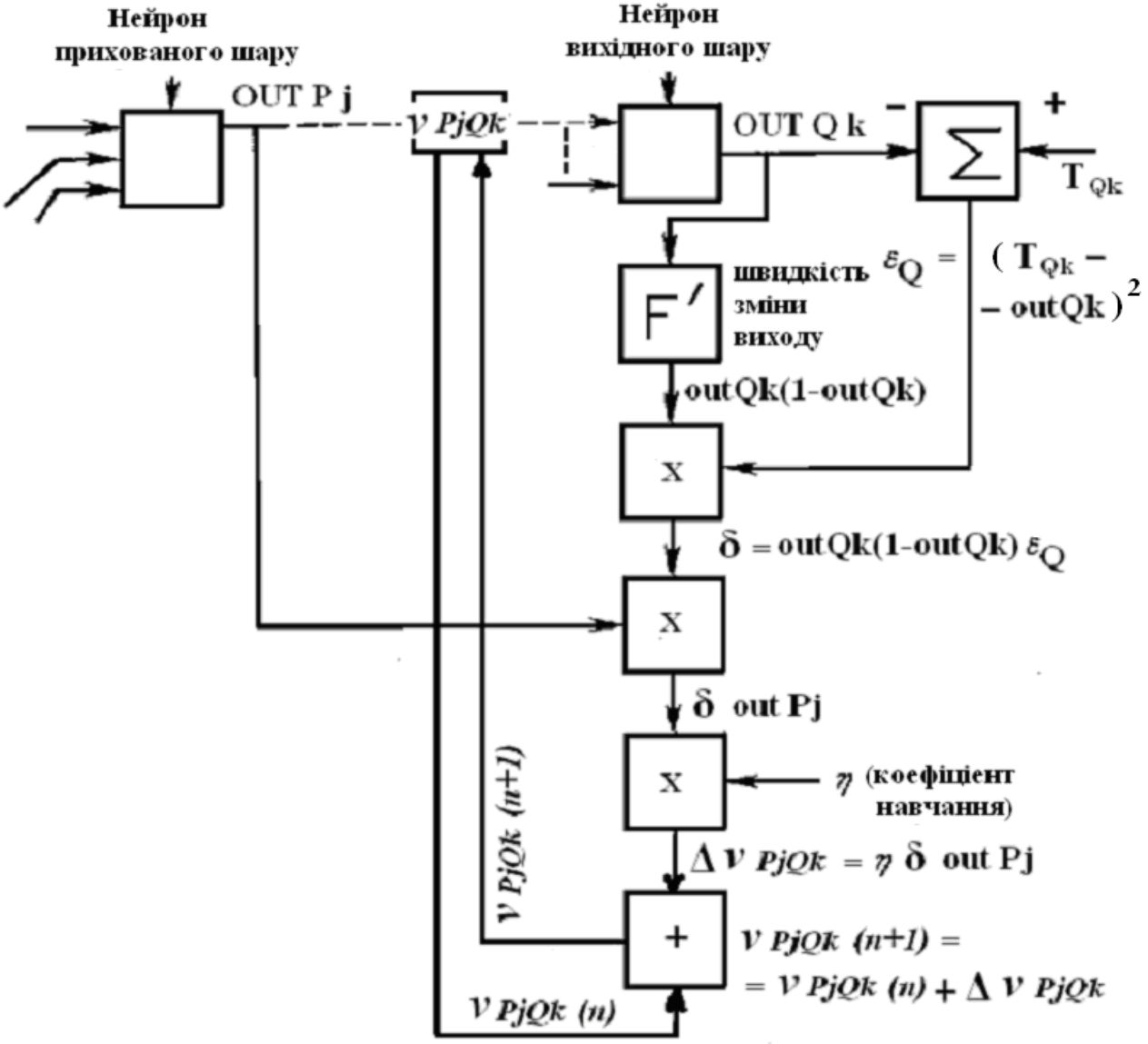


Рис. 1.3 – Налаштування вагів вихідного шару

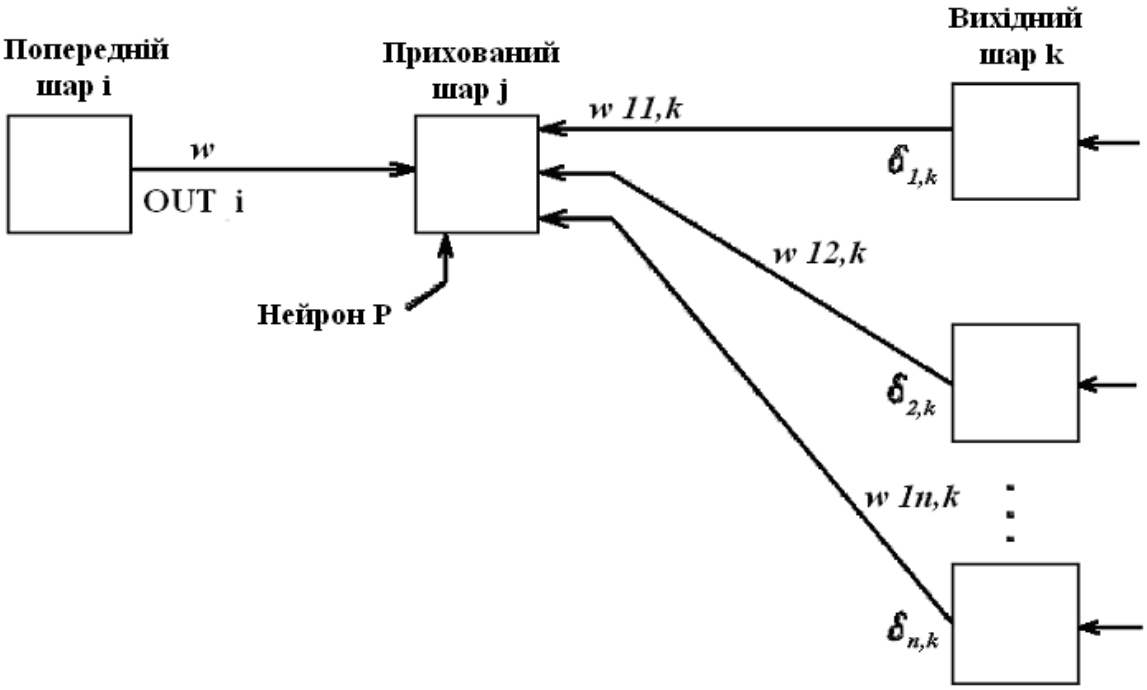


Рис. 1.4 – Налаштування вагів прихованого шару

При обчисленні δ_j тут і був застосований принцип зворотного поширення помилки: часткові похідні беруться лише по змінних наступного шару. З отриманих формул модифікуються ваги нейронів прихованого шару. Якщо в нейронній мережі є декілька прихованих шарів, процедура зворотного поширення застосовується послідовно для кожного з них, починаючи з шару, який передує вихідному, і далі до шару, що йде за вхідним шаром. При цьому формули зберігають свій вид з 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 вирішення задачі оптимізації функціонала помилки градієнтним методом. Суть зворотного поширення помилки полягає в тому, що для її оцінки для нейронів прихованих шарів можна прийняти зважену суму помилок наступного шару.

Параметр h має сенс темпу навчання і обирається досить малим для забезпечення збіжності методу. Необхідно відмітити, 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, налаштування вагів виконується незалежно для кожної пари образів навчальної вибірки. При цьому поліпшення функціонування на деякій заданій парі може, взагалі, призводити до погіршення роботи на попередніх образах. У цьому сенсі, немає достовірних (окрім дуже великої практики застосування методу) гарантій збіжності.

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

1.2.3 Алгоритм градієнтного спуску

У системі комп'ютерної математики (СКМ) застосування для вивчення мережі принципу зворотного поширення помилки на основі алгоритму градієнтного спуску з адаптивним налаштуванням параметра темпу навчання здійснюється за командою traingdx.

По цій команді береться мережа NET, застосовуються для її навчання вхідний вектор *X* і цільовий вектор *T* і повертається навчена мережа NET [NET, TR] = traingdx (NET, *X*, *T*) разом з записом результатів навчання TR.

[NET, TR] = traingdx (NET, X, T, X_i, A_i, EW) має додаткові необов'язкові аргументи, необхідні для вивчення динамічних мереж з заданою помилкою.

X_i і A_i – відповідно початкові стани входу і затримки шару, а EW задає значення помилки, використовуваної для вказівки відносної важливості кожного цільового значення.

Навчання відбувається відповідно до заданих за умовчанням параметрів навчання. Будь-який з цих параметрів або усі вони можуть бути перевизначені шляхом додавання в список вхідних аргументів, або шляхом додавання структури *Аргумент з полями*, що має одне або декілька з приведених нижче найменувань:

epochs 1000 – максимальна кількість епох для навчання
goal 0 – мінімальна помилка
lr 0.01 – коефіцієнт навчання
lr_inc 1,05 – коефіцієнт збільшення швидкості навчання
lr_dec 0,7 – коефіцієнт зменшення швидкості навчання
max_fail 5 – максимальна помилка перевірки
max_perf_inc 1.04 – максимальне збільшення продуктивності
mc 0,9 – стала часу
min_grad 1e-5 – мінімальний градієнт продуктивності
show 25 – відображення результатів через вказану кількість епох
showCommandLine false – генерація виведення командного рядка
showWindow true – показати навчання в графічному інтерфейсі користувача GUI
time inf – показати максимальний час навчання в секундах

Щоб запустити алгоритм градієнтного спуску за умовчанням необхідно змінити налаштування параметрів використовуючи команди

```
net.trainFcn = 'traingdx';
net.trainParam
```

1.2.4 Алгоритм Левенберга-Марквардта

Алгоритм Левенберга-Марквардта призначений для оптимізації параметрів нелінійних регресійних моделей при зворотному поширенні помилки. Передбачається, що в якості критерія оптимізації використовується середньоквадратична помилка моделі на навчальній вибірці. Алгоритм полягає в послідовному наближенні заданих початкових значень параметрів до шуканого локального оптимуму (рис. 1.5).

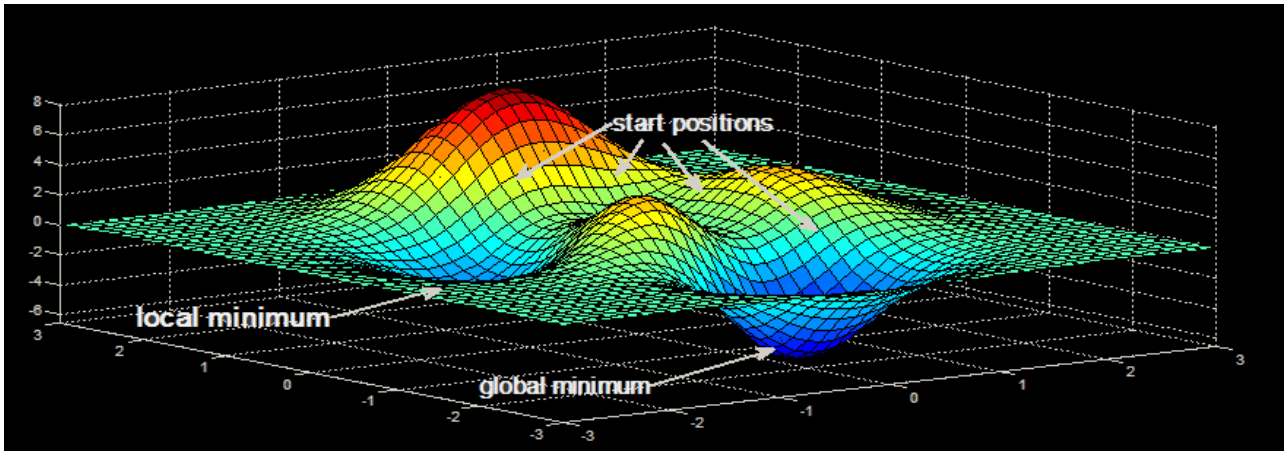


Рис. 1.5 – Поверхня помилки штучної нейронної мережі

Задана навчальна вибірка – множина пар вільної змінної $x \epsilon X^M$ (входи мережі) і залежної змінної $y \epsilon Y^M$. Задана функціональна залежність, що є регресійною моделлю $y = f(w, x_n)$, яка безперервно диференціюється в області $W \cdot X$. Параметр w є вектором вагових коефіцієнтів. Вимагається знайти таке значення вектору w, яке б доставляло локальний мінімум функції помилки

$$E_D = \sum_{n=i}^{N} \left(y_n - f(w, x_n)\right)^2$$

Перед початком роботи алгоритму задається початковий вектор вагових коефіцієнтів w.

