##### **Нейронні мережі в системах керування динамічними об'єктами**

##### **Технічне завдання**

### Харків

2005**Технічне завдання**

## **Зміст**

# Вступ

1. Підстава для розробки.
2. Призначення розробки.
3. Вимоги до програмного виробу.
   1. Вимоги до функціональних характеристик.
   2. Вимоги до надійності.
   3. Умови експлуатації.
   4. Вимоги до маркування і пакування.
   5. Вимоги до транспортування і збереження.
4. Вимоги до складу і параметрів технічних засобів.
5. Вимоги до програмної документації.
6. Стадії і етапи розробки.
7. Порядок контролю і приймання.

##### **Вступ**

Головна мета даного курсового проекту провести огляд проблеми нейронних мереж, основних напрямків розвитку нейрокібернетики і областей використання нейронних мереж особливо для керування динамічними системами. Програмний виріб є демонстрацією можливостей нейронних мереж і методики їх побудови. Це програма, що реалізує нейронну мережу Хебба, що навчається користувачем поданням ряду символьних зображень на її вхід. Після завершення навчання програма виконує розпізнавання символів відповідно до набору зображень, що вона вивчила. Даний програмний виріб орієнтований на комп'ютери серії ІBM PS сумісних ПЭВМ із операційною системою Windows 95, 98, Me, XP, NT.

**1. Підстава для розробки:**

Підставою для розробки є завдання на курсове проектування.

**2. Призначення розробки:**

Метою розробки є огляд напрямків науки нейрокібернетики, що займається проблемою розробки і використання нейронних мереж для систем керування динамічними процесами. А також написання демонстраційної програми, що програмно реалізує мережу Хебба для розпізнавання символів.

**3. Вимоги до програмного виробу:**

3.1. Вимоги до функціональних характеристик:

3.1.1. Програмний виріб повинен реалізувати нейронну мережу Хебба.

3.1.2. Програмний виріб повинен реалізувати зручний інтерфейс навчання нейронної мережі і процесу розпізнавання символів.

3.1.3. Має бути реалізована візуалізація процес навчання і розрахунку важелів зв’язків нейронів.

3.1.4. Має бути реалізована можливість збереження і завантаження результатів навчання нейронної мережі.

3.2. Вимоги до надійності:

Програмний виріб повинний стійко функціонувати в апаратному і програмному середовищі, обговореному в п.4

3.3. Умови експлуатації:

Умови експлуатації програмного виробу цілком визначаються умовами експлуатації застосованих технічних засобів і носіїв даних.

* 1. Вимоги до маркування і пакування :

Вимоги до маркування і пакування не висуваються.

* 1. Вимоги до транспортування і збереження :

Вимоги до транспортування і збереження не висуваються.

**4. Вимоги до складу і параметрів технічних засобів:**

Для функціонування програмного виробу необхідна ІBM AT/XT і PS/2 чи сумісна з нею, з наступними мінімальними системними характеристиками:

* OS Windows 95
* Pentium I 100 MHz
* 16 Mb RAM
* відеоадаптер будь-якого класу;
* комп’ютерна миша;
* клавіатура.

**5. Вимоги до програмної документації:**

- Пояснювальна записка;

- Опис програми;

- Текст програми;

1. **Стадії і етапи розробки.**

Розробка програми повинна вестись за наступним планом :

1. одержання завдання;
2. огляд літератури і вирішення поставлених задач, обгрунтування необхідності розробки;
3. розробка структури вхідних і вихідних даних, уточнення методів рішення завдання;
4. опис і розробка алгоритму рішення завдання;
5. розробка структури програми та остаточне визначення конфігурації технічних засобів;
6. розробка програми, іспити і налагодження;
7. розробка програмної документації.
8. **Порядок контролю і приймання.**

При прийманні програмного виробу перевіряється :

1. комплектність, зміст і оформлення програмної документації;
2. відповідність програмного виробу програмним характеристикам;
3. відповідність програмного виробу обговореному програмному та апаратному середовищу;
4. надійність функцінування.

**Опис програми**

## Зміст

1. Опис інтерфейсу програмного виробу
2. Опис змінних
3. Опис процедур і функцій

Список літератури

1. **Опис інтерфейсу програмного виробу**

Програмний виріб – це програма, розроблена в середі програмування Delphi, і є програмною моделлю нейронної мережі Хебба. Даний виріб дозволяє проводити розпізнання вхідного сигналу відповідно до чотирьох навчальних, а також відслідковувати процес навчання і функціонування нейронної мережі.

Запуск програми відбувається із натисканням на відповідному виконуваному файлі (.exe), після чого користувач може спостерігати вікно програми, зображене на рисунку 1.

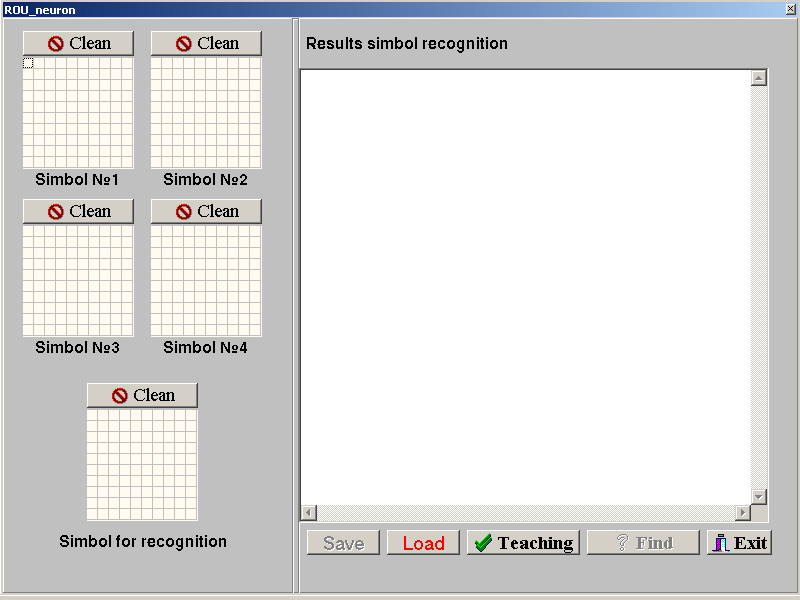


Рис. 1 Вигляд вікна програми на початку роботи

В лівій частині вікна ми можемо спостерігати п’ять полів, розбитих на 10×10 комірок – це нейрони, поля, куди заносяться користувачем, шляхом натискання курсором миші, навчальні зображення (верхні чотири поля) і сигнал, що розпізнається (п’яте окреме поле). Кожне поле супроводжується кнопкою "Clean", що дозволяє швидко очистити поле, якщо введено невірне зображення.

В правій частині вікна програми знаходиться інформаційне поле (Рис. 1), куди виводиться текстова інформація, щодо ініціалізації важелів зв’язків нейронів, стану нейронної мережі і результатів розпізнання. Це поле дозволяє з середини спостерігати за нейронною мережею.

Під інформаційним вікном знаходиться п'ять кнопок: "Save", "Load", "Teaching", "Find" і "Exit". При запуску програми кнопки "Save" і "Find" не активні.

На початку роботи користувач може задати навчальні зображення (Рис. 2), або скористатися кнопкою "Load", про яку розмова піде пізніше. Коли навчальні зображення задані, користувач має навчити нейронну мережу шляхом натискання на кнопку "Teaching", після чого мережа навчається і готова до розпізнання вхідних зображень. В інформаційному полі можна спостерігати стан нейронної мережі. Кнопки "Save" і "Find" активізувались (Рис. 2).

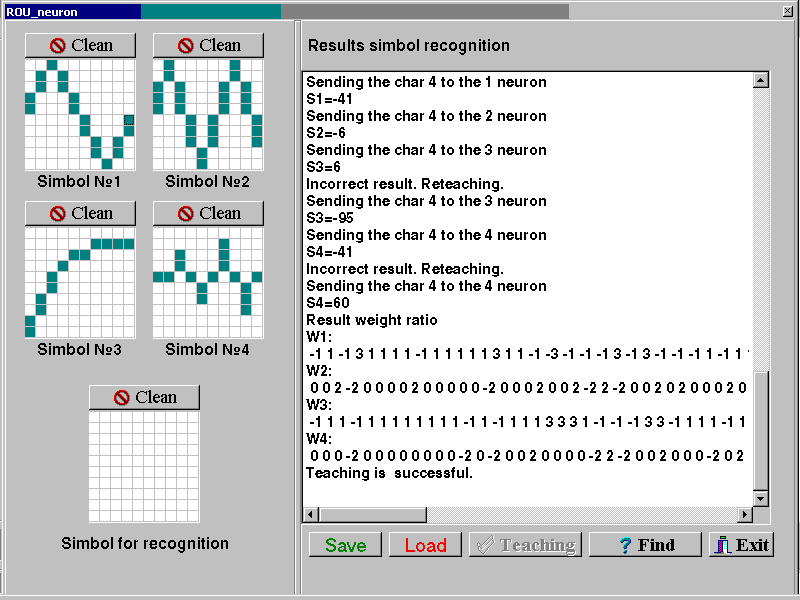


Рис. 2 Навчання нейронної мережі

Після навчання нейронної мережі користувач може задати вхідний сигнал в лівому нижньому полі і натиснути кнопку "Find". Нейронна мережа подасть вхідне зображення на кожний нейрон і ідентифікує вхідний сигнал відповідно до навчальних зображень, виділивши відповідну назву червоним кольором і повідомивши про це в інформаційному полі (Рис. 3).

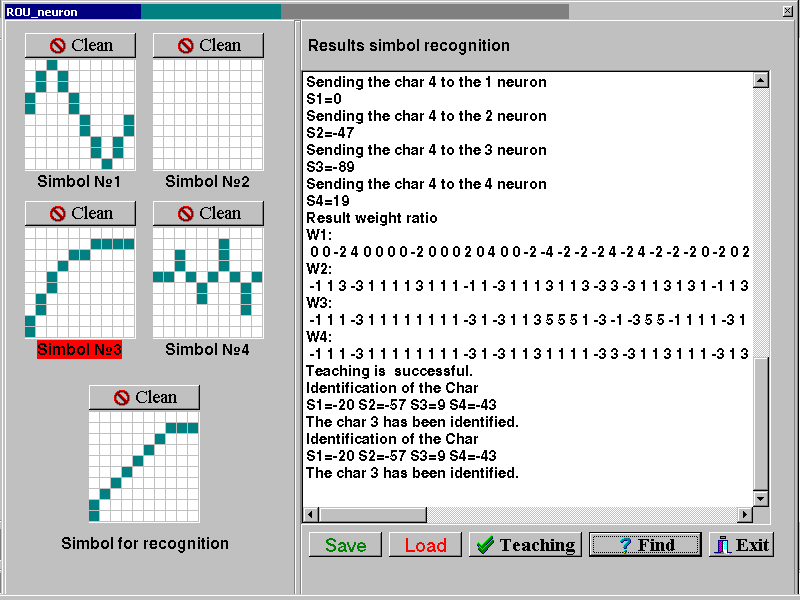
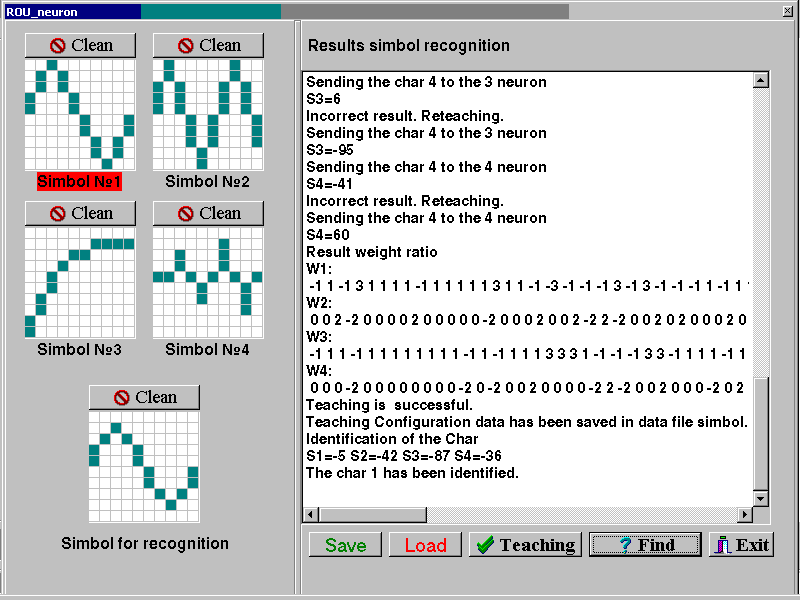


Рис. 3 Пошук відповідності вхідного сигналу еталонному

Слід зазначити, що процес вводу одних і тих же навчальних зображень при кожній роботі з нейронною мережею можна не виконувати. Для цього після навчання мережі треба зберегти її поточний стан і навчальні зображення у файлі шляхом натискання кнопки "Save" (Рис. 4).

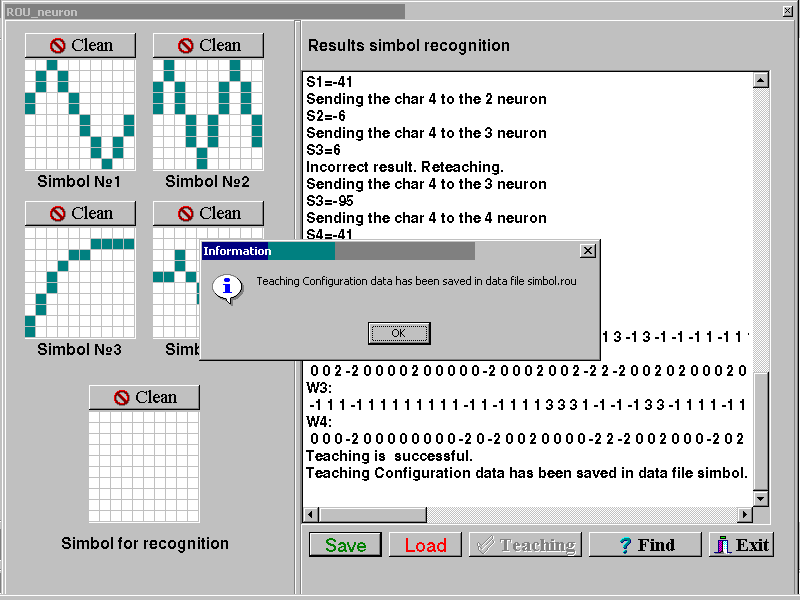


Рис. 4 Збереження поточного стану нейронної мережі

Збережені дані (навчальні зображення і стан нейронної мережі) можуть бути відновлені із файлу шляхом натискання на кнопку "Load" (Рис. 5).

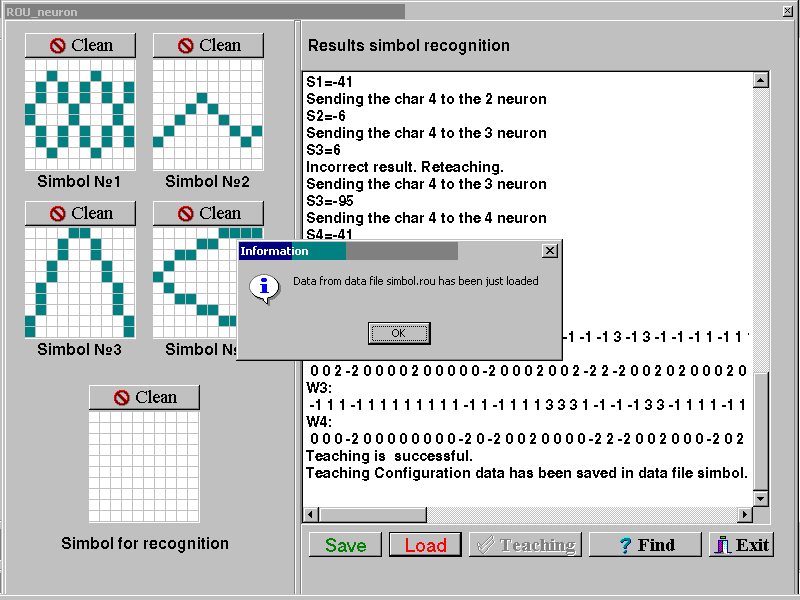


Рис. 5 Відновлення стану нейронної мережі із файлу

Програма повідомляє про щасливе завершення операції або помилку, якщо інформаційний файл відсутній чи неможливе його відкриття або збереження на диску.

Ці можливості і кнопки "Clean" (Рис. 6) значно полегшують роботу із програмою і є дуже зручними.

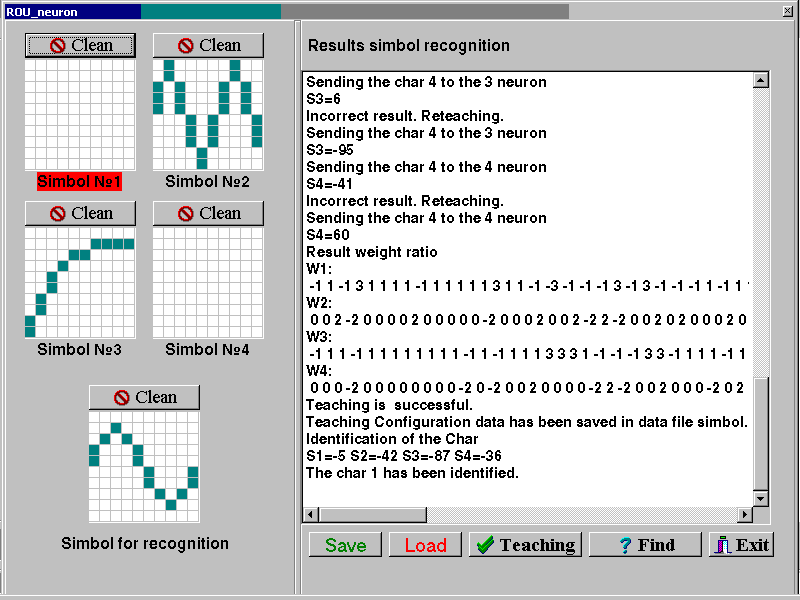


Рис. 6 Використання кнопок "Clean" для очищення полів нейронів

Завершення роботи програми виконується при натисканні на кнопку "Exit", або хрестик в верхньому правому куті вікна. Слід зберегти стан нейронної мережі і навчальні зображення якщо ви бажаєте повернутися до них пізніше, оскільки в іншому випадку всі напрацювання будуть втрачені.

1. **Опис змінних.**

|  |  |  |
| --- | --- | --- |
| Змінна | Тип | Призначення |
| Form1 | TForm1 | Змінна, що ідентифікує головну форму. Слугує для звернення до елементів форми |
| M | array [0..4] of  one\_element;  one\_element = array  [0..str\*stol + 4] of integer | Масив, що описує матриці важелів зв’язків нейронів, а також реакцію кожного нейрона на зображення |
| W | W\_type;  W\_type = array [0.3, 0..str\*stol] of integer | Масив, що описує навчальні зображення, які подаються на вхід нейронної мережі |
| S | S: S\_type;  S\_type = array [0..3] of integer | Масив результуючих сум реакції нейронів мережі на подане зображення |
| f | f: file of W\_type;  W\_type = array [0..3, str\*stol] of integer | Змінна, що ідентифікує файл, який складається із записів типу W\_type. Необхідна для роботи із файлом для збереження і завантаження стану нейронної мережі |

Вибір константних даних

yes = 'y';

no = 'n';

Слуговують для ідентифікації станів активності і неактивності комірок елементів TStringGrig, що описують нейрони.

str = 10;

stal = 10;

Задають розміри, точніше – кількість комірок одного нейрона, його роздільну здатність.

1. **Опис процедур і функцій.**

procedure BitBtn1Click (Sender: TObject);

Обробляє натискання на кнопку "Teaching". Виконує навчання нейронної мережі за поданими навчальними зображеннями доки мережа не почне реагувати на вхідні зображення вірно.

procedure BitBtn2Click (Sender: TObject);

Обробляє натискання на кнопку "Find". Реалізує пошук (ідентифікацію) відповідності поданого зображення із зображеннями які вивчила нейронна мережа шляхом послідовного подання зображення на нейрони мережі і порівняння реакції на нього.

procedure BitBtn5Click (Sender: TObject);

Обробляє натискання на кнопку "Save". Виконує збереження у файлі поточного стану мережі і навчальних зображень у спеціальному файлі.

procedure BitBtn4Click (Sender: TObject);

Обробляє натискання на кнопку "Load". Реалізує завантаження із спеціального файлу, збережені в ньому раніше важелі зв’язків нейронів нейронної мережі і навчальні зображення.