1. На кожному кроці ітерації цей вектор замінюється на вектор $w+\Delta w$

$$w = w + \Delta w.$$

Для оцінки приросту Δw використовується лінійне наближення функції

$$f(w + \Delta w, x) \approx f(w, x) + J\Delta w,$$

де J – якобіан функції $f(w,x_n)$в точці w.

Матрицю J наочно можна представити у вигляді

$$J = \begin{bmatrix} \frac{\partial f(w, x_1)}{\partial w_1} & \cdots & \frac{\partial f(w, x_1)}{\partial w_R} \\ \vdots & \ddots & \vdots \\ \frac{\partial f(w, x_N)}{\partial w_1} & \cdots & \frac{\partial f(w, x_N)}{\partial w_R} \end{bmatrix}$$

Тут вектор вагових коефіцієнтів $w = [w_1, \dots, w_R]$.

Приріст Δw в точці w, яка доставляє мінімум E_D дорівнює нулю. Тому для знаходження наступного приросту Δw прирівняємо нулю вектор часткових похідних E_D по w

$$E_D = |y - f(w + \Delta w)|^2,$$

де і

$$f(w + \Delta w) = [f(w + \Delta w, x_1), \dots, f(w + \Delta w, x_N)]^T.$$

Перетворюючи і диференціюючи цей вираз

$$|y - f(w + \Delta w)|^2 = \big(y - f(w + \Delta w)\big)^T \big(y - f(w + \Delta w)\big) = \\ = f^T(w + \Delta w) f(w) - 2y^T f(w + \Delta w) + y^T y$$

отримаємо

$$\frac{\partial E_D}{\partial w} = (J^T J)\Delta w - J^T \big(y - f(w)\big) = 0\,.$$

Таким чином, щоб знайти значення Δw треба вирішити систему лінійних рівнянь

$$\Delta w = (J^T J)^{-1} J^T \big(y - f(w)\big).$$

Оскільки число обумовленості матриці J^TJ є квадрат числа обумовленості матриці J, то матриця J^TJ може виявитися істотно виродженою. Тому Марквардтом введений параметр регуляризації $\lambda \geq 0$.

2. $\Delta w = (J^T J + \lambda I)^{-1} J^T \big(y - f(w)\big)$, де I – одинична матриця. Цей параметр призначається на кожній ітерації алгоритму. Якщо значення похибки E_D зменшується швидко, мале значення λ зводить цей алгоритм до алгоритму Гауса-Ньютона.

3. Алгоритм зупиняється у тому випадку, якщо приріст Δw в наступній ітерації менше заданого значення, або якщо вектор вагових коефіцієнтів доставляє помилку E_D, меншу заданої величини, або якщо вичерпано число циклів навчання мережі. Значення вектору w на останній ітерації вважається шуканим.

Недолік алгоритму – значне зменшення швидкості апроксимації при

збільшенні параметра λ.

У СКМ застосування для навчання мережі принципу зворотного поширення помилки на основі алгоритму Левенберга-Марквардта здійснюється за командою trainlm – це мережна навчальна функція, яка оновлює ваги і значення зміщень згідно оптимізації Левенберга-Марквардта. Цей алгоритм часто є найшвидшим алгоритмом зворотного поширення в інструментарії вивчення нейронних мереж і рекомендується як алгоритм першого вибору, хоча він вимагає більше пам'яті, ніж інші алгоритми.

По цій команді береться мережа NET, застосовуються для її навчання вхідний вектор *X* і цільовий вектор *T* і повертається навчена мережа NET [NET, TR] = trainlm (NET, X, T) разом із записом результатів вивчення TR.

[NET, TR] = trainlm (NET, X, T, X_i, A_i, EW) має додаткові необов'язкові аргументи, необхідні для вивчення динамічних мереж з заданою помилкою.

X_i і A_i – відповідно початкові стани входу і затримки шару, а EW задає значення помилки, яка використовується для вказівки відносної важливості кожного цільового значення.

Навчання відбувається відповідно до заданих за умовчанням параметрів навчання. Будь-який з цих параметрів або усі вони можуть бути перевизначені шляхом додавання в список вхідних аргументів, або шляхом додавання структури *Аргумент з полями*, яке має одне або декілька з приведених нижче найменувань:

epochs 1000 – максимальна кількість епох для навчання
goal 0 – мінімальна помилка
mu 0.001 – початковий мю
mu_inc 10 – коефіцієнт збільшення мю
mu_dec 0,1 – коефіцієнт зменшення мю
mu_max 1e10 – максимальний мю
max_fail 6 – максимальна помилка перевірки
min_grad 1e-7 – мінімальний градієнт продуктивності
show 25 – відображення результатів через вказану кількість епох
showCommandLine false – генерація виведення командного рядка
showWindow true – показати навчання в графічному інтерфейсі користувача GUI
time inf – показати максимальний час навчання в секундах

Щоб запустити алгоритм Левенберга-Марквардта за умовчанням необхідно змінити налаштування параметрів використовуючи команди

```
net.trainFcn = 'trainlm';
net.trainParam
```

1.3 Штучна нейронна мережа Хопфілда

1.3.1 Архітектура

Розглянута раніше нейромережна архітектура відноситься до класу мереж

з спрямованим потоком поширення інформації і не містить зворотних зв'язків. Після навчання на етапі функціонування мережі кожен нейрон виконує свою функцію – передачу вихідного сигналу рівно один раз.

У загальному випадку може бути розглянута нейронна мережа, яка містить довільні зворотні зв'язки, тобто шляхи, передавальні сигнали від виходів до входів. Відгук таких мереж є динамічним, тобто після подачі нового входу обчислюється вихід і, передаючись по зворотному зв'язку, модифікує вхід. Потім вихід повторно обчислюється, і процес повторюється знову і знову. Для стійкої мережі послідовні ітерації призводять до все менших змін виходу, і в результаті вихід стає постійним. Для багатьох мереж процес ніколи не закінчується, такі мережі називають *нестійкими*. Нестійкі мережі мають цікаві властивості і можуть розглядатися як приклад *хаотичних* систем, але для більшості практичних застосувань використовуються мережі, які дають постійний вихід. Подібною мережею є мережа Хопфілда.

Хопфілд використовував свою мережу для моделювання спінових слідів. Під цим маються на увазі матеріали, атоми яких мають магнітний диполь. При моделюванні кожному диполю відповідав нейрон, орієнтація диполя в магнітному полі здійснювалася збудженням відповідного нейрона, а мережа описувала магнітні взаємодії полів. Розглянуті ним застосування включають асоціативну пам'ять, аналого-цифрове перетворення, рішення задачі оптимізації.

Мережа Хопфілда має наступні особливості:

а) мережа є одношаровою і містить N нейронів, число яких є одночасно числом входів і виходів мережі,

б) кожен нейрон мережі пов'язаний з усіма іншими нейронами, а також має один вхід, на який подається вхідний сигнал,

в) жоден нейрон не має власного зворотного зв'язку ($w_{ij} = 0$),

г) ваги мережі є симетричними, тобто вага зв'язку між i-м і j-м нейронами дорівнює вазі зв'язку між j-м і i-м нейронами $w_{ij} = w_{ji}$,

д) кожен нейрон має порогову функцію активації,

е) вхідними є двійкові сигнали.

Структурна схема мережі Хопфілда наведена на (рис. 1.6).

Слід зазначити, що оскільки "мережна" архітектура мозку включає зворотні зв'язки, то мережа Хопфілда в цьому відношенні найбільш відповідає природному процесу обробки інформації.

Ця мережа не використовує, строго кажучи, ні навчання з викладачем, ні навчання без викладача. Вагові коефіцієнти в ній розраховуються лише перед початком функціонування мережі на основі інформації про оброблювані дані, і усе вивчення мережі зводиться саме до цього розрахунку. З одного боку, надання апріорної інформації можна розцінювати як допомогу вчителя, а з іншої – мережа фактично просто запам'ятовує образи до того, як на її вхід поступають реальні дані, і не може змінювати свою поведінку, тому говорити про зворотний зв'язок з вчителем не доводиться.

У мережах з зворотними зв'язками стан нейронів обчислюється до тих

пір, поки вони не виявляться сталими, не змінюваними згодом. Можна показати, що мережа Хопфілда за певних умов сходиться до сталого стану за кінцевий час.

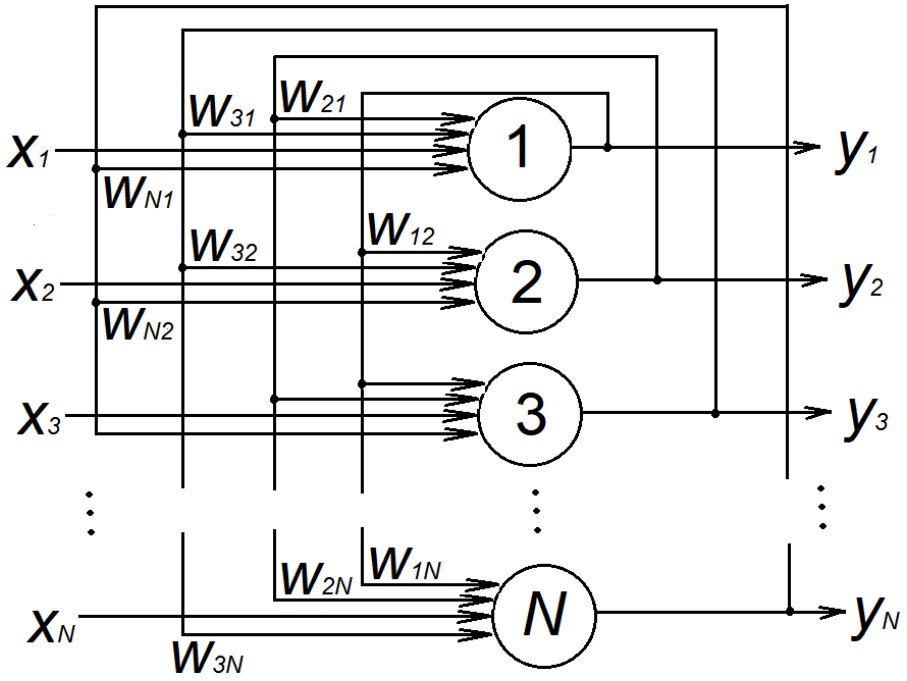


Рис. 1.6 – Структурна схема мережі Хопфілда

Мережі надається P образів, описуваних N-мірними (N – число нейронів) векторами $x = (x_1, x_2,..., x_N)^T$ з компонентами, які набувають двійкових значень, наприклад, «–1» і «+1». Позначимо вектор вхідних сигналів, який описує p-й образ, як как $x^p (p = 1,P)$. Коли мережа розпізнає наданий образ, вона сформує вектор вихідних сигналів, компоненти якого співпадають з відповідними компонентами образу, тобто $y = (y_1, y_2,..., y_N)^T$ – вектор вихідних сигналів мережі. Інакше вихідний сигнал не співпадатиме ні з одним з наданих.