Інші методи є допоміжними і слугують для малювання форми або для забезпечення інтерфейсу і спеціальних дій і викликаються із основних функцій, що описані вгорі.

**Список літератури**

1. Баас Р., Фервай М., Гюнтер Х. Delphi 7: полное руководство. – К.: Издательская группа BHV, 2004. – 800 с.
2. Барский А.Б. Нейронные сети: распознавание, управление, принятие решений. – М.: Финансы и статистика, 2004. – 248 с.
3. Браславский И.Я., Зюзев А.М., Ишматов З.Ш., Аверьянов М.А., Барац Е.И., Костылев А.В. Синтез нейронного наблюдателя для асинхронного привода с прямым управлением моментом // Электротехника. – 2001. – № 12. – С. 31 – 34.
4. Кей А. Искусственные нейронные сети // Computerworld. – 2001. – № 9. – С. 34 – 38.
5. Клепиков В.Б., Сергеев С.А., Махотило К.В., Обруч И.В. Применение методов нейронных сетей и генетических алгоритмов в решении задач управления электроприводами. – М.: Электротехника, 1999. – № 5. –   
   С. 2 – 6.
6. Клепиков В.Б., Сергеев С.А., Махотило К.В., Обруч И.В. Применение методов нейронных сетей и генетических алгоритмов в решении задач управления электроприводами // Электротехника. – 1999. – № 5. – С. 2 – 6.
7. Комашинский В.И. Нейронные сети и их применение в системах управления и святи. – М.: "Горячая линия – Телеком", 2003. – 96 с.
8. Методические указания к лабораторной работе "Нейронная сеть Хебба". – Харьков: НТУ "ХПИ", 2002. – 12 с.
9. Назаров А., Лоскутов А. Нейросетевые алгоритмы прогнозирования и оптимизации систем. – С-Пб.: Наука и Техника, 2003. – 492 с.
10. Перехов В.М. Современные способы управления и их применение в электроприводе // Электротехника. – 2000. – № 2. – С. 25 – 28.
11. Терехов В.А., Ефимов Д.В., Тюкин И.Ю., Антонов В.Н. Нейросетевые системы управления. – С-Пб: Издательство С.-Петербургского университета, 1999. – 265 с.
12. Уоссермен Ф. Нейрокомпьютерная техника. – М.: Мир, 1997. – 430 с.
13. Щетинин В.Г. Многослойная самоорганизация нейронных сетей оптимальной сложности // Автоматика и вычислительная техника. – Рига, 1998. – j4. – С. 30 – 37.
14. Handschin E., Kehlmann D., Hoffman W. Fault diagnosis in electrical energy systems using devic-specific artificial neural networks // Eng. Intell. Syst. Electr. Eng. Commun. – 1994. – Vol. 2, №. 4. – P. 255 – 262.
15. Hsieh K.-R., Chen W.-T. A Neural Network Model which Combines Unsupervised and Supervised Learning // IEEE Trans. on Neural Networks. – 1993. – Vol. 4, №. 2. – P. 35 – 39.
16. Mielczarska G.M., Mielczarski W. Применение нейронных сетей для оценки динамического состояния синхронных генераторов // Dynamic state estimation of a synchronous generator using neural-networks techniques. – № 92/15. – Inst. Eng., Austral. – 1992. – Р. 21 – 28.
17. Sankar K. Mitra S. Multilayer Perceptron, Fuzzy Sets and Classification // IEEE Transactions on Neural Networks. – Vol. 3, №5. – 1992. – Р. 683 – 696.
18. Srinivasan D. A novel approach to electrical load Forecasting based on a neural network // INNC-91. – 1991. – Р. 1172 – 1177.
19. Trzynadlowski A.M. Application of neural networks to the optimal control of Three-phase Voltage-Controlled Inverters // IEEE Trans. on Power Electronics. – 1994. – Vol. 9, №4. – Р. 34 – 38.
20. Trzynadlowski A.M., Legowski S. Application of neural networks to the optimal control of three-phase voltage-controlled inverters // IEEE Trans. Power Electron. – 1999. – Vol. 9, №.4. – P. 397 – 404.
21. Vaubois G. de La C., Moulinoux С., Derot B. The N Programming Language // Neurocomputing. – Vol. F68. – P. 89 – 92.
22. Widrow B., Lehr M.A. 30 Years of Adaptive Neural Networks: Perception, Madeline, and Back propagation // IEEE Computer Society. – 1992. –   
    Р. 327 – 354.

**Пояснювальна записка**

зміст

1. Нейронні мережі – основні поняття і визначення
2. Історія еволюції нейронних мереж. Їх основні моделі
   1. Модель Маккалоха
   2. Модель Розенблата
   3. Модель Хопфілда
   4. Мережі зі зворотним розповсюдженням
3. Основні алгоритми навчання і функціонування нейронних мереж
   1. Алгоритм навчання з вчителем (алгоритм зворотного розповсюдження багатошарових нейронних мереж)
   2. Алгоритм навчання без вчителя (алгоритм прямого розповсюдження нейронних мереж)
   3. Алгоритми функціонування мереж Хопфілда і Хемінга
4. Мережа Хебба. Алгоритм Хебба навчання нейронних мереж
5. Області вживання і задачі розв’язувані за допомогою нейронних мереж
6. Сучасні проекти і вироби, засновані на нейронних мережах
7. **Нейронні мережі – основні поняття і визначення**

В останні десятиліття у світі бурхливо розвивається нова прикладна галузь математики, що спеціалізується на штучних нейронних мережах (НМ). Актуальність досліджень у цьому напрямку підтверджується масою різних застосувань НМ. Це автоматизація процесів розпізнавання образів, адаптивне керування, апроксимація функціоналів, прогнозування, створення експертних систем, організація асоціативної пам'яті і багато інших додатків. За допомогою НМ можна, наприклад, пророкувати показники біржового ринку, виконувати розпізнавання оптичних або звукових сигналів, створювати системи, що самонавчаються, здатні керувати автомашиною при паркуванні або синтезувати мову за текстом.

Широке коло задач, розв'язувані НМ, не дозволяє в даний час створювати універсальні, могутні мережі, змушуючи розробляти спеціалізовані НМ, що функціонують за різними алгоритмами. Самі ж моделі НМ можуть бути програмного й апаратного виконання.

Незважаючи на істотні розходження, окремі типи НМ володіють декількома загальними рисами. Так в основу штучних нейронних мереж покладені наступні риси живих нейронних мереж:

* простий обробний елемент - нейрон;
* дуже велике число нейронів бере участь в обробці інформації;
* один нейрон зв'язаний з великим числом інших нейронів (глобальні зв'язки);
* що змінюються по вазі зв'язку між нейронами;
* масована паралельність обробки інформації.

Прототипом для створення нейрона послужив біологічний нейрон головного мозку. Біологічний нейрон має тіло, сукупність відростків – дендридів (синапсів), по яких у нейрон надходять вхідні сигнали, і відросток – аксон, що передає вихідний сигнал нейрона іншим клітинам. Біологічна модель штучного нейрона приведена на рис. 1:

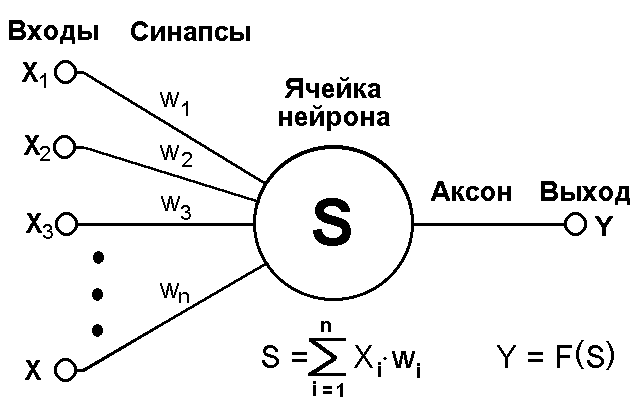


Рис. 1 Штучний нейрон

Спрощене функціонування нейрона можна представити в такий спосіб:

1. Нейрон одержує від дендридів набір (вектор) вхідних сигналів;
2. У тілі нейрона оцінюється сумарне значення вхідних сигналів. Однак входи нейрона нерівнозначні. Кожен вхід характеризується деяким ваговим коефіцієнтом, що визначає важливість інформації, що надходить по ньому. Таким чином, нейрон не просто підсумовує значення вхідних сигналів, а обчислює скалярний добуток вектора вхідних сигналів і вектора вагових коефіцієнтів;
3. Нейрон формує вихідний сигнал, інтенсивність якого залежить від значення обчисленого скалярного добутку. Якщо - воно не перевищує деякого заданого порогу, то вихідний сигнал не формується зовсім – нейрон "не спрацьовує";
4. Вихідний сигнал надходить на аксон і передається дендридам інших нейронів.

У такий спосіб поточний стан нейрона визначається, як зважена сума його входів плюс сигнал зсуву (зазвичай це 1), помножений на його коефіцієнт:

 (1.1)

де S – сумарний вхідний сигнал; wi () – вагові коефіцієнти зв'язків вхідних сигналів х1, …, хп; w0 – ваговий коефіцієнт зв'язку сигналу зсуву.

А вихід нейрона є функція його стану:

y = f(s) (1.2)

Виходячи з цього, біологічне представлення нейрону замінюють моделлю процесорного елемента наступного виду:



#### S

ƒ

*y*

**





1





Рис. 2 Процесорний елемент нейронної мережі

Нелінійна функція f називається активаційною і може мати різний вигляд. Найбільш розповсюдженими функціями активації є бінарна

 (1.3)

або біполярна

 (1.4)

Багато авторів при описанні моделі нейрона використовують не сигнал зсуву, а поріг  нейрона, що приводить до еквівалентної моделі елемента. У цьому випадку вираження (1.3) і (1.4) приймають відповідно вигляд:

 (1.5)

 (1.6)

де

 (1.7)

Графічне зображення бінарної і біполярної функцій активації для цього випадку представлене на рис. 3а і 3b.

##### S

*e*)

Рис. 3 Функції активації нейронів

##### S

*y*

*e*)

0

1

##### S

*y*

0

##### S

*y*

*f*)

0

1

–1

##### S

*a*)

1

##### S

*y*

*a*)

0

1

θ

##### S

*c*)

1

##### S

*y*

*c*)

0

1

θ1

θ2

##### S

*b*)

θ

##### S

*y*

0

1

–1

θ

##### S

*y*

θ1

θ2

##### S

*y*

*d*)

0

1

–1

θ1

θ2

##### S

*y*

##### S

θ1 θ2 θ*k*

##### S

*y*

*h*)

0

1

1

##### S

*g*)

##### S

*y*

–σ 0 σ

1

##### S

*i*)

0

θ1 θ2 θ*k*

-1

Зі зіставлення виразів (1.1)–(1.3) і (1.4)–(1.6) випливає, що кожному значенню порогу нейрона може бути поставлений у відповідність ваговий коефіцієнт w0 зв'язку сигналу зсуву і навпаки.