Розрізняють три стадії (фази) функціонування мережі:

— ініціалізація;
— надання вхідного образу;
— обчислення стану нейронів.

Ініціалізація. На цій стадії мережі встановлюються наступні значення вагових коефіцієнтів:

$$w_{ij} = \begin{cases} \sum_{p=0}^{p} x_i^p x_j^p, & i \neq j \\ 0, & i = j \end{cases},$$

тут x_i^p і x_j^p – i-а і j-а компоненти вектору x^p.

Цю стадію можна розглядати як стадію навчання мережі, після закінчення якої вона стає здатною правильно відтворювати образи, які неї надаються. Говорять, що мережа здатна до самоасоціативної роботи.

Надання мережі вхідного образу фактично здійснюється безпосередньою початковою (нульовою) установкою компонент вихідних сигналів

$$y_i(0) = x_i,\ i = \overline{1, N}\ ,$$

тому позначення на схемі мережі вхідних сигналів в явному виді носить чисто умовний характер.

Обчислення станів нейронів. За формулою

$$z_j(k+1) = \sum_{i=1}^{N} w_i\, y_i(k) + x_j\ ,$$

визначаються послідовні стани нейронів, на підставі чого розраховуються відповідні значення вихідних сигналів

$$y_j(k+1) = f_\alpha\left(z_j(k+1)\right) = \begin{cases} 1, & \text{если } z_j(k+1) > \theta_j; \\ 0, & \text{если } z_j(k+1) < \theta_j; \\ y_j(k) & \text{в інших випадках,} \end{cases}$$

де $f_\alpha = (\cdot)$ – порогова активаційна функція; θ_j – заданий поріг.

Якщо вихідні значення змінилися, тоді за останніми двома формулами розраховуються стани і вихідні сигнали мережі, якщо не змінилися – робота мережі завершується (мережа знаходиться в стійкому стані). В цьому випадку на виходах мережі сформувався вектор, якнайкраще відповідний наданому образу.

Активація мережі Хопфілда може здійснюватися асинхронно, коли на кожному такті змінює свій стан лише один випадковим чином обраний нейрон, і синхронно, коли усі нейрони змінюють свій стан одночасно.

1.3.2 Навчання і робота

Робота мережі Хопфілда може бути пояснена в термінах енергетичного ландшафту (за аналогією з рис. 1.5). Є ландшафт, який представляє собою гористу місцевість, на вершині якої знаходиться куля. Потім куля котиться по схилу, поки не зупиниться в якій-небудь низині (западині). Ці низини відбивають стійкі стани мережі, і кожна з них відповідає певному наданому образу (образу навчання). У такому представленні куля, яка має велику потенційну енергію, котиться в низину з меншою потенційною енергією, досягаючи локального мінімуму. Щоб знову опинитися в початковому стані, він повинен зробити у фізичному сенсі роботу, тобто витратити енергію. Таким чином, робота мережі Хопфілда може бути охарактеризована деякою енергетичною функцією.

Якщо при аналізі персептрона вибір такої функції (функціонала) особливих ускладнень не викликав, оскільки були відомі реальні і бажані

значення виходів мережі, тоді в даному випадку ситуація декілька інша. По-перше, структура мережі Хопфілда не дозволяє заздалегідь визначити шлях рішення, тобто бажану або необхідну послідовність станів нейронів. По-друге, наявність зворотних зв'язків призводить до того, що виходи мережі у будь-який момент часу представляють набір входів. Крім того, має бути врахована відсутність у нейронів власних зворотних зв'язків. Вказані особливості мережі відбиваються в енергетичній функції вигляду

$$E = -\frac{1}{2}\sum_i \sum_{j \neq i} w_{ij} y_i y_j - \sum_i y_i x_i + \sum_i y_i \theta_i \,, \qquad (1.1)$$

де θ_i – значення порогу w_{ij} –го нейрона;

w_{ij} – вага зв'язку i-го і j-го нейронів;

x_i – зовнішній вхідний сигнал i-го нейрона (постійна величина для спостережуваного моменту часу);

y_i – значення вихідного сигналу i-го нейрона.

Хопфілд показав, що при активізації мережі ця функція не зростає і досягає локального мінімуму в деякому сталому стані. А оскільки число таких стійких станів обмежене, мережа за певних умов досягає одного з них за кінцеве число ітерацій. При цьому, як вже вказувалося, низини енергетичного ландшафту (енергетичні мінімуми) відповідають образам, які зберігаються. Щоб мережа Хопфілда правильно класифікувала образи, ці низини не повинні перекриватися.

Запам'ятовування образу в мережі Хопфілда. Останній функціонал (1.1) можна розглядати як критерій помилки (чим далі від рішення, тим його значення більше). Щоб зберігати надані мережі образи, необхідно мінімізувати значення енергетичної функції для кожного з них, тобто визначити локальні мінімуми енергетичного ландшафту. Проте додавання нових образів не повинне знищувати вже наявну інформацію. А оскільки вся інформація відносно образів, яка зберігається, міститься у ваговій матриці, завдання навчання зводиться до визначення значень вагів, які доставляють мінімум енергетичній функції.

При мінімізації цього функціонала необхідно враховувати наступне. Як видно, мінімум цього виразу знаходиться в від'ємній області. Для виконання умови $\sum_i y_i \theta_i < 0$ необхідно, щоб компоненти y_i і θ_i постійно мали протилежні позначки. При апріорі заданому θ_i це неможливо, тому доцільно прийняти $\theta_i = 0$, що призводить до критерію у вигляді

$$E = -\frac{1}{2}\sum_i \sum_{j \neq i} w_{ij} y_i y_j - \sum_i y_i x_i. \qquad (1.2)$$

Позначимо за аналогією з вищевикладеним x_m, y_m – компоненти m-го образу. Тоді матрицю вагів можна розбити на дві підматриці, перша з яких (w_{ij}^1) відбиває зв'язки усіх образів, окрім m-го, а друга (w_{im}) – лише m-й образ

$$E = -\frac{1}{2}\sum_i \sum_{j\neq i} w_{ij}^1 x_i x_j - \sum_{i\neq m} x_i\, y_i - \frac{1}{2}\sum_{i\neq m} w_{im}\, y_i y_m - x_m y_m. \quad (1.3)$$

Перші два доданки можна розглядати як збурення, що виражає загальний вплив усіх образів, окрім *m*-го, на функціонал, а другі два – як корисний сигнал. Так, щоб мінімізувати цей останній функціонал для заданого *m*-го образу, необхідно розглянути другий доданок, тобто

$$E_m = -\frac{1}{2}\sum_{i\neq m} w_{im}\, y_i y_m - x_m y_m. \quad (1.4)$$

Мінімізація його еквівалентна максимізації

$$\sum_{i\neq m} w_{im}\, y_i y_m + x_m y_m. \quad (1.5)$$

Таким чином, вибір $w_{im} = y_i y_m$ забезпечує максимум цьому останньому функціоналу (1.5). Неважко побачити, що за нульових початкових умів надання на вхід образу *x* призводить до появи на виході мережі вектору *y=x*. Цим пояснюється установка вагових коефіцієнтів на стадії ініціалізації.

Виклик образу. Після того, як усі надані образи запам'ятовуються мережею (побудований енергетичний ландшафт), інформація про них може бути отримана шляхом мінімізації функціонала помилки на фазі навчання. Використання енергетичного ландшафту дозволяє визначити вплив окремої вершини на енергію, а мінімізація енергії – шлях переходу мережі в який-небудь стійкий стан.

Нехай енергетична функція мережі Хопфілда у момент часу *k* визначається відповідно до (1.1) виразом

$$E(k) = -\frac{1}{2}\sum_i \sum_{j\neq i} w_{ij}\, x_i(k) x_j(k) + \sum_i x_i\,(k)\theta_i. \quad (1.6)$$

Вхідні сигнали мережі в даний момент часу не змінюються. Збудження нейронів мережі у момент часу *(k+1)* призводить до зміни її енергетичної функції, яка обчислюється так:

$$\Delta E(k+1) = E(k+1) - E(k) == \left[-\frac{1}{2}\sum_i \sum_{j\neq i} w_{ij}\, x_i(k+1) x_j(k+1) + \sum_i x_i\,(k+1)\theta_i\right] - \left[-\frac{1}{2}\sum_i \sum_{j\neq i} w_{ij}\, x_i(k) x_j(k) + \sum_i x_i\,(k)\theta_i\right]. \quad (1.7)$$

Припустимо, що у момент часу *(k+1)* змінюється лише стан одного *m*-го нейрона (з нульового переходить в одиничне або навпаки), всі інші нейрони свого стану не змінюють. Внаслідок цього в (1.7) усі доданки з *i=m*і *j=m* однакові. Враховуючи, що в мережі Хопфілда $w_{ij} = 0$ $w_{im} = w_{mi}$ для *i=j≠m*, останній вираз можна переписати таким чином:

$$\Delta E(k+1) = \left[-\sum_{j\neq m} w_{im}\, x_i(k+1)x_m(k+1) + \theta_j x_m(k+1)\right] - \\ -\left[-\sum_{i\neq m} w_{im}\, x_i(k)x_m(k) + \theta_j x_m(k)\right] \quad (1.8)$$

або

$$\Delta E(k+1) = [-\sum_{i\neq m} w_{im}\, x_i(k+1) + \theta_m]x_m(k+1) - \\ -[-\sum_{i\neq m} w_{im}\, x_i(k) + \theta_m]x_m(k). \quad (1.9)$$

Внаслідок того, що для усіх нейронів $i\neq m$ $x_i(k+1) = x_i(k)$, отримуємо

$$\Delta E(k+1) = [-\sum_{i\neq m} w_{im}\, x_i(k+1) + \theta_m] \cdot [x_m(k+1) - x_m(k)] = [z_m(k+ \\ 1) - \theta_m]\Delta x_m(k+1), \quad (1.10)$$

де $\Delta x_m(k+1) = x_m(k+1) - x_m(k)$.

При аналізі (1.10) можливі два випадки.

1. $Z_m(k+1) > \theta_m$.

Це означає, що m-й нейрон буде активований, тобто на попередньому такті він був пасивним. Таким чином, $\Delta x_m(k+1) > 0$ и $\Delta E(k+1) < 0$.

2. $Z_m(k+1) < \theta_m$.

В цьому випадку нейрон перейде в нульовий стан з одиничного, тобто Δ $\Delta x_m(k+1) < 0$, а тау і $Z_m(k+1) - \theta_m < 0$ тоді знову $\Delta E(k+1) < 0$.

Отже, при збудженні m-го нейрона асинхронної мережі Хопфілда енергетична функція на кожному такті зменшується. Виклик образу з мережі відповідає проходженню через послідовність станів, кожне наступне з яких має нижчу енергію в порівнянні з попереднім. Цей спуск в стани з нижчим енергетичним рівнем триває до досягнення мережею енергетичного мінімуму, тобто до відтворення мережею образу, який викликається. Цей процес є аналогом градієнтного спуску, який сходиться до локального екстремуму, і пояснює використовуваний в мережі механізм навчання.

Необхідно відмітити наступне. Зазвичай передбачається, що вхідний сигнал x_i діє короткий час і в розрахунках не враховується, тобто замість (1.1) розглядається функціонал

$$E = -\frac{1}{2}\sum_i \sum_{j\neq i} w_{ij} y_i\, y_j + \sum_i y_i\, \theta_i. \quad (1.11)$$

Таким чином, алгоритм функціонування мережі Хопфілда полягає в наступному.

1. *Навчання.*

Надаються образи $x^p (p = \overline{1,P})$. Обчислюються елементи w_{ij} матриці вагів і для активних нейронів (що знаходяться в стані +1) на підставі їх поточного і попереднього станів обчислюється енергія кожного образу

$$E = -\frac{1}{2}\sum_{i=1}^{N} x_i^p(k+1) \sum_{j=1}^{N} w_{ij} x_j^p(k). \quad (1.12)$$

2. *Розпізнавання.*

Надаються образи, можливо, спотворені $\tilde{x}$ і обчислюються послідовні стани і значення вихідних сигналів нейронів до досягнення ними стійких станів.

Синхронна мережа Хопфілда. Як вже відзначалося, в цій мережі нейрони змінюють свої стани одночасно, прагнучі перейти в деякі стійкі стани. Проте при цьому поняття стійкого стану має декілька інший сенс. Розглянемо це на прикладі. Нехай на виходах синхронної мережі Хопфілда, заданою матрицею вагів $w_{ii} = 0, \quad w_{ji} = w_{ji}\left(i, j = \overline{1,3}\right)$ та значеннями порогів $\theta_i = -1$, в деякий момент часу k при наданні на її входи вектору $x(k) = (000)^T$ з'являються одиничні сигнали $y(k) = (111)^T$. Тоді в наступний *(k+1)* момент часу нейрони перейдуть в нові стани, визначувані як

$$z_i(k+1) = 0 + (-1) \cdot 1 + (-1) \cdot 1 = -2 ,$$

і на їх виходах з'являться сигнали

$$x_i\,(k+1) = 0 \;\; (i = \overline{1,3}).$$

На (k+1)-му такті сигнали $y_i\,(k+1)$ поступлять на входи нейронів, переводячи їх в стани

$$z_i(k+2) = 0 + (-1) \cdot 0 + (-1) \cdot 0 = 0.$$

після чого на їх виходах з'являться сигнали

$$y_i\,(k+2) = 1, \left(i = \overline{1,3}\right)$$

тобто мережа перейшла в початковий стан. Неважко помітити, що мережа осцилює.

З енергією мережі відбувається наступне. У момент часу k значення енергетичної функції (1.1) було дорівнювало

$$E(k) = -\frac{6}{2}[(-1) \cdot 0 \cdot 1] + 3[0 \cdot (-1)] = 0,$$

а при переході в новий стан на *(k + 1)*-му такті

$$E(k+1) = -\frac{6}{2}[(-1) \cdot 0 \cdot 0] + 3[0 \cdot (-1)] = 0.$$

Таким чином, хоча синхронна мережа осцилює, значення її енергетичної функції не змінюється.

Використання енергетичної інтерпретації дозволяє побачити основну

відмінність між синхронним і асинхронним способами активації мережі Хопфілда. Дійсно, випадковий вибір вершини при асинхронній активації призводить до зміни шляху досягнення мережею свого стійкого стану, тобто до зміни послідовності аналізу мережею проміжних образів. Отже, асинхронна активація збільшує деяку невизначеність шляху переходу мережі з початкового в кінцевий стан. При синхронній активації усі вершини оновлюються разом, тому проміжні образи не змінюються. Крім того, мережа осцилює між двома різними станами. Обидва ці способи активації призводять до одного результату, тому зазвичай їх вибір не є принциповим.

Безперервна мережа Хопфілда. Безперервний варіант мережі Хопфілда є узагальненням дискретної мережі на випадок використання замість порогової функції активації безперервної. Зазвичай у якості такої функції використовують синусоїдальну або функцію гіперболічного тангенса, а динаміку мережі описують в безперервному часі. У найбільш загальному вигляді такий опис може бути представлений так:

$$\tau_i \frac{dx_i}{dt} = -x + f_i\left(\sum_{j=1}^{n} w_{ij} x_j\right), \quad i = \overline{1,n},$$

де τ_i – деяка стала часу; $f(\cdot)$ – функція активації виду

$$f(x) = \frac{1}{1+\exp(-x)}.$$

Система, динаміка якої описується рівнянням (1.7), може прагнути до деякого стійкого стану, або знаходитися в хаотичному русі, або осцилювати. Вибір симетричної матриці вагових коефіцієнтів дозволяє забезпечити рух системи до стійкого стану. При цьому збіжність гарантується теоремою Коена-Гросберга. Стійкі стани свідчать про те, що система знаходиться в рівновазі і справедлива рівність

$$\frac{dx_i}{dt} = 0 \text{ для всіх } i.$$

В цьому випадку виходи системи визначаються так:

$$x_i = f_i\left(\sum_j w_{ij} x_j\right), \quad i = \overline{1,n},$$

де x, набуває дійсних значень з інтервалу [0, 1].

Динаміка безперервної мережі Хопфілда може бути описана за аналогією з (1.7) шляхом заміни змінних x на стани нейронів $z_i\left(i = \overline{1,n}\right)$

$$z_i = \sum_j w_{ij} x_j, \; i = \overline{1,n}.$$

В цьому випадку система диференціальних рівнянь, які описують мережу,

набуває вигляд

$$\tau_i \frac{dz_i}{dt} = -z_i + \sum_{j=1}^{n} w_{ij} f(z_j), \quad i = 1, 2, \dots, n,$$

а стійкі стани мережі можуть бути визначені таким чином:

$$Z_i = \sum_j w_{ij} f(z_j), \; i = \overline{1,n} .$$

Як і у разі дискретної мережі, аналіз безперервної мережі Хопфілда може бути проведений за допомогою енергетичної функції. Проте доказ збіжності для безперервної мережі загального вигляду є досить складним. Для аналізу безперервного випадку Хопфілд використовував енергетичну функцію наступного вигляду:

$$E = -\frac{1}{2} \sum_i \sum_j w_{ij} x_i x_j + \sum_i \int_0^{x_i} f^{-1}(x) dx .$$

Можна показати, що при переході в стійкий стан, як і в дискретному випадку, енергетична функція мережі зменшується.

1.3.3 Правило Хебба

Правило Хэбба засновано на тому факті, що зміна синаптичної сили (ваги) нейрона пропорційна зміні пре- і постсинаптичних сил, які є, відповідно, вхідними і вихідними сигналами нейрона, і може бути записана так:

$$w_{ij} = \gamma x_i y_j, \;\; i = \overline{1,N} \;\; j = \overline{1,M} \tag{1.13}$$

або в матричному вигляді

$$W = \gamma y x^T, \tag{1.14}$$

де γ – коефіцієнт пропорційності (навчання).

Відмітимо, що оскільки співвідношеннями (1.13), (1.14) описується будь-яка з P навчальних пар (x_p, y_p), слідувало б це представити так:

$$W_p = \gamma y_p x_p^T, \;\; p = \overline{1,P}, \tag{1.15}$$

Тоді матриця вагів W, яка відбиває зберігання в пам'яті усіх P образів, матиме вигляд:

$$W = \sum_{p=1}^{p} W_p . \tag{1.16}$$

Оскільки асоціативна пам'ять заснована на запам'ятовуванні предметів, спостережуваних одночасно, співвідношення (1.13) і (1.14) припускають одночасну появу *x* і *y*.

У записі (1.13) вага w_{ij}, яка є синапсом між входом x_i. і виходом y_j відбиває той факт, що якщо позитивний (негативний) x_i викликає позитивний (негативний) y_j, тоді синоптичний зв'язок посилюється. Така зміна синаптичного зв'язку може бути записана таким чином:

$$W(k+1) = W(k) + \gamma y(k) x^T(k), \tag{1.17}$$

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

Правило (1.17), яке містить фактичні значення входів і виходів мережі і яке не використовує ніякої інформації, яка стосується необхідної мети, називається *неконтрольованим правилом навчання Хебба*. Згідно з цим правилом, зміна ваги пропорційна виникненню активності на будь-якій стороні синапсу, тобто вага збільшується не лише коли *x* і *y* позитивні, але і коли обоє змінні негативні. Іншими словами, правило (1.13) засноване на обчисленні кореляції між вхідними і вихідними сигналами мережі.

Співвідношення (1.17) може бути переписане у вигляді

$$W(k) = W(0) + \gamma \sum_{i=1}^{k} y(i) x^T(i) \tag{1.18}$$

звідки видно, що при тривалому наданні входів, тобто з зростанням *k* значення вагів можуть стати скільки завгодно великими.

Таким чином, зі збільшенням кількості надання образів вагові коефіцієнти зростають і з часом можуть стати скільки завгодно великими. Це не узгоджується з біологічними принципами, оскільки синаптичні ваги не можуть зростати до безкінечності, і є істотним недоліком неконтрольованого правила (1.17).

Іншим недоліком, тісно пов'язаним з першим, є відсутність в (1.17) механізму зменшення вагів, що призводить знову-таки до зростання вагів мережі і її неправильної реакції при дії шумів на входи і виходи мережі.

Для усунення вказаних недоліків в алгоритмі (1.17) використовують механізм забування, який має вигляд

$$\begin{aligned} W(k) &= W(k-1) + \gamma y(k) x^T(k) - \alpha W(k-1) = \\ &= (1-\alpha) W(k-1) + \gamma y(k) x^T(k), \end{aligned} \tag{1.19}$$

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

При $\alpha \to 0$ алгоритм (1.19) перетворюється на (1.17), а при $\alpha \to 1$ алгоритм забуває стару інформацію і пам'ятає тільки найновішу, усуваючи тим самим нескінченне зростання вагових коефіцієнтів. Максимальне значення $w_{ij}^{\max}$ визначається величинами α і γ. Оскільки максимізація швидкості навчання відбувається при $x_i = y_i = 1$, для сталого стану, коли ваги не змінюються,

можна записати

$$w_{ij}^{\max} = (1-\alpha)w_{ij}^{\max} + \gamma x_i y_j = (1-\alpha)w_{ij}^{\max} + \gamma, \qquad (1.20)$$

звідки

$$w_{ij}^{\max} = \frac{\gamma}{\alpha}. \qquad (1.21)$$

Правило (1.20) гарантує, що малі зміни вагових параметрів, що виникають внаслідок дії шуму, з часом зникнуть, тобто шум не приведе до появи помилкових асоціацій.