Рідше використовуються лінійні бінарні або біполярні функції активації (рис. 3с и 3d):

 (1.8)

де а дорівнює нулю для бінарних вихідних сигналів нейронів і а дорівнює мінус одиниці для біполярних сигналів; k, a0 постійні коефіцієнти.

Також широко використовуються бінарна сигмоидальная або логічна сигмоидальная функція (рис. 3e):

, (1.9)

де τ – постійний коефіцієнт;

і біполярна сигмоїдальна (рис. 3f):

, (1.10)

При зменшенні  сигмоїд стає більш положистим, і в межі при =0 вироджується в горизонтальну лінію на рівні 0.5, а при збільшенні  сигмоїд наближається, за зовнішнім виглядом, до функції одиничного стрибка з порогом  у крапці x=0. Зручність сигмоїдальної функції в тому, що вона диференціюється на всій осі абсцис, що використовується в деяких алгоритмах навчання. Крім того, вона має властивість підсилювати слабкі сигнали краще, ніж сильні, і запобігає насичення від великих сигналів, тому що вони відповідають областям аргументів, де сигмоїд має положистий нахил.

У нейронних мережах також використовуються й інші функції:

радіально-симетрична (рис. 3g):

, (1.11)

К-значна бінарна (рис. 3h):

 (1.12)

К-значна біполярна (рис. 3i):

 (1.13)

Нейронні мережі відрізняються не тільки активаційною функцією їхніх нейронів, вони бувають одно- і багатошарової структури, відрізняються за засобом навчання. Так навчання НМ може вестися з вчителем або без нього. У першому випадку мережі пред'являються значення як вхідних, так і бажаних вихідних сигналів, і вона по деякому внутрішньому алгоритмі підбудовує ваги своїх синаптичних зв'язків. В другому випадку виходи НМ формуються самостійно, а ваги змінюються по алгоритму, що враховує тільки вхідні і похідні від них сигнали.

Крім того різні алгоритми навчання, поділяються на два великих класи: детерміністські і стохастичні. У першому з них підстроювання ваг являє собою тверду послідовність дій, у другому – вона виробляється на основі дій, що підкорюються деякому випадковому процесові.

Розвиваючи далі питання про можливу класифікацію НМ, важливо відзначити існування бінарних і аналогових мереж. Перші з них оперують із двійковими сигналами, вихід кожного нейрона може приймати тільки два значення: логічний нуль ("загальмований" стан) і логічна одиниця ("збуджений" стан). В аналогових мережах вихідні значення нейронів здатні приймати безперервні значення.

Ще одна класифікація поділяє НМ на синхронні й асинхронні. У першому випадку у кожен момент часу свій стан змінює лише один нейрон. В другому – стан змінюється відразу в цілій групі нейронів, як правило, усього шару. Алгоритмічно хід часу в НМ задається ітераційним виконанням однотипних дій над нейронами.

Вибір типу НМ, методу навчання, її структури здійснюється відповідно до особливостей і складності задачі. Для вирішення деяких окремих типів задач вже існують оптимальні, на сьогоднішній день, конфігурації. Якщо ж задача не може бути зведена до жодного з відомих типів, розробник змушений вирішувати складну проблему синтезу нової конфігурації. При цьому він керується декількома основними принципами: можливості мережі зростають зі збільшенням числа осередків мережі, щільності зв'язків між ними і числом виділених шарів; введення зворотних зв'язків поряд зі збільшенням можливостей мережі піднімає питання про динамічну стійкість мережі; складність алгоритмів функціонування мережі (у тому числі, наприклад, введення декількох типів синапсів – збудливих, гальмуючих та ін.) також сприяє посиленню потужності НМ. Питання про необхідні і достатні властивості мережі для вирішення того або іншого роду задач являє собою цілий напрямок нейрокомп’ютерної науки. Так як проблема синтезу НМ сильно залежить від розв'язуваної задачі, дати загальні докладні рекомендації важко. У більшості випадків оптимальний варіант виходить на основі інтуїтивного підбору відповідно до потужності і можливостей обчислювальної машини або мікросхеми на якій виконується НМ.

1. **Історія еволюції нейронних мереж. Їх основні моделі**
   1. **Модель Маккалоха**

Теоретичні основи нейроматематики були закладені на початку 40-х років. У 1943 році У. Маккалох та його учень У. Питтс сформулювали основні положення теорії діяльності головного мозку. Ними були отримані наступні результати:

– розроблена модель нейрона як найпростішого процесорного елементу, що виконує обчислення перехідної функції від скалярного добутку вектора вхідних сигналів і вектора вагових коефіцієнтів;

– запропонована конструкція мережі таких елементів для виконання логічних і арифметичних операцій;

– зроблено основне припущення про те, що така мережа здатна навчатися, розпізнавати образи, узагальнювати отриману інформацію.

Незважаючи на те, що за минулі роки нейроматематика пішла далеко вперед, багато тверджень Макклоха залишаються актуальними і зараз. Зокрема, при великій розмаїтості моделей нейронів принцип їхньої дії, закладений Макклохом і Питтсом, залишається незмінним.

Недоліком даної моделі є сама модель нейрона "пороговим" видом перехідної функції. У формалізмі У. Маккалоха і У. Питтса нейрони мають стани 0, 1 та граничну логіку переходу зі стану в стан. Кожен нейрон у мережі визначає зважену суму станів всіх інших нейронів і порівнює її з порогом, щоб визначити свій власний стан. Пороговий вид функції не надає нейронній мережі достатньої гнучкості при навчанні і настроюванні на поставлене завдання. Якщо значення обчисленого скалярного добутку, навіть незначно, не досягає до заданого порогу, то вихідний сигнал не формується зовсім і нейрон "не спрацьовує". Це значить, що губиться інтенсивність вихідного сигналу (аксона) даного нейрона і, отже, формується невисоке значення рівня на зважених входах у наступному шарі нейронів.

* 1. **Модель Розенблата**

Значний розвиток нейрокибернетика одержала в роботах американського нейрофізіолога Френсиса Розенблата (Корнельский університет). У 1958 році він запропонував свою модель нейронної мережі. Розенблат ввів у модель Маккаллока і Питтса здатність зв'язків до модифікації, що зробило її навчальною. Ця модель була названа персептроном. Спочатку персептрон являв собою одношарову структуру з твердою пороговою функцією процесорного елемента і бінарними або багатозначними входами. Перші персептрони були здатні розпізнавати деякі літери латинського алфавіту. Згодом модель персептрона була значно вдосконалена.

Персептрон застосовувався для задачі автоматичної класифікації, що у загальному випадку полягає в поділі простору ознак між заданою кількістю класів. У двомірному випадку потрібно провести лінію на площині, що відокремлює одну область від іншої. Персептрон здатний поділяти простір тільки прямими лініями (площинами).

Алгоритм навчання персептрона виглядає наступним чином:

1) системі пред'являється еталонний образ;

2) якщо виходи системи спрацьовують правильно, вагові коефіцієнти зв'язків не змінюються;

3) якщо виходи спрацьовують неправильно, ваговим коефіцієнтам дається невелике збільшення убік підвищення якості розпізнавання.

Серйозним недоліком персептрона є те, що не завжди існує така комбінація вагових коефіцієнтів, при якій наявна безліч образів буде розпізнаватися даним персептроном. Причина цього недоліку полягає в тому, що лише невелика кількість задач припускає, що лінія, що розділяє еталони, буде прямою. Звичайно це досить складна крива, замкнута або розімкнута. Якщо врахувати, що одношаровий персептрон реалізує тільки лінійну поділяючу поверхню, застосування його там, де потрібно нелінійна, приводить до невірного розпізнавання (ця проблема називається лінійної нероздільністю простору ознак). Виходом з цього положення є використання багатошарового персептрона, здатного будувати ламаний кордон між розпізнаваними образами.

Описана проблема не є єдиними труднощами, що виникають при роботі з персептронами – також слабко формалізований метод навчання персептрона. Персептрон поставив ряд питань, робота над вирішенням яких призвела до створення більш "розумних" нейронних мереж і розробці методів, що знайшли застосування не тільки в нейрокибернетиці (наприклад, метод групового обліку аргументів, застосовуваний для ідентифікації математичних моделей).

* 1. **Модель Хопфілда**

У 70-і роки зацікавленість до нейронними мережами значно зменшала, однак роботи з їхнього дослідження продовжувалися. Був запропонований ряд цікавих розробок, таких, наприклад, як когнитрон, здатний добре розпізнавати досить складні образи (ієрогліфи і т.п.) незалежно від повороту і зміни масштабу зображення. Автором когнитрона є японський вчений И. Фукушима.

Новий виток швидкого розвитку моделей нейронних мереж, що почався 8-9 років тому , пов'язаний з роботами Амари, Андерсона, Карпентера, Кохена та інших, і, особливо, Хопфилда, а також під впливом багатообіцяючих успіхів оптичних технологій і зрілої фази розвитку СБІС для реалізації нових архітектур.

Початок сучасному математичному моделюванню нейронних обчислень було покладено роботами Хопфилда в 1982 році, у яких була сформульована математична модель асоціативної пам'яті на нейронній мережі з використанням правила Хеббиана для програмування мережі. Але не стільки сама модель стала поштовхом до появи робіт інших авторів на цю тему, скільки введена Хопфилдом функція обчислення енергії нейронної мережі. Це аналог функції Ляпунова в динамічних системах. Показано, що для одношарової нейронної мережі зі зв'язками типу "усі на всіх" характерна збіжність до однієї з кінцевої безлічі рівноважних крапок, що є локальними мінімумами функції енергії, що містить у собі усю структуру взаємозв'язків у мережі. Розуміння такої динаміки в нейронній мережі було й в інших дослідників. Однак, Хопфилд і Тэнк показали як конструювати функцію енергії для конкретної оптимізаційної задачі і як використовувати її для відображення задачі у нейронну мережу. Цей підхід одержав розвиток і для вирішення інших комбінаторних оптимізаційних задач. Привабливість підходу Хопфилда полягає в тому, що нейронна мережа для конкретної задачі може бути запрограмована без навчальних ітерацій. Ваги зв'язків обчислюються на підставі виду функції енергії, сконструйованої для цієї задачі.