З іншого боку, за відсутності нових навчальних сигналів асоціації, що зберігаються мережею, втрачаються. Так, при $x_i = 0$ з (1.17) отримуємо

$$w_{ij}(k) = (1-\alpha)w_{ij}(k-1), \qquad (1.22)$$

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

Таким чином, використання правила навчання з забуванням вимагає повторення надання мережі навчальних послідовностей.

У зв'язку з цим доцільною представляється така модифікація алгоритму (1.17), яка дозволяла б використовувати механізм забування (обмеження зростання елементів вагової матриці), тільки якщо нейрон активний, тобто коли $y \neq 0$. Ця ідея реалізується так:

$$w_{ij}(k) = w_{ij}(k-1) + \gamma y_i(k)x_j(k) - \alpha y_i(k)w_{ij}(k-1). \qquad (1.23)$$

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

Якщо в (1.23) обрати $\gamma=\alpha$, отримаємо співвідношення

$$w_{ij}(k) = w_{ij}(k-1) + \alpha y_i(k)\Big(x_j(k) - w_{ij}(k-1)\Big), \qquad (1.24)$$

зване *правилом вхідної зв'язки* або *стандартним правилом навчання вхідної зірки.*

Особливістю правила (1.24) є те, що, якщо вектор вхідних сигналів нормалізований, вагові вектори також будуть нормалізованими.

При контрольованому навчанні правило Хебба (1.15) використовує замість фактичного значення виходу y потрібне y^*

$$W_p = \gamma y_p^* x_p^T. \qquad (1.25)$$

Особливості цього правила розглянемо для випадку γ=1. Помножимо обидві частини (1.25) ліворуч на x_p

$$W_p x_p = y_p^* x_p^T x_p \tag{1.26}$$

З (1.26) витікає, що при ортонормованих вхідних сигналах, $\|x_p\|^2 = 1$,матриця (1.25) відразу ж дає бажаний вихідний сигнал

$$W_p x_p = y_p^* \tag{1.27}$$

Оскільки мережа повинна зберігати усі P образів, матриця мережі W визначається згідно (1.16). При наданні мережі вхідного образу x_p вона повинна видати асоційований з x_p вихідний сигнал y_p. Фактичний вихідний сигнал при наданні деякого образу x_k обчислюється так:

$$y_k = W x_k = \left(\sum_{p=1}^{p} \mathrm{y_p^* x_p^T}\right) x_k = \mathrm{y_k^* x_k^T x_k} + \sum_{p\neq 1}^{p} \mathrm{y_p^* x_p^T x_k}\,. \tag{1.28}$$

Якщо вхідні сигнали ортонормовані

$$\mathrm{x_k^T x_k} = \begin{cases} 1 & \text{при} \quad p = k \\ 0 & \text{при} \quad p \neq k \end{cases} \tag{1.29}$$

і ортогональні, тоді $y_k y_k^*$, мережа безпомилково відтворює асоційований з заданим вхідним образом вихідний. Якщо вхідні вектори не ортонормовані, вихідний сигнал міститиме деяку помилку, величина якої залежить від міри їх лінійної залежності або корельованості. Це справедливо незалежно від того, співпадає або не співпадає розмірність векторів входу і виходу мережі.

2 ПРАКТИЧНА ЧАСТИНА ЛАБОРАТОРНОЇ РОБОТИ

"Дослідження процесу розпізнавання друкарських символів за допомогою штучних нейронних мереж"

2.1 Мета лабораторної роботи

Вивчення способів і засобів розпізнавання друкарських символів. Придбання практичних навичок побудови і застосування штучних нейронних мереж різних парадигм для вирішення завдання розпізнавання друкарських символів.

2.2 Технічне забезпечення лабораторної роботи

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

2.3 Хід виконання лабораторної роботи

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

Напередодні лабораторної роботи:

1) вивчити завдання і порядок виконання лабораторної роботи,
2) вивчити теоретичні відомості по лабораторній роботі.

Під час лабораторної роботи:

1) отримати допуск до проведення лабораторної роботи, відповівши на питання викладача по теоретичній частині досліджень при проведенні лабораторної роботи,
2) написати програмний код по кожному пункту завдання і проаналізувати результати, отримані при виконанні програмних кодів,
3) зробити висновки по лабораторній роботі.

Після лабораторної роботи:

1) підготувати звіт про лабораторну роботу відповідно до приведених в цьому методичному посібнику вимог,
2) захистити звіт перед викладачем, відповівши на його запитання по практичному проведенню досліджень при виконанні лабораторної роботи.

2.4 Завдання на лабораторну роботу і порядок їх виконання

Постановка завдання.

Здійснити розпізнавання друкарських символів за допомогою штучних нейронних мереж різної архітектури з різними функціями помилки, активації і алгоритмами навчання.

Завдання 1.

Здійснити розпізнавання друкарських символів з застосуванням штучної нейронної мережі прямого поширення сигналу з різною кількістю прихованих шарів, з навчанням за алгоритмом зворотного поширення помилки, при різних функціях активації нейронів, при різних видах помилки і з застосуванням конкуруючих нейронів.

Здійснимо очищення командного рядка і пам'яті, закриємо усі відкриті графічні вікна і запустимо лічильник часу:

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

Графічні символи (букви А, Б, В, Г, Д, Е в прямому, похилому і напівжирному зображенні – всього 18 букв, і цифри від 0 до 9 в прямому зображенні – всього 10 цифр) сформовані заздалегідь у вигляді файлів з графічним розширенням *.bmp. За допомогою команди imread зробимо перетворення файлів зображень букв графічного формату в числові масиви. Результатом будуть файли формату uint8 з розмірністю 20×20×3. Остання розмірність пов'язана зі збереженням кольору. Перевизначимо масиви зробивши їх двовимірними, тобто у відтінках сірого (що часто виходить в результаті сканування текстів), і перетворюваний з формату uint8 у формат double для того, щоб мати можливість проводити математичні операції з вмістом отриманих числових масивів:

```
A_P=imread('A_P.bmp'); A_P=A_P(:,:, 1); A_P=double(A_P);
A_K=imread('A_K.bmp'); A_K=A_K(:,:, 1); A_K=double(A_K);
A_G=imread('A_G.bmp'); A_G=A_G(:,:, 1); A_G=double(A_G);
B_P=imread('B_P.bmp'); B_P=B_P(:,:, 1); B_P=double(B_P);
B_K=imread('B_K.bmp'); B_K=B_K(:,:, 1); B_K=double(B_K);
B_G=imread('B_G.bmp'); B_G=B_G(:,:, 1); B_G=double(B_G);
V_P=imread('V_P.bmp'); V_P=V_P(:,:, 1); V_P=double(V_P);
V_K=imread('V_K.bmp'); V_K=V_K(:,:, 1); V_K=double(V_K);
V_G=imread('V_G.bmp'); V_G=V_G(:,:, 1); V_G=double(V_G);
G_P=imread('G_P.bmp'); G_P=G_P(:,:, 1); G_P=double(G_P);
G_K=imread('G_K.bmp'); G_K=G_K(:,:, 1); G_K=double(G_K);
```

```
G_G=imread('G_G.bmp'); G_G=G_G(:,:, 1); G_G=double(G_G);
D_P=imread('D_P.bmp'); D_P=D_P(:,:, 1); D_P=double(D_P);
D_K=imread('D_K.bmp'); D_K=D_K(:,:, 1); D_K=double(D_K);
D_G=imread('D_G.bmp'); D_G=D_G(:,:, 1); D_G=double(D_G);
E_P=imread('E_P.bmp'); E_P=E_P(:,:, 1); E_P=double(E_P);
E_K=imread('E_K.bmp'); E_K=E_K(:,:, 1); E_K=double(E_K);
E_G=imread('E_G.bmp'); E_G=E_G(:,:, 1); E_G=double(E_G);
```

Для контролю візуалізуємо один з отриманих числових масивів (рис. 2.1):

```
figure; imshow(E_G);
```

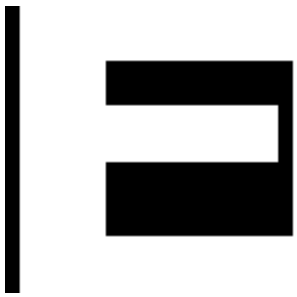


Рис. 2.1 – Візуалізований двовимірний числовий масив букви Е в напівжирному зображенні

У системі комп'ютерної математики МатЛаб на входи штучної нейронної мережі прямого поширення сигналу для її навчання надаються вектори. Навчальний вектор є числовим масивом у вигляді вектору-стовпця. Оскільки навчатися мережа буде буквам, тоді кожну з них треба перетворити у вектор-стовпець. Після чого потрібно зібрати вектори-стовпці, які вийшли, в масив, в якому кожен стовпець представлятиме одну букву і надати цей масив штучній нейронній мережі як навчальний. Виконаємо описані дії.

Визначимо необхідну розмірність масивів:

```
[str, stb]=size(A_P)
razm=str*stb
```

Отримаємо для кожної букви (а вони однаковій розмірності) кількість рядків і стовпців і загальну кількість елементів в кожній букві:

str = 20 stb = 20 razm = 400

Застосувавши команду reshape перетворимо квадратні масиви букв розмірністю 20×20 у вектори-стовпці розмірністю 400×1:

```
AP=reshape(A_P, razm, 1);
BP=reshape(B_P, razm, 1);
VP=reshape(V_P, razm, 1);
GP=reshape(G_P, razm, 1);
DP=reshape(D_P, razm, 1);
```

```
EP=reshape(E_P, razm, 1);
AK=reshape(A_K, razm, 1);
BK=reshape(B_K, razm, 1);
VK=reshape(V_K, razm, 1);
GK=reshape(G_K, razm, 1);
DK=reshape(D_K, razm, 1);
EK=reshape(E_K, razm, 1);
AG=reshape(A_G, razm, 1);
BG=reshape(B_G, razm, 1);
VG=reshape(V_G, razm, 1);
GG=reshape(G_G, razm, 1);
DG=reshape(D_G, razm, 1);
EG=reshape(E_G, razm, 1);
```

Об'єднаємо всі вектори-стовпці букв в єдиний масив за допомогою команди cat, тим самим створивши навчальну множину (масив):

```
P=cat(2, AP, BP, VP, GP, DP, EP, AK, BK, VK, GK, DK, EK, AG, BG, VG, GG, DG, EG);
```

Структура навчального масиву P наступна: кожен стовпець представляє певну букву в певному зображенні (всього 18 стовпців), а рядки є кодом кожної з букв (400 рядків), тому розмірність його 400×18. Буква А різних зображень знаходиться в 1-м, 7-м і в 13-м стовпцях, буква Б відповідно в 2-м, 8-м і в 14-м стовпцях і так далі.