Розвитком моделі Хопфилда для вирішення комбінаторних оптимізаційних задач і задач штучного інтелекту є машина Больцмана, запропонована і досліджена Джефери Е. Хинтоном і Р. Земелом. У ній, як і в інших моделях, нейрон має стани 1, 0 і зв'язок між нейронами має вагу. Кожен стан мережі характеризується визначеним значенням функції консенсусу (аналог функції енергії). Максимум функції консенсусу відповідає оптимальному вирішенню задачі. Є наступна інформація про результати моделювання на ЕОМ роботи нейронної мережі. Моделювалася асинхронна робота мережі Хопфилда. Мережа працює добре, тобто без помилок відновлює еталонні образи з випадкових, якщо в неї записується не більш 15 % еталонних образів. Іспити проводилися для 30 нейронів і для 100 нейронів у мережі. Бралася деяка кількість випадкових векторів у якості еталонних і будувалася відповідна матриця ваг зв'язків. Моделювання при 100 нейронах було істотно більш повільним процесом, ніж при 30 нейронах, хоча якісна картина і у тому і в іншому випадках була та сама. Приблизно 88 % іспитів закінчувалися в еталонних станах, 10 % – у стійких станах, близьких до еталонного. При відстані ≤ 5 між початковим і еталонним векторами, еталонний стан досягався в 90 % випадків. Зі збільшенням відстані, імовірність влучення в найбільш близький еталонний стан гладко спадала. При відстані 12 імовірність дорівнювала 0.2. Стійкі стани, занадто близькі друг до друга, мають тенденцію "зливатися", вони попадають в одну западину на енергетичній поверхні. Програмувалася задача комівояжера на основі мережі Хопфилда. Мережею зі 100 нейронів для 20 різних випадкових початкових станів були визначені маршрути, 16 з яких були прийнятними, 50 % спроб дали 2 шляхи 2.83 і 2.71 (цифри приводяться, щоб показати як вони близькі) при найкоротшому 2.67. Це результати моделювання роботи мережі з безперервною моделлю нейрона. Моделювалася також задача комівояжера, але для мережі типу машина Больцмана, проводилася при наступних значеннях керуючих параметрів: A = 0.95, L = 10, M = 100 (A – позитивне число менше одиниці, але близьке до неї, L – число іспитів, що проводяться без змін, M – число послідовних іспитів, що не приводять до зміни стану машини, як критерію завершення процесу). Процес запускався 100 разів для n = 10 (усього в мережі N = n^2 нейронів) і 25 разів для n = 30 при різних нормальних станах машини Больцмана. Для n = 10 вийшов оптимальний результат, для n = 30 – вирішення на 14 % гірше оптимального. Відзначимо, що імовірний механізм функціонування машини Больцмана дає можливість одержати на ній трохи кращі результати оптимізації, чим на моделі Хопфилда.

* 1. **Мережі зі зворотним розповсюдженням**

З розвитком теорії нейронних мереж вони стають усе більше і здобувають переважно багатошарову структуру. Одним із розповсюджених способів навчання цих мереж став спосіб зворотного поширення (back propogatіon). У таких НМ зв'язок між собою мають тільки сусідні шари, при цьому кожен нейрон попереднього шару зв'язаний із усіма нейронами наступного шару. Нейрони звичайно мають сигмоїдальну функцію активації. Перший шар нейронів називається вхідним і містить число нейронів відповідне розпізнаному образові. Останній шар нейронів називається вихідним і містить стільки нейронів, скільки класів образів розпізнається. Між вхідним і вихідним шарами розташовується один або більше схованих (тіньових) шарів. Визначення числа схованих шарів і числа нейронів у кожнім шарі для конкретної задачі є неформальною задачею.

Принцип навчання такої нейронної мережі базується на обчисленні відхилень значень сигналів на вихідних процесорних елементах від еталонних і зворотньому "прогоні" цих відхилень до їхніх елементів, що породили, з метою корекції помилки. Ще в 1974 році Поль Дж. Вербос винайшов значно більш ефективну процедуру для обчислення величини, що називається похідною помилки по вазі, коли працював над своєю докторською дисертацією в Гарвардському університеті. Процедура, відома тепер як алгоритм зворотного поширення, стала одним з найбільш важливих інструментів у навчанні нейронних мереж. Однак цьому алгоритмові властиві і недоліки, головний з яких - відсутність скільки-небудь прийнятних оцінок часу навчання. Розуміння, що мережа зрештою навчиться, мало втішає, якщо на це можуть піти роки. Проте, алгоритм зворотного поширення має найширше застосування. Наприклад, успіх фірми NEC у розпізнаванні букв, був досягнутий саме завдяки алгоритмові зворотного поширення.

1. **Основні алгоритми навчання і функціонування нейронних мереж**
   1. **Алгоритм навчання з вчителем (алгоритм зворотного розповсюдження багатошарових нейронних мереж)**

Серед різних структур нейронних мереж (НМ) однієї з найбільш відомих є багатошарова структура, у якій кожен нейрон довільного шару зв'язаний із усіма аксонами нейронів попереднього шару або, у випадку першого шару, із усіма входами НМ. Такі НМ називаються повнозв’язними. Коли в мережі тільки один шар, алгоритм її навчання з вчителем досить очевидний, тому що правильні вихідні стани нейронів єдиного шару свідомо відомі, і підстроювання синаптичних зв'язків йдуть у напрямку, мінімізуючому помилку на виході мережі. На цьому принципі будується, наприклад, алгоритм навчання одношарового перцептрона. У багатошарових же мережах оптимальні вихідні значення нейронів усіх шарів, крім останнього, як правило, не відомі, і двох або більш шаровий перцептрон уже неможливо навчити, керуючись тільки величинами помилок на виходах НМ. Один з варіантів рішення цієї проблеми – розробка наборів вихідних сигналів, що відповідають вхідним, для кожного шару НМ, що, звичайно, є дуже трудомісткою операцією і не завжди здійсненно. Другий варіант – динамічне підстроювання вагових коефіцієнтів синапсів, у ході якої вибираються, як правило, найбільш слабкі зв'язки і змінюються на малу величину в ту або іншу сторону, а зберігаються тільки ті зміни, що спричинили зменшення помилки на виході всієї мережі. Очевидно, що даний метод "тику", незважаючи на свою удавану простоту, вимагає великих рутинних обчислень. І, нарешті, третій, більш прийнятний варіант - поширення сигналів помилки від виходів НМ до її входів, у напрямку, зворотньому прямому поширенню сигналів у звичайному режимі роботи. Цей алгоритм навчання НМ одержав назву процедури зворотного поширення. Саме він буде розглянутий надалі.

Відповідно до методу найменших квадратів, мінімізуючою цільовою функцією помилки НМ є величина:

, (3.1)

де  – реальний вихідний стан нейрону j вихідного шару N нейронної мережі при подаванні на її входи p-го образу; djp – ідеальний (бажаний) вихідний стан цього нейрону.

Підсумовування ведеться по всіх нейронах вихідного шару і по всім оброблюваним мережею образам. Мінімізація ведеться методом градієнтного спуску, що означає підстроювання вагових коефіцієнтів у такий спосіб:

. (3.2)

Тут wij – ваговий коефіцієнт синаптичного зв’язку, який поєднує i-ий нейрон шару n-1 с j-им нейроном шару n, η – коефіцієнт швидкості навчання, 0<η<1.

Відомо що,

. (3.3)

Тут під yj, як і раніш, мається на увазі вихід нейрона j, а під sj – зважена сума його вхідних сигналів, тобто аргумент активаційної функції. Оскільки множник dyj/dsj є похідна цієї функції по її аргументу, з цього випливає, що похідна активаційної функції повинна бути визначена на всій осі абсцис. У зв'язку з цим функція одиничного стрибка та інші активаційні функції з неоднорідностями не підходять для розглянутих НМ. У них застосовуються такі гладкі функції, як гіперболічний тангенс або класичний сигмоїд з експонентою. У випадку гіперболічного тангенсу маємо:

. (3.4)

Третій множник ∂sj/∂wij, очевидно, дорівнює виходу нейрона попереднього шару yi(n-1).

Що стосується першого множника в (3.3), він легко розкладається наступним чином :

. (3.5)

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

Запровадивши нову змінну

, (3.6)

ми отримаємо рекурсивну формулу для підрахунку величин δj(n) шару n з величин δk(n+1) більш старшого шару n+1:

. (3.7)

Для вихідного шару

. (3.8)

Тепер ми можемо записати формулу (3.2) у розкритому вигляді:

. (3.9)

Іноді для додання процесові корекції ваг деякої інерційності, що згладжує різкі стрибки при переміщенні по поверхні цільової функції, (3.9) доповнюється значенням зміни ваги на попередній ітерації

, (3.10)

де μ – коефіцієнт інерційності, t – номер поточної ітерації.

Таким чином, повний алгоритм навчання НМ за допомогою процедури зворотного розповсюдження будується так:

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

, (3.11)

де M – число нейронів у шарі n-1 з урахуванням нейрону з постійним вихідним станом +1, який задає зсув; yi(n-1)=xij(n) – i-ий вхід нейрону j шару n;

yj(n) = f(sj(n)), (3.12)

де f() – сигмоїд;

yq(0)=Iq, (3.13)

де Iq – q-а компонента вектору вхідного образу.

2. Обчислити δ(N) для вихідного шару по формулі (3.8). Обчислити по формулі (3.9) або (3.10) зміни вагів Δw(N) шару N.

3. Обчислити по формулам (3.7) і (3.9) (або (3.7) і (3.10)) відповідно δ(n) і Δw(n) для усіх інших шарів, n=N-1,...1.

4. Скорегувати усі ваги в НМ:

. (3.14)

5. Якщо помилка мережі значна, перейти до кроку 1. В протилежному випадку – кінець.

Мережі на кроці 1 поперемінно у випадковому порядку пред'являються всі тренувальні образи, щоб мережа, образно кажучи, не забувала одні в міру запам'ятовування інших. Алгоритм ілюструється на рис. 4.

З виразу (3.9) випливає, що коли вихідне значення yi(n-1) наближається до нуля, ефективність навчання помітно знижується. При двійкових вхідних векторах у середньому половина вагових коефіцієнтів не буде коректуватися, тому ділянку можливих значень виходів нейронів [0,1] бажано локалізувати в межі [-0.5,+0.5], що досягається простими модифікаціями логістичних функцій. Наприклад, сигмоїд з експонентою перетвориться до виду:

. (3.15)

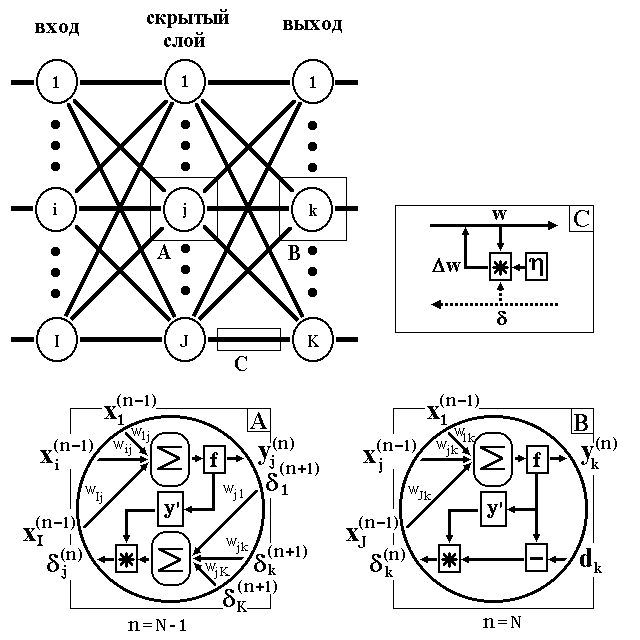


Рис. 4. Діаграма сигналів у мережі при навчанні за алгоритмом зворотного поширення

Тепер торкнемося питання ємності НМ, тобто числа образів, пропонованих на її входи, що вона здатна навчитися розпізнавати. Для мереж з числом шарів більше двох, він залишається відкритим. Для НМ із двома шарами, тобто вихідним і одним схованим шаром, детерміністська ємність мережі Cd оцінюється так:

Nw/Ny<Cd<Nw/Ny⋅log(Nw/Ny), (3.16)

де Nw – число ваг, що підбудовуються, Ny – число нейронів у вихідному шарі.

Слід зазначити, що даний вираз отриманий із урахуванням деяких обмежень. По-перше, число входів Nx і нейронів у схованому шарі Nh повинно задовольняти нерівності Nx+Nh>Ny. По-друге, Nw/Ny>1000. Однак вищенаведена оцінка виконувалася для мереж з активаційними функціями нейронів у вигляді порогу, а ємність мереж із гладкими активаційними функціями, наприклад – (3.15), зазвичай більше. Крім того, що фігурує в назві ємності, прикметник "детерміністський" означає, що отримана оцінка ємності підходить абсолютно для всіх можливих вхідних образів, що можуть бути представлені Nx входами. У дійсності розподіл вхідних образів, як правило, має деяку регулярність, що дозволяє НМ проводити узагальнення і, таким чином, збільшувати реальну ємність. Так як розподіл образів, у загальному випадку, заздалегідь не відомий, ми можемо говорити про таку ємність тільки приблизно, але зазвичай вона разів у два перевищує ємність детерміністську.

У продовження розмови про ємність НМ логічно торкнутися питання про необхідну потужність вихідного шару мережі, що виконує кінцеву класифікацію образів. Справа в тому, що для поділу безлічі вхідних образів, наприклад, по двох класах, досить всього одного виходу. При цьому кожен логічний рівень – "1" і "0" – буде позначати окремий клас. На двох виходах можна закодувати вже 4 класи і так далі. Однак результати роботи мережі, організованої таким чином, можна сказати – "під зав'язку", – не дуже надійні. Для підвищення вірогідності класифікації бажано увести надмірність шляхом виділення кожному класові одного нейрона у вихідному шарі або, що ще краще, декількох, кожен з яких навчається визначати належність образу до класу зі своїм ступенем вірогідності, наприклад: високим, середнім і низьким. Такі НМ дозволяють проводити класифікацію вхідних образів, об'єднаних у нечіткі (розмиті або пересічні) безлічі. Ця властивість наближає подібні НМ до умов реального життя.

Розглянута НМ має декілька "вузьких місць". По-перше, у процесі навчання може виникнути ситуація, коли великі позитивні або негативні значення вагових коефіцієнтів змістять робочу крапку на сигмоїдах багатьох нейронів в ділянку насичення. Малі величини похідної від логістичної функції призведуть у відповідність з (3.7) і (3.8) до зупинки навчання, що паралізує НМ. По-друге, застосування методу градієнтного спуску не гарантує, що буде знайдений глобальний, а не локальний мінімум цільової функції. Ця проблема зв'язана ще з однією, а саме – з вибором величини швидкості навчання. Доказ збіжності навчання в процесі зворотного поширення засновано на похідних, тобто збільшення ваг і, отже, швидкість навчання повинна бути нескінченно малою, однак у цьому випадку навчання буде відбуватися неприйнятно повільно. З іншого боку, занадто великі корекції ваг можуть призвести до постійної нестійкості процесу навчання. Тому в якості швидкості навчання зазвичай вибирається число менше 1, але не дуже маленьке, наприклад, 0.1, і воно, узагалі говорячи, може поступово зменшуватися в процесі навчання. Крім того, для виключення випадкових влучень у локальні мінімуми іноді, після того як значення вагових коефіцієнтів стабілізуються, швидкість навчання короткочасно сильно збільшують, щоб почати градієнтний спуск із нової крапки. Якщо повторення цієї процедури кілька разів приведе алгоритм у той самий стан НМ, можна більш-менш впевнено сказати, що знайдено глобальний максимум, а не якийсь інший. Проте даний метод є досить ефективним і широко застосовується.

Слід зазначити, що існує й інший метод виключення локальних мінімумів і, крім того, паралічу НМ, що полягає в застосуванні стохастичних НМ, про які йшла мова в попередніх главах.

У такий спосіб, розглянутий у даній главі, алгоритм навчання нейронної мережі за допомогою процедури зворотного поширення має на увазі наявність якоїсь зовнішньої ланки, що надає мережі крім вхідних так само і цільові вихідні образи. Алгоритми, що користуються подібною концепцією, називаються алгоритмами навчання з вчителем. Для їхнього успішного функціонування необхідна наявність експертів, що створюють на попередньому етапі для кожного вхідного образу еталонний вихідний.

* 1. **Алгоритм навчання без вчителя (алгоритм прямого розповсюдження нейронних мереж)**

Оскільки створення штучного інтелекту рухається по шляху копіювання природних прообразів, вчені не припиняють суперечку на тему, чи можна вважати алгоритми навчання з вчителем натуральними або ж вони є цілком штучними. Наприклад, навчання людського мозку, на перший погляд, відбувається без вчителя: на зорові, слухові, тактильні та інші рецептори надходить інформація ззовні, і усередині нервової системи відбувається якась самоорганізація. Однак, не можна заперечувати і того, що в житті людини не мало вчителів – і в буквальному, і в переносному значенні, – які координують зовнішні впливи. Разом з тим, чим би не закінчилася суперечка прихильників цих двох концепцій навчання, вони обидві мають право на існування.

Головна риса, що робить навчання без вчителя привабливим, – це його "самостійність". Процес навчання, як і у випадку навчання з вчителем, полягає в підстроюванні ваг синапсів. Деякі алгоритми, щоправда, змінюють і структуру мережі, тобто кількість нейронів і їхні взаємозв'язки, але такі перетворення краще назвати більш широким терміном – самоорганізацією. Очевидно, що підстроювання синапсів може проводитися тільки на підставі інформації, доступної в нейроні, тобто його стану і вже наявних вагових коефіцієнтів. Виходячи з цього розуміння і, що більш важливо, за аналогією з відомими принципами самоорганізації нервових клітин, побудовані алгоритми навчання Хебба.

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

, (3.17)

де yi(n-1) – вихідне значення нейрону i шару (n-1), yj(n) – вихідне значення нейрону j шару n; wij(t) і wij(t-1) – ваговий коефіцієнт синапсу, який поєднує ці нейрони, на ітераціях t и t‑1 відповідно; α – коефіцієнт швидкості навчання. Тут і надалі, для загальності, під n розуміється довільний шар мережі. При навчанні по даному методу підсилюються зв’язки між збудженими нейронами.

Існує також і диференціальний метод навчання Хебба:

. (3.18)

Тут yi(n-1)(t) та yi(n-1)(t-1) – вихідне значення нейрону i шару n-1 відповідно на ітераціях t і t-1; yj(n)(t) та yj(n)(t-1) – теж саме для нейрону j шару n. Як видно з формули (3.18), сильніше всього навчаються синапси, поєднуючи ті нейрони, виходи яких найбільш динамічно змінились в бік зростання.

Повний алгоритм навчання із застосуванням вищенаведених формул буде виглядати так:

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

2. На входи мережі подається вхідний образ, і сигнали порушення поширюються по всіх шарах відповідно до принципів класичних прямопоточних (feedforward) мереж, тобто для кожного нейрона розраховується зважена сума його входів, до якої потім застосовується активаційна (передаточна) функція нейрона, у результаті чого виходить його вихідне значення yі(n), і=0...Mі-1, де Mі – число нейронів у шарі і; n=0...N-1, а N – число шарів у мережі.

3. На підставі отриманих вихідних значень нейронів по формулі (3.17) або (3.18) виробляється зміна вагових коефіцієнтів.

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

На другому кроці циклу позмінно пред'являються всі образи з вхідного набору.

Слід зазначити, що вид відгуків на кожен клас вхідних образів не відомий заздалегідь і буде являти собою довільне сполучення станів нейронів вихідного шару, обумовлене випадковим розподілом ваг на стадії ініціалізації. Разом з тим, мережа здатна узагальнювати схожі образи, відносячи їх до одного класу. Тестування навченої мережі дозволяє визначити топологію класів у вихідному шарі. Для приведення відгуків навченої мережі до зручного представлення можна доповнити мережу одним шаром, що, наприклад, по алгоритму навчання одношарового перцептрона необхідно змусити відображати вихідні реакції мережі в необхідні образи.

Інший алгоритм навчання без вчителя – алгоритм Кохонена – передбачає підстроювання синапсів на підставі їхніх значень від попередньої ітерації.