Сформуємо цільову функцію (масив). Розпізнаються 6 букв. Це означає, що нейронна мережа повинна мати 6 виходів. Найбільша активність одного з виходів свідчитиме про те, що розпізнана відповідна буква. Але кожна з букв має три зображення. Загальне число букв – 18. Тому в цільовій функції необхідно вказати які з різних зображень букв відносяться до кожної з букв, тобто потрібно вказати місця розташування однакових букв в різних зображеннях. Нехай рядки відповідають буквам: перший рядок – букві А, другий рядок – букві Б, третій – букві В, четвертий – букві Г, п'ятий – Д, шостий, – Е. Нехай стовпці відповідають 18-ти буквам в різних зображеннях. У кожному з стовпців треба помітити однакові букви в різних зображеннях. Помітити можна проставляючи одиницю в кожному стовпці для відповідної букви. В результаті отримаємо наступну структуру цільового масиву T (обведено потовщеною лінією):

			Накреслення букв																	
		Букви	А	Б	В	Г	Д	Е	*А*	*Б*	*В*	*Г*	*Д*	*Е*	**А**	**Б**	**В**	**Г**	**Д**	**Е**
Вихідні нейрони	1	А	**1**	0	0	0	0	0	**1**	0	0	0	0	0	**1**	0	0	0	0	0
	2	Б	0	**1**	0	0	0	0	0	**1**	0	0	0	0	0	**1**	0	0	0	0
	3	В	0	0	**1**	0	0	0	0	0	**1**	0	0	0	0	0	**1**	0	0	0
	4	Г	0	0	0	**1**	0	0	0	0	0	1	0	0	0	0	0	**1**	0	0
	5	Д	0	0	0	0	**1**	0	0	0	0	0	**1**	0	0	0	0	0	**1**	0
	6	Е	0	0	0	0	0	**1**	0	0	0	0	0	**1**	0	0	0	0	0	**1**

Завдання побудови цільового масиву такої структури вирішується з застосуванням функції побудови одиничної діагональної матриці:

```
T=[eye(6) eye(6) eye(6)];
```

В результаті проведених операцій отримали навчальний масив P і цільовий масив T.

Створимо мережу прямого поширення сигналу з одним прихованим шаром для СКМ 9[1]:

```
net=newff(P, T, 8,{'tansig ',' purelin''}, trainlm');
```

Для мережі з одним прихованим шаром в атрибутивах вказані відповідно навчальний масив P, цільовий масив T, кількість нейронів в прихованому шарі – 8, функції активації прихованого і вихідного шарів tansig і purelin, вид алгоритму зворотного поширення помилки trainlm[2].

В результаті отримаємо архітектуру мережі, відбиту на (рис. 2.2):

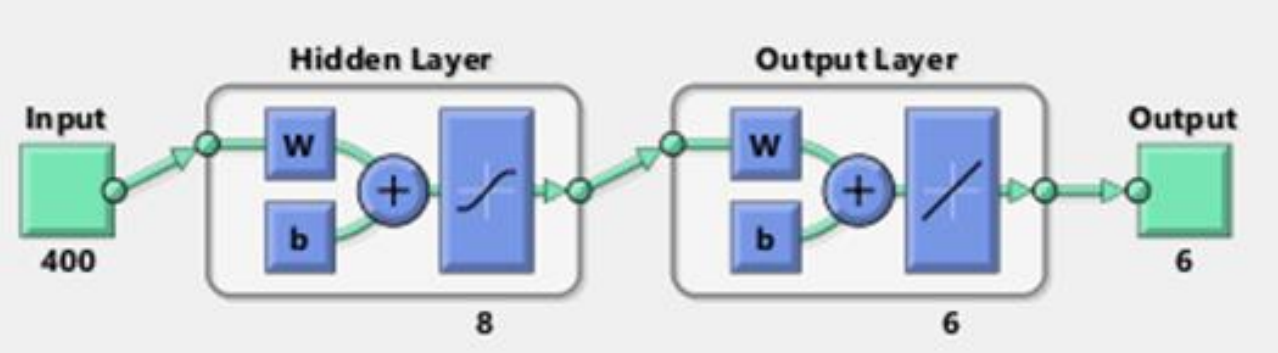


Рис. 2.2 – Архітектура створеної штучної нейронної мережі прямого поширення сигналу з одним прихованим шаром

Під час проведення дослідження необхідно змінювати число нейронів (щоб уникнути затягування рішення небажано робити більше 15...20 нейронів), змінювати функції активації і вид алгоритму зворотного поширення помилки.

---

[1] для СКМ 6.5: net=newff(minmax(P),[8,18],{'tansig' 'tansig' 'purelin'},'trainlm', 'learngdm','mse'); для мережі прямого поширення сигналу з двома прихованими шарами для СКМ 9: net=newff(P,T,[8 8],{'tansig','tansig','purelin'},'trainlm');

[2] Для мережі з двома прихованими шарами в атрибутивах добавлені кількість нейронів в другому прихованому шарі та функція активації для нього.

Для СКМ 6.5 задаються мінімальне та максимальне значення даних в навчальному масиві, кількість нейронів в прихованих шарах – 8 і 18 (цим задається число прихованих шарів), функції активації для прихованих і вихідного шарів tansig, tansig, purelin, навчальна функція зворотного поширення помилки trainlm, алгоритм налаштування вагів та зміщень learngdm, функція оцінки якості роботи мережі mse.

При створенні мережі прямого поширення задаються: функція оцінки якості роботи мережі, значення помилки і гранична кількість епох навчання:

```
net.PerformFcn='mse';          % змінювати на sse
net.trainParam.goal=0.01;   % змінювати на 0.1   0.001
net.trainParam.epochs=100;
```

У разі потреби можна провести ініціалізацію вагів і зміщень випадковим чином:

```
net.inputWeights{1,1}.initFcn='rands';
net.biases{1}.initFcn='rands';
```

або нулями:

```
net=init(net);
```

Проведемо процедуру навчання мережі:

```
net=train(net, P, T);
```

Мережа прямого поширення навчена. Результат навчання відображений на (рис. 2.3):

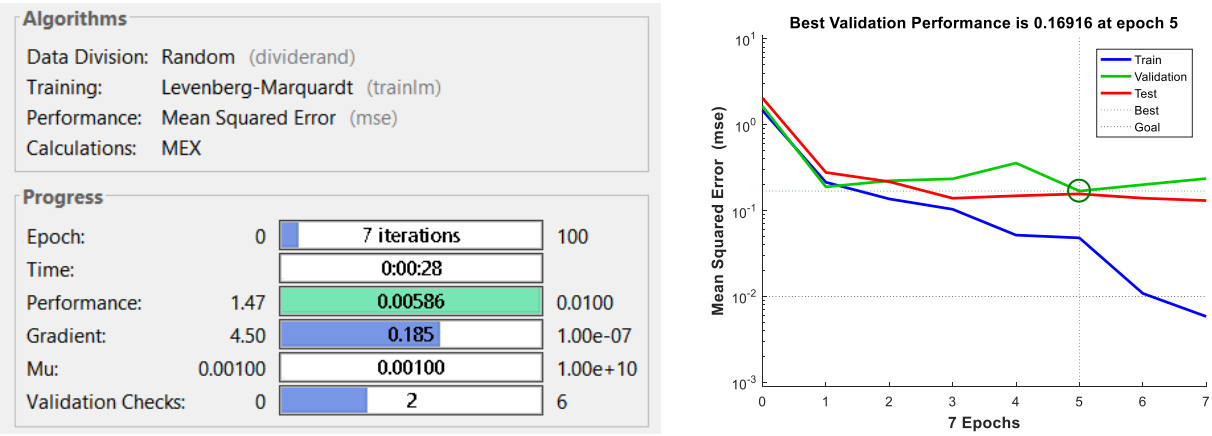


Рис. 2.3 – Результат навчання створеної штучної нейронної мережі прямого поширення сигналу з одним прихованим шаром

Тепер необхідно перевірити якість роботи вивченої мережі.

Застосуємо навчену мережу для розпізнавання векторів з навчальної множини, тобто букв, на яких мережа навчалася. Букви треба міняти, аналізуючи отримувані результати. Для прикладу розпізнаємо букву А напівжирного зображення:

```
y1=sim(net, AG) % змінювати букви, дивитися результат розпізнавання
```

Результат: y1 =

```
   1.0558
   0.5563
  -0.3963
  -0.3220
   0.4872
   0.3967
```

Найбільше значення в першому рядку, який відповідає букві А. Буква розпізнавана правильно. Застосуємо конкуренцію нейронів вихідного шару:

```
y2=find(compet(y1)==1)
```

Результат: y2 = 1

тобто найбільший відгук має перший нейрон, відповідний букві А. Це відповідає результату y1.

Розпізнаємо відразу усі букви, яким навчалася мережа, тобто розпізнаємо всю навчальну множину з застосуванням конкуренції нейронів у вихідному шарі:

```
y3=sim(net, P)
for k=1:18
Y(k)=find(compet(y3(:, k))==1);
end;
Y
```

Результат: Y =

1 3 3 4 5 6 1 2 3 4 5 6 1 6 3 3 5 6

а повинно бути відповідно до навчального масиву P і цільовим масивом T:

1 2 3 4 5 6 1 2 3 4 5 6 1 2 3 4 5 6

Видно помилки в розпізнаванні букв Б прямого і напівжирного зображення і букви Г напівжирного зображення. Звернути увагу на те, що при кожному новому запуску програми результат виходить різний.

Вимкнемо лічильник часу:

```
toc
```

Зробити висновки за виконаним Завданням 1.

Завдання 2.

Здійснити розпізнавання друкарських символів з застосуванням штучної нейронної мережі прямого поширення сигналу з різною кількістю прихованих шарів, з навчанням за алгоритмом зворотного поширення помилки, при різних функціях активації нейронів, при різних видах помилки і з застосуванням конкуруючих нейронів.

Відмінність у виконанні Завдання 2 від Завдання 1 в тому, що в Завданні 1 початкові дані представляють 8-ми розрядні числа в діапазоні значень від 0 до 255, а в Завданні 2 ті ж 8-розрядні числа знаходяться в діапазоні 0 або 1. Тому Завдання 2, окрім іншого, є дослідженням впливу форми представлення початкових даних на роботу штучної нейронної мережі прямого поширення сигналів.

Для цього завдання початкова частина коду виглядає таким чином:

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

введення даних :

```
load APbit; load AKbit; load AGbit;
load BPbit; load BKbit; load BGbit;
load VPbit; load VKbit; load VGbit;
load GPbit; load GKbit; load GGbit;
load DPbit; load DKbit; load DGbit;
load EPbit; load EKbit; load EGbit;
```

Як приклад відображення вхідних даних відобразимо ту ж букву, що і в Завданні 1 (рис. 2.4):

```
figure; imshow(EGbit);
```

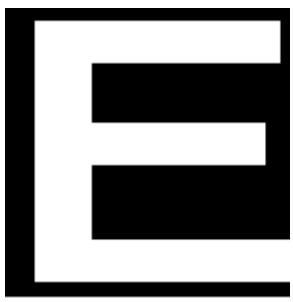


Рис. 2.4 – Візуалізований двовимірний бітовий числовий масив букви Е в напівжирному зображенні

Перетворення букв для надання мережі:

```
[str, stb]=size(APbit);
razm=str*stb;
 AP=reshape(APbit, razm, 1);
BP=reshape(BPbit, razm, 1);
VP=reshape(VPbit, razm, 1);
GP=reshape(GPbit, razm, 1);
DP=reshape(DPbit, razm, 1);
EP=reshape(EPbit, razm, 1);
AK=reshape(AKbit, razm, 1);
BK=reshape(BKbit, razm, 1);
VK=reshape(VKbit, razm, 1);
GK=reshape(GKbit, razm, 1);
DK=reshape(DKbit, razm, 1);
EK=reshape(EKbit, razm, 1);
AG=reshape(AGbit, razm, 1);
BG=reshape(BGbit, razm, 1);
VG=reshape(VGbit, razm, 1);
GG=reshape(GGbit, razm, 1);
DG=reshape(DGbit, razm, 1);
EG=reshape(EGbit, razm, 1);
```

Подальший хід дослідження співпадає з ходом Завдання 1.

Зробити висновки за виконаним Завданням 2.

Завдання 3.

Здійснити розпізнавання друкарських символів з застосуванням штучної нейронної мережі з зворотними зв'язками (мережа Хопфілда), порівняти результати розпізнавання з результатами, отриманими за допомогою мережі прямого поширення сигналу без зворотних зв'язків. Здійснити зашумлення символів і провести їх розпізнавання мережею Хопфілда.

Здійснимо очищення командного рядка і пам'яті, закриємо усі відкриті графічні вікна і запустимо лічильник часу:

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

Проведемо введення даних відповідно до міркувань, вказаних в Завданні 1:

```
A_P=imread('A_P.bmp'); A_P=A_P(:,:, 1); A_P=double(A_P);
A_K=imread('A_K.bmp'); A_K=A_K(:,:, 1); A_K=double(A_K);
A_G=imread('A_G.bmp'); A_G=A_G(:,:, 1); A_G=double(A_G);
B_P=imread('B_P.bmp'); B_P=B_P(:,:, 1); B_P=double(B_P);
B_K=imread('B_K.bmp'); B_K=B_K(:,:, 1); B_K=double(B_K);
B_G=imread('B_G.bmp'); B_G=B_G(:,:, 1); B_G=double(B_G);
V_P=imread('V_P.bmp'); V_P=V_P(:,:, 1); V_P=double(V_P);
V_K=imread('V_K.bmp'); V_K=V_K(:,:, 1); V_K=double(V_K);
V_G=imread('V_G.bmp'); V_G=V_G(:,:, 1); V_G=double(V_G);
G_P=imread('G_P.bmp'); G_P=G_P(:,:,1); G_P=double(G_P);
G_K=imread('G_K.bmp'); G_K=G_K(:,:,1); G_K=double(G_K);
G_G=imread('G_G.bmp'); G_G=G_G(:,:,1); G_G=double(G_G);
D_P=imread('D_P.bmp'); D_P=D_P(:,:,1); D_P=double(D_P);
D_K=imread('D_K.bmp'); D_K=D_K(:,:,1); D_K=double(D_K);
D_G=imread('D_G.bmp'); D_G=D_G(:,:,1); D_G=double(D_G);
E_P=imread('E_P.bmp'); E_P=E_P(:,:,1); E_P=double(E_P);
E_K=imread('E_K.bmp'); E_K=E_K(:,:,1); E_K=double(E_K);
E_G=imread('E_G.bmp'); E_G=E_G(:,:,1); E_G=double(E_G);
```

Для контролю введення даних відобразимо графічно, наприклад, букву Е напівжирного зображення (див. рис. 2.1):

```
figure(1); imshow(E_G);
```

У системі комп'ютерної математики СКМ 9 на вхід мережі Хопфілда надаються дані у вигляді +1 і –1. Тому потрібно додаткове перетворення букв в набір +1 і –1 для надання мережі. Операція нескладна і полягає в тому, що усім значенням амплітуд в пікселях букви більше або рівних деякому значенню ставиться у відповідність –1, а меншим деякого значення ставиться у відповідність +1. Таким чином, фон, на якому відображується буква, укладатиметься з +1 і буде білим, а накреслена буква укладатиметься з –1 і буде чорною. Ця операція проводиться з усіма буквами, використовуваними для навчання мережі, і при розпізнаванні. Заздалегідь визначається розмірність букв і далі застосовується оператор циклу для заміни значень амплітуд в пікселях усіх букв:

```
[str,stb]=size(A_P);
razm=str*stb;

for m=1:str
   for n=1:stb
      if A_P(m,n)>=120
         A_P(m,n)=-1;
      else A_P(m,n)=1;
      end;
   end;
end;
```

```
for m=1:str
   for n=1:stb
      if A_K(m,n)>=120
         A_K(m,n)=-1;
      else A_K(m,n)=1;
      end;
   end;
end;

for m=1:str
   for n=1:stb
      if A_G(m,n)>=120
         A_G(m,n)=-1;
      else A_G(m,n)=1;
      end;
   end;
end;

for m=1:str
   for n=1:stb
      if B_P(m,n)>=120
         B_P(m,n)=-1;
      else B_P(m,n)=1;
      end;
   end;
end;

for m=1:str
   for n=1:stb
      if B_K(m,n)>=120
         B_K(m,n)=-1;
      else B_K(m,n)=1;
      end;
   end;
end;

for m=1:str
   for n=1:stb
      if B_G(m,n)>=120
         B_G(m,n)=-1;
      else B_G(m,n)=1;
      end;
   end;
end;

for m=1:str
   for n=1:stb
      if V_P(m,n)>=120
```

```
            V_P(m,n)=-1;
        else V_P(m,n)=1;
        end;
    end;
end;

for m=1:str
    for n=1:stb
        if V_K(m,n)>=120
            V_K(m,n)=-1;
        else V_K(m,n)=1;
        end;
    end;
end;

for m=1:str
    for n=1:stb
        if V_G(m,n)>=120
            V_G(m,n)=-1;
        else V_G(m,n)=1;
        end;
    end;
end;

for m=1:str
    for n=1:stb
        if G_P(m,n)>=120
            G_P(m,n)=-1;
        else G_P(m,n)=1;
        end;
    end;
end;

for m=1:str
    for n=1:stb
        if G_K(m,n)>=120
            G_K(m,n)=-1;
        else G_K(m,n)=1;
        end;
    end;
end;

for m=1:str
    for n=1:stb
        if G_G(m,n)>=120
            G_G(m,n)=-1;
        else G_G(m,n)=1;
        end;
    end;
```

```
end;

for m=1:str
   for n=1:stb
      if D_P(m,n)>=120
         D_P(m,n)=-1;
      else D_P(m,n)=1;
      end;
   end;
end;

for m=1:str
   for n=1:stb
      if D_K(m,n)>=120
         D_K(m,n)=-1;
      else D_K(m,n)=1;
      end;
   end;
end;

for m=1:str
   for n=1:stb
      if D_G(m,n)>=120
         D_G(m,n)=-1;
      else D_G(m,n)=1;
      end;
   end;
end;

for m=1:str
   for n=1:stb
      if E_P(m,n)>=120
         E_P(m,n)=-1;
      else E_P(m,n)=1;
      end;
   end;
end;

for m=1:str
   for n=1:stb
      if E_K(m,n)>=120
         E_K(m,n)=-1;
      else E_K(m,n)=1;
      end;
   end;
end;

for m=1:str
   for n=1:stb
```

```
      if E_G(m,n)>=120
         E_G(m,n)=-1;
      else E_G(m,n)=1;
      end;
   end;
end;
```

Для контролю перетворення візуалізуємо, для прикладу, числовий масив букви Е напівжирного зображення (рис. 2.5):

```
figure(2); imshow(E_G);
```

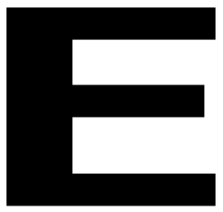


Рис. 2.5 – Візуалізований числовий масив букви Е в напівжирному зображенні для надання мережі Хопфілда (+1, –1)

Як і в Завданні 1 для мережі Хопфілда вимагається створити навчальну множину з усіх букв, заздалегідь сформувавши з кожної букви вектор-стовпець розмірністю 400×1, а потім об'єднавши їх в єдиний масив розмірністю 400×18 для векторного вирішення завдання мережею:

```
AP=reshape(A_P,razm,1);
BP=reshape(B_P,razm,1);
VP=reshape(V_P,razm,1);
GP=reshape(G_P,razm,1);
DP=reshape(D_P,razm,1);
EP=reshape(E_P,razm,1);
AK=reshape(A_K,razm,1);
BK=reshape(B_K,razm,1);
VK=reshape(V_K,razm,1);
GK=reshape(G_K,razm,1);
DK=reshape(D_K,razm,1);
EK=reshape(E_K,razm,1);
AG=reshape(A_G,razm,1);
BG=reshape(B_G,razm,1);
VG=reshape(V_G,razm,1);
GG=reshape(G_G,razm,1);
DG=reshape(D_G,razm,1);
EG=reshape(E_G,razm,1);

P=cat(2,AP,BP,VP,GP,DP,EP,AK,BK,VK,GK,DK,EK,AG,BG,VG,GG,DG,EG);
```

Навчальний масив P для надання мережі Хопфілда створений.

Здійснимо створення і навчання мережі Хопфилда:

```
net=newhop(P);
```

Застосуємо навчену мережу Хопфілда для розпізнавання усієї навчальної множини:

```
[Y]=sim(net, 18,[],P);
```

В результаті в Y буде повернений масив усіх букв розмірністю 400×18, який відобразити відразу увесь неможливо, хіба лише в числовому вигляді, яка не має практичного сенсу.

Сенс має відображення результатів розпізнавання по кожній букві. Наприклад, букву Г напівжирного зображення після її перетворення з вектору-стовпця у формат відображення розмірністю 20×20 (рис. 2.6):

```
[Y1]=sim(net,1,[],GG);
bukva1=reshape(Y1,str,stb);
figure(3); imshow(bukva1);
```

Рис. 2.6 – Візуалізований числовий масив розпізнаною мережею Хопфілда букви Г в напівжирному зображенні

На рисунку видно практично ідеальний результат розпізнавання букви, на якій, у тому числі, була навчена мережа Хопфілда.

Порівняйте результати розпізнавання друкарських символів мережею прямого поширення сигналів і мережею Хопфилда. Зробіть висновки.

Здійснимо зашумлення друкарських символів, наприклад, букву Е прямого зображення. Це може статися насправді при неякісному скануванні тексту. Зашумляти будемо шляхом формування масиву, аналогічного за розміром вектору-стовпцю букви, з нормальним (гаусовим) законом розподілу і накладанням вектору-стовпця букви на вектор-стовпець шуму. Максимальне значення амплітуди випадкових викидів досягає +1, тобто амплітуда шумових викидів порівнянна з амплітудою вхідних даних. При цьому є можливість управляти значеннями шумових викидів змінюючи числові значення

(помічений коефіцієнт в коді) повернених даних функції randn (рис. 2.7). При цьому коефіцієнт 0.0 відповідає відсутності шуму :

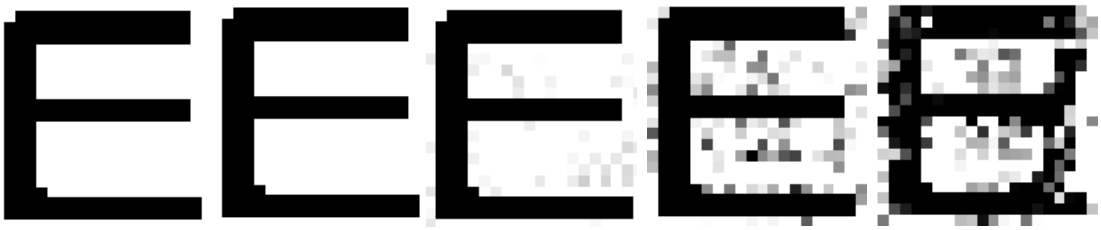
```
EP=EP+1.8*randn(size(EP))
bukva2=reshape(EP,str,stb);
figure(4); imshow(bukva2);
```

а *б* *в* *г* *д*

Рис. 2.7 – Зашумлення букви Е: *а* – 0.0, *б* – 0.1, *в* – 0.5, *г* – 1.5, *д* – 2.5

Здійснимо розпізнавання навченою штучною нейронною мережею Хопфілда зашумленої букви Е прямого зображення і відобразимо її графічно (рис. 2.8):

```
[Y2]=sim(net,1,[],EP);
bukva3=reshape(Y2,str,stb);
figure(5); imshow(bukva3);
```



а *б* *в* *г* *д*

Рис. 2.8 – Результат розпізнавання навченою штучною нейронною мережею Хопфілда зашумленої букви Е:
а – 0.0, *б* – 0.1, *в* – 0.5, *г* – 1.5, *д* – 2.5

Вимкнемо лічильник часу:

```
toc
```

Зробіть висновки за виконаним Завданням 3.