. (3.19)

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

Повний алгоритм навчання має приблизно таку ж структуру, як у методах Хебба, але на кроці 3 з усього шару вибирається нейрон, значення синапсів якого максимально походять на вхідний образ, і підстроювання ваг по формулі (3.19) проводяться тільки для нього. Ця, так названа, акредитація може супроводжуватися загальмуванням всіх інших нейронів шару і введенням обраного нейрона в насичення. Вибір такого нейрона може здійснюватися, наприклад, розрахунком скалярного добутку вектора вагових коефіцієнтів з вектором вхідних значень. Максимальний добуток дає нейрон, що виграв.

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

, (3.20)

де j – індекс нейрону у шарі n, i – індекс підсумовування по нейронам шару (n‑1), wij – вага синапса, який поєднує нейрони; виходи нейронів шару (n-1) являються вхідними значеннями для шару n. Корінь у формулі (3.20) брати не обов’язково, тому що важлива тільки відносна оцінка різних Dj.

У даному випадку "перемагає" нейрон з найменшою відстанню. Іноді це занадто, тому нейрони, що часто одержують акредитацію, примусово виключаються з розгляду, щоб "зрівняти права" усіх нейронів шару. Найпростіший варіант такого алгоритму полягає в гальмуванні тільки того нейрону, що виграв.

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

, (3.21)

де xi – i-а компонента вектору вхідного образу або вектору вагових коефіцієнтів, а n – його розмірність. Це дозволяє скоротити довжину процесу навчання.

Ініціалізація вагових коефіцієнтів випадковими значеннями може призвести до того, що різні класи, яким відповідають щільно розподілені вхідні образи, зіллються або, навпаки, роздрібняться на додаткові підкласи у випадку близьких образів того самого класу. Для запобігання такій ситуації використовується метод опуклої комбінації. Сутність його зводиться до того, що вхідні нормалізовані образи піддаються перетворенню:

, (3.22)

де xi – i-а компонента вхідного образу, n – загальне число його компонент, α(t) - коефіцієнт, що змінюється в процесі навчання від нуля до одиниці, у результаті чого спочатку на входи мережі подаються практично однакові образи, а з часом вони усе більше сходяться до вихідного. Вагові коефіцієнти встановлюються на кроці ініціалізації рівними величині:

, (3.23)

де n – розмірність вектору вагів для нейронів ініціалізуємого шару.

На основі розглянутого вище методу будуються нейронні мережі особливого типу – так названі структури, що самоорганізовуються – self-organizing feature maps. Для них після вибору із шару n нейрона j з мінімальною відстанню Dj (3.20) навчається по формулі (3.19) не тільки цей нейрон, але і його сусіди, розташовані в околиці R. Величина R на перших ітераціях дуже велика, так що навчаються всі нейрони, але з часом вона зменшується до нуля. Таким чином, чим ближче кінець навчання, тим точніше визначається група нейронів, що відповідають кожному класові образів.

Необхідно відзначити, що навчання без вчителя набагато більш відчутно до вибору оптимальних параметрів, ніж навчання з вчителем. По-перше, його якість сильно залежить від початкових величин синапсів. По-друге, навчання критичне до вибору радіуса навчання і швидкості його зміни. І нарешті, зрозуміло, дуже важливий характер зміни власне коефіцієнта навчання. У зв'язку з цим розробнику, швидше за все, буде потрібно провести попередню роботу по підбору оптимальних параметрів навчання мережі.

Незважаючи на деякі складності реалізації, алгоритми навчання без вчителя знаходять велике й успішне застосування. По суті, за алгоритмом навчання без вчителя функціонують найбільш складні з відомих на сьогоднішній день штучні нейронні мережі - когнитрон і неокогнитрон, що максимально наблизилися у своєму втіленні до структури мозку. Однак вони, звичайно, істотно відрізняються від розглянутих вище мереж і набагато більш складні. Проте, на основі вищевикладеного матеріалу можна створити реально діючі системи для розпізнавання образів, стиску інформації, автоматизованого керування, експертних оцінок і багато іншого що буде продемонстроване на прикладі розробки мережі Хебба розпізнавання символів.

* 1. **Алгоритми функціонування мереж Хопфілда і Хемінга**

Серед різних конфігурацій штучних нейронних мереж (НМ) зустрічаються такі, при класифікації яких за принципом навчання, строго кажучи, не підходять ні навчання з вчителем, ні навчання без вчителя. У таких мережах вагові коефіцієнти синапсів розраховуються тільки один раз перед початком функціонуванням мережі на основі інформації про оброблення даний, і все навчання мережі зводиться саме до цього розрахунку. З одного боку, пред'явлення апріорної інформації можна розцінювати, як допомога вчителя, але з іншого боку - мережа фактично просто запам'ятовує зразки до того, як на її вхід надходять реальні дані, і не може змінювати своє поводження, тому казати про ланку зворотного зв'язку зі "світом" (вчителем) не приходиться. З мереж з подібною логікою роботи найбільш відомі мережа Хопфилда і мережа Хеммінга, що звичайно використовуються для організації асоціативної пам'яті.

Структурна схема мережі Хопфилда приведена на рис. 5. Вона складається з єдиного шару нейронів, число яких є одночасно числом входів і виходів мережі. Кожен нейрон зв'язаний синапсами зі всіма іншими нейронами, а також має один вхідний синапс, через який здійснюється введення сигналу. Вихідні сигнали, як звичайно, утворюються на аксонах.

Задача, розв'язувана даною мережею як асоціативна пам'ять, як правило, формулюється в такий спосіб. Відомий деякий набір двійкових сигналів (зображень, звукових оцифровок, інших даних, що описують якісь об'єкти або характеристики процесів), що вважаються зразковими. Мережа повинна вміти з довільного неідеального сигналу, поданого на її вхід, виділити ("згадати" за частковою інформацією) відповідний зразок (якщо такий є) або "дати висновок" про те, що вхідні дані не відповідають жодному зі зразків.

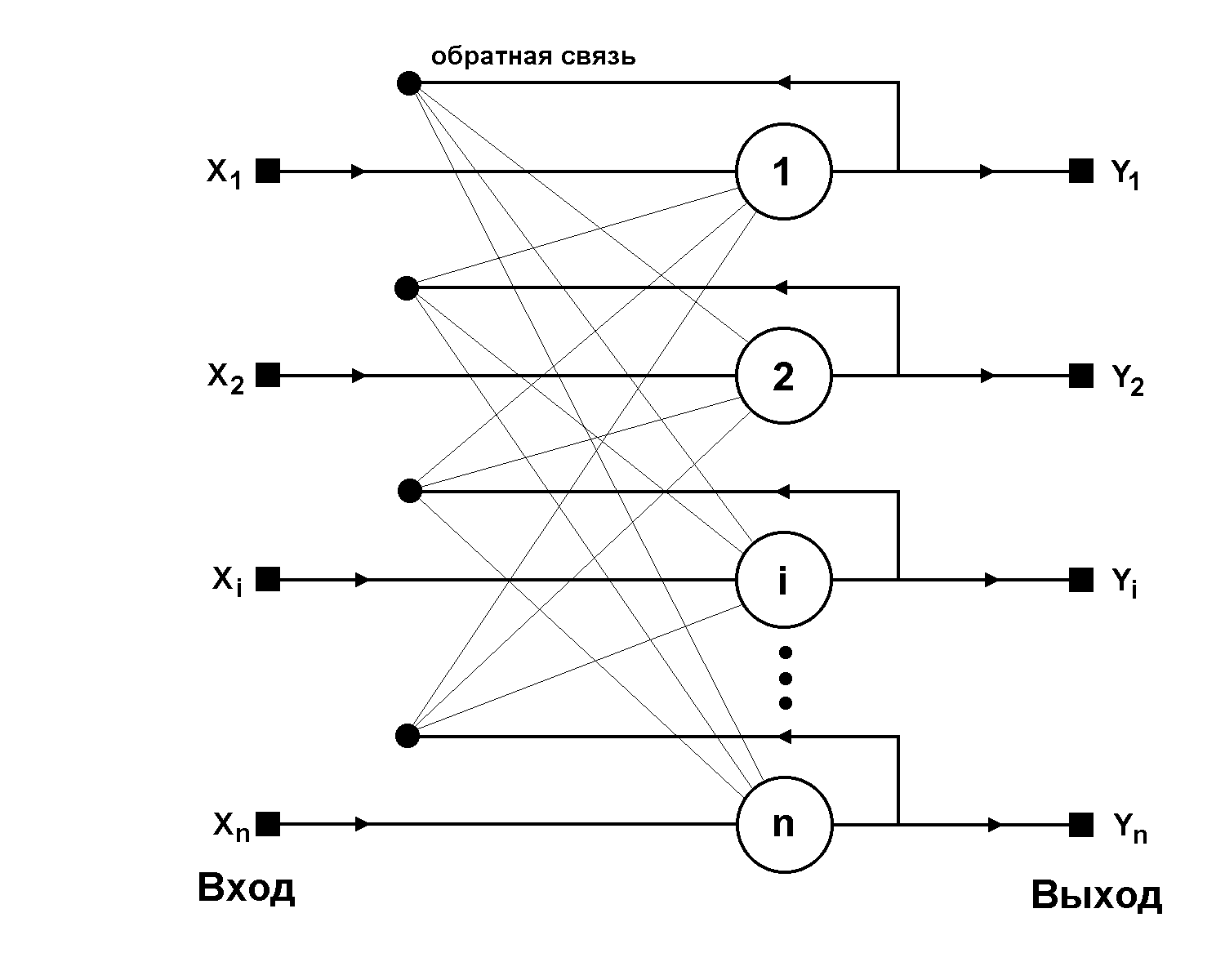


Рис. 5 Структурна схема мережі Хопфилда

У загальному випадку, будь-який сигнал може бути описаний вектором X = { xi: i=0...n-1}, n – число нейронів в мережі і розмірність вхідних та вихідних векторів. Кожен елемент xi дорівнює або +1, або –1. позначимо вектор, який описує k-ий зразок, через Xk, а його компоненти, відповідно, – xik, k=0...m-1, m – число зразків. Коли мережа розпізнає (або "згадає") який-небудь зразок на основі пред'явлених їй даних, її виходи будуть містити саме його, тобто Y = Xk, де Y – вектор вихідних значень мережі: Y = { yi: i=0,...n-1}. В противному випадку, вихідний вектор не співпадає не з одним зразковим.

Якщо, наприклад, сигнали являють собою якісь зображення, то, відобразивши в графічному вигляді дані з виходу мережі, можна буде побачити картинку, що цілком збігається з однією зі зразкових (у випадку успіху) або ж "вільну імпровізацію" мережі (у випадку невдачі).

На стадії ініціалізації мережі вагові коефіцієнти синапсів встановлюються в такий спосіб:

. (3.24)

Тут i та j – індекси, відповідно, предсинаптичного и постсинаптичного нейронів; xik, xjk – i-ий та j-ий елементи вектору k-ого зразку.

Алгоритм функціонування мережі наступний (p – номер ітерації):

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

yi(0) = xi , i = 0...n-1, (3.25)

тому позначення на схемі мережі вхідних синапсів у явному вигляді носить чисто умовний характер. Нуль у дужці праворуч від yі означає нульову ітерацію у циклі роботи мережі.

2. Обчислюється новий стан нейронів:

, j=0...n-1, (3.26)

і нові значення аксонів:

, (3.27)

де f – активаційна функція у вигляді стрибка, приведена на рис. 6а.

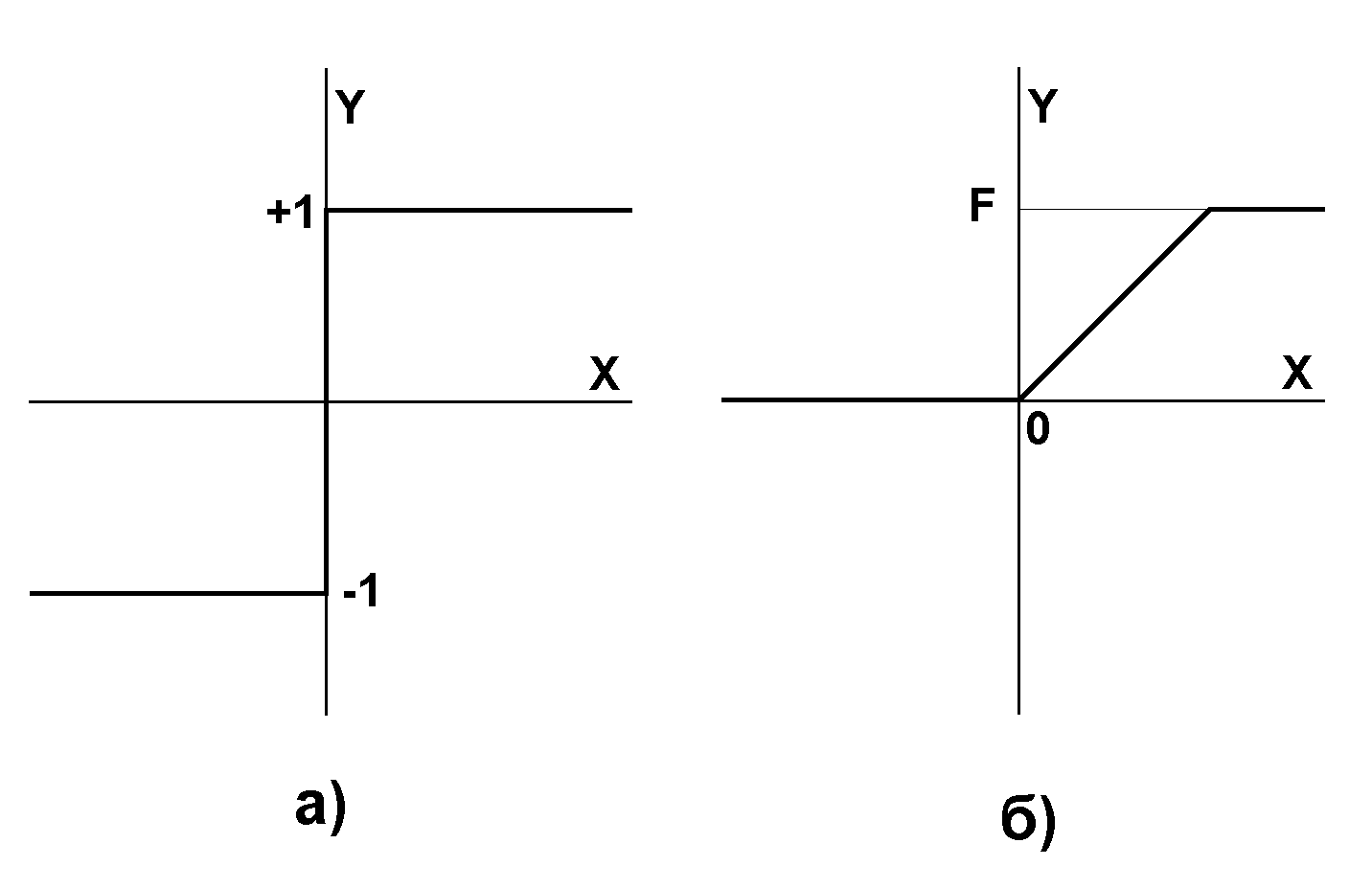


Рис. 6 Активаційні функції

3. Перевірка, чи змінилися вихідні значення аксонів за останню ітерацію. Якщо так – перехід до пункту 2, інакше (якщо виходи застабілізувалися) – кінець. При цьому вихідний вектор являє собою зразок, що найкраще сполучиться з вхідними даними.

Як було сказано вище, іноді мережа не може провести розпізнавання і видає на виході неіснуючий образ. Це зв'язано з проблемою обмеженості можливостей мережі. Для мережі Хопфилда число образів, що запам'ятовуються, m не повинне перевищувати величини, приблизно рівної 0.15n. Крім того, якщо два образи А і Б сильно схожі, вони, можливо, будуть викликати в мережі перехресні асоціації, тобто пред'явлення на входи мережі вектору А призведе до появи на її виходах вектору Б и навпаки.

Коли немає необхідності, щоб мережа в явному вигляді видавала зразок, тобто досить, скажемо, одержувати номер зразка, асоціативну пам'ять успішно реалізує мережа Хеммінга. Дана мережа характеризується, у порівнянні з мережею Хопфилда, меншими витратами на пам'ять і обсягом обчислень, що стає очевидним з її структури (рис. 7).

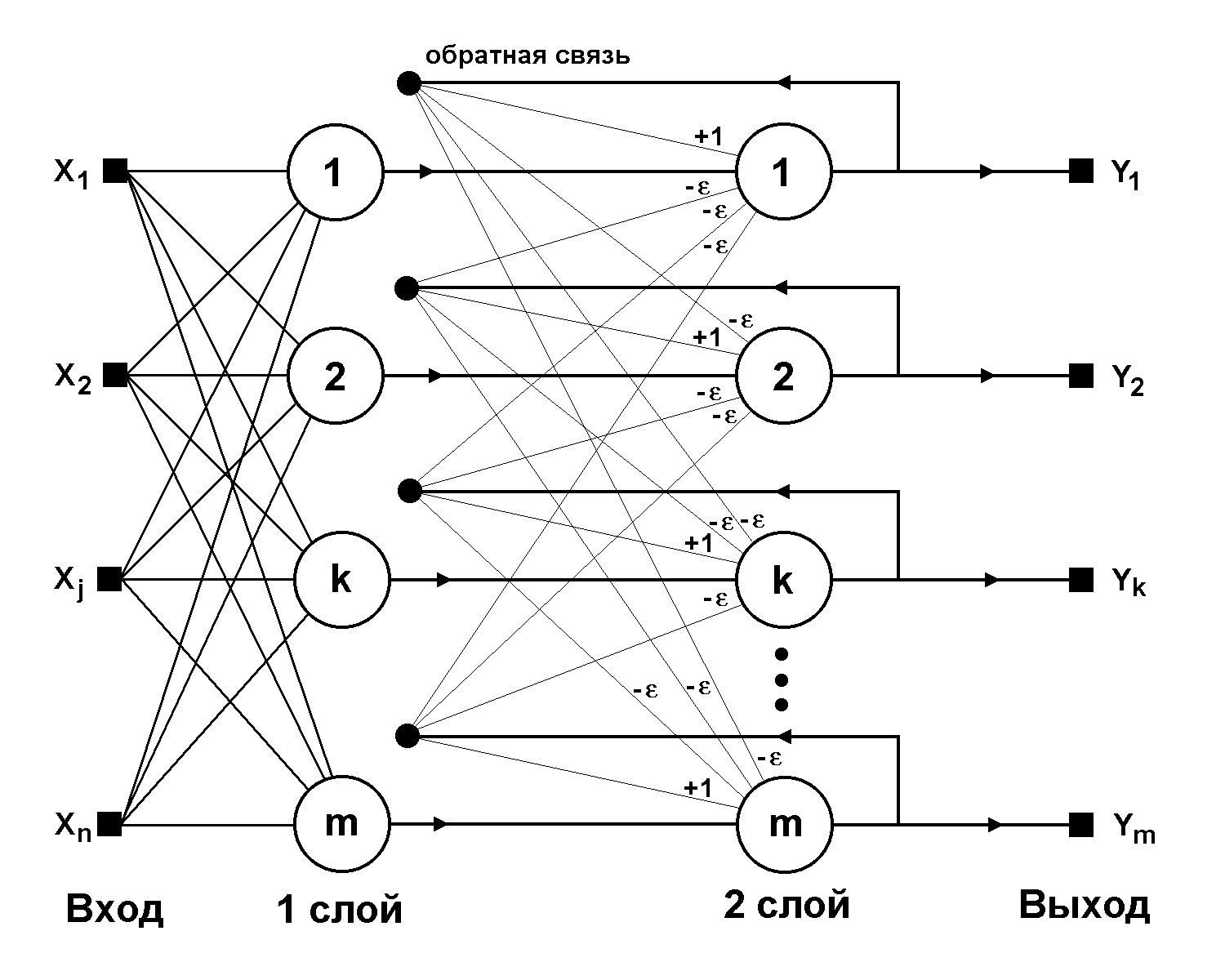


Рис. 7 Структурна схема мережі Хеммінга

Мережа складається з двох шарів. Перший і другий шари мають по m нейронів, де m – число зразків. Нейрони першого шару мають по n синапсів, з'єднаних із входами мережі (утворюючими фіктивний нульовий шар). Нейрони другого шару зв'язані між собою інгибіторними (негативними зворотними) синаптичними зв'язками. Єдиний синапс із позитивним зворотним зв'язком для кожного нейрона з'єднаний з його ж аксоном.

Ідея роботи мережі складається в перебуванні відстані Хеммінга від образу, що тестується, до всіх зразків. Відстанню Хеммінга називається число що відрізняється від бітів у двох бінарних векторах. Мережа повинна вибрати зразок з мінімальною