Завдання 4.

Вивчити властивості штучної нейронної мережі зі зворотними зв'язками (мережі Хопфілда).

Мережа Хопфілда має чудову властивість – запам'ятовувати. Причому пам'яттю асоціативною. Тобто вона реалізує окрім відомих властивостей штучних нейронних мереж, таких як навчання, узагальнення, застосовність, ще і властивість абстрагування. Полягає ця властивість в тому, що на основі спотворених образів, яка пред'являються, мережа породжує ідеальний образ, який їй жодного разу не був пред'явлений. Тобто. мережа витягає суть з наданих їй даних і автоматично цю суть зберігає, в чому і проявляється здатність мережі Хопфілда запам'ятовувати. При будь-якому збудженні мережа видає збережений і запам’ятований образ. Якщо мережу навчити декільком образам, тоді при відповідному збудженні вона їх генеруватиме. Це добре видно при виконанні Завдання 3.

Для дослідження властивостей запам'ятовування і абстрагування мережі Хопфілда на прикладі цифр виконаємо наступні операції.

Здійснимо очищення командного рядка і пам'яті, закриємо усі відкриті графічні вікна і запустимо лічильник часу:

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

Виконаємо введення даних (цифри можна вводити різні) і введеній цифрі присвоїмо деякий узагальнений ідентифікатор obraz:

```
load C1bit;  % можна і інші цифри
obraz=C1bit;
```

Відобразимо графічно введену цифру (рис. 2.9), яка виконуватиме роль ідеального образу:

```
figure(1); imshow(obraz);
```

Рис. 2.9 – Ідеальний образ цифри 1

Визначимо параметри даних:

```
[str, stb]=size(obraz);
razm=str*stb;
```

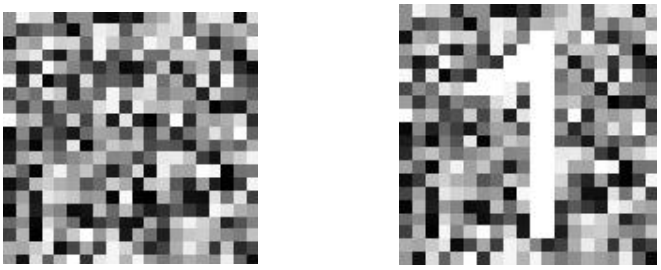
str = 20 stb = 20 razm = 400

Для дослідження властивості абстрагування вимагається створити деяку кількість (вибірку) спотворених шумом образів ідеального образу введеної цифри і надати вибірку мережі для навчання. При цьому включення в навчальну вибірку ідеального образу не допускається. Формувати образи шуму s і спотворені шумом образи введеної цифри CCC1 будемо в циклі у вигляді двовимірного масиву зображення: формуємо образ шуму, накладаємо його на ідеальний образ цифри, запам'ятовуємо, і, таким чином, отримуємо потрібну кількість (vyborka) спотворених шумом образів ідеального образу цифри.

```
vyborka=30; % змінювати від 2 до 50...100
for m=1:str
   for n=1:stb
      for k=1:vyborka
        m, n, k)=1.0*unifrnd(0,1);   % застосувати unifrnd normrnd
        CC1(m, n, k)=obraz(m, n)+s(m, n, k); % міняти рів. шуму від 1
      end;
   end;
end;
```

Для контролю візуалізуємо останні з сформованих образів шуму і спотворені цим шумом ідеальні образи цифри (рис. 2.10):

```
figure(2); imshow(s(:,:,vyborka));
figure(3); imshow(CCC1(:,:,vyborka));
```



а *б*

Рис. 2.10 – Образ шуму (*а*),
спотворений шумом ідеальний образ цифри 1 (*б*)

Оскільки мережа Хопфілда вимагає в якості вхідних даних числа +1 і –1, проведемо операцію перетворення даних у вигляд, придатний для подачі в мережу. Якщо амплітуда в пікселі рівна або перевищує значення 0.7, тоді у відповідний піксель навчального образу CC1 записуємо +1, якщо менше 0.7 – тоді записується –1 (рис. 2.11):

```
for m=1:str
   for n=1:stb
```

```
      for k=1:vyborka
         if CCC1(m,n,k)>=0.7
            CC1(m,n,k)=1;
         else CC1(m,n,k)=-1;
         end;
      end;
   end;
end;
figure(4); imshow(CC1(:,:,vyborka));
```

Рис. 2.11 – Останній з сформованих навчальних образів у вигляді спотвореного шумом ідеального образу цифри 1

Мережа Хопфілда, як і мережа прямого поширення, вимагає набору навчальних векторів у вигляді векторів-стовпців. Тому всі зашумлені навчальні образи представимо у вигляді векторів-стовпців і об'єднаємо їх вздовж стовпців. Результатом стане навчальний масив C1 розмірністю 400×30, де 400=20×20 є вектори-стовпці зашумлених цифр, а 30 → vyborka або кількість сформованих зашумленых образів цифри:

```
for k=1:vyborka
   C1(:,k)=reshape(CC1(:,:,k),razm,1);
end;
% C1(:,vyborka)
```

Навчальний масив сформований. Надаємо його мережі Хопфілда для навчання:

```
net=newhop(C1);
```

Мережа навчена. Надаємо мережі зашумлений образ цифри з середини сформованого навчального масиву (рис. 2.12) для перевірки запам'ятовування і факту породження ідеального образу (рис. 2.13):

```
Z0=reshape(C1(:,round(vyborka/2)),str,stb);
figure(5); imshow(Z0);
```

Рис. 2.12 – Середній з сформованих навчальних образів у вигляді спотвореного шумом ідеального образу цифри 1

```
Y1=sim(net,C1(:,round(vyborka/2)));
Z1=reshape(Y1,str,stb);
figure(6); imshow(Z1);
```

Рис. 2.13 – Реакція навченої мережі Хопфілда на дію середнього з сформованих навчальних образів у вигляді спотвореного шумом ідеального образу цифри 1

Результат – породжений і запам'ятований ідеальний образ цифри 1, яка не надавалася мережі при її навчанні.

Незначна логічна обробка отриманого результату дозволяє поліпшити якість еталонного образу:

```
for m=1:str
   for n=1:stb
      if Z1(m,n)>=0.5
         Z(m,n)=1;
      else Z(m,n)=0;
      end;
   end;
end;
```

Відобразимо результат поліпшення отриманого образу (рис. 2.14):

```
figure(7); imshow(Z);
```

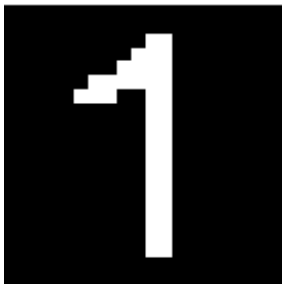


Рис. 2.14 – Результат поліпшення отриманого образу

Проте, образ, який запам'ятований, породжується не лише образом, на якому мережа навчалась.

Перевіримо це твердження. Подаємо на вхід мережі іншу цифру, наприклад, двійку:

```
load C2bit;
obraz2=C2bit;
```

Відобразимо її (рис. 2.15):

```
figure(8); imshow(obraz2);
```

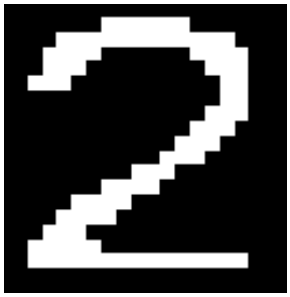


Рис. 2.15 – Інша дія на вході навченої на цифрі 1 мережі Хопфілда

```
CCC2=reshape(obraz2,razm,1);
for r=1:razm
   if CCC2(r)>=0.7
      CC2(r)=1;
   else CC2(r)=-1;
   end;
end;
CC2=CC2';
Y2=sim(net,CC2);
Z2=reshape(Y2,str,stb);
```

Відгук навченої на цифрі 1 мережі Хопфілда на дію у вигляді цифри 2 (рис. 2.16):

```
figure(9); imshow(Z2);
```

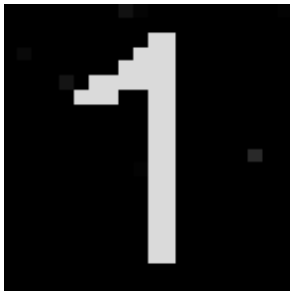


Рис. 2.16 – Вихід навченої на цифрі 1 мережі Хопфілда на дію у вигляді цифри 2

Вимкнемо лічильник часу:

toc

Дослідження проводити, спираючись на вказані в коді в коментарях інструкції.

Після закінчення досліджень зробити висновки за Завданням 4.

2.5 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 додаванням результатів виконання усіх операцій і команд.
4) Розділ: Висновки по лабораторній роботі.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Опишіть архітектуру мережі прямого поширення сигналу
2. Опишіть алгоритм зворотного поширення помилки
3. Охарактеризуйте алгоритм градієнтного спуску
4. Охарактеризуйте алгоритм Левенберга-Марквардта
5. Опишіть мережі з зворотними зв'язками
6. Охарактеризуйте особливості мереж з зворотними зв'язками
7. У чому полягають особливості мережі Хопфілда?
8. Опишіть архітектуру мережі Хопфілда
9. Як описується робота мережі Хопфілда з енергетичної точки зору?
10. Яким чином здійснюється запам'ятовування образу в мережі Хопфілда?
11. Яким чином здійснюється виклик образу в мережі Хопфілда?
12. Опишіть узагальнений алгоритм роботи мережі Хопфілда
13. Охарактеризуйте відмінності в роботі синхронної і несинхронної мережі Хопфілда
14. Охарактеризуйте відмінності в роботі дискретної і безперервної мережі Хопфілда
15. Опишіть правило Хебба

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

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. – 222 с.
2. Руденко О.Г., Бодянський Є.В. Штучні нейронні мережі: Навчальний посібник. – Харків: ТОВ „Компанія СМІТ”, 2005. – 408 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 редакторові.
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 листа.
4) Хід лабораторної роботи оформляється в одному розділі з нового листа.
5) Висновки по лабораторній роботі оформляються в одному розділі з нового листа.
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 та інформатики

ЗВІТ
про лабораторну роботу

ДОСЛІДЖЕННЯ ПРОЦЕСУ РОЗПІЗНАВАННЯ ДРУКАРСЬКИХ СИМВОЛІВ ЗА ДОПОМОГОЮ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

по дисципліні
ШТУЧНІ НЕЙРОННІ МЕРЕЖІ В ЗАДАЧАХ ОБРОБКИ ДАНИХ

Виконав(ла) студент(ка) групи К- 41
Іванов Петро Сидорович

(підпис студента)

Перевірив Перелигін Б.В.

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

Одесса – 2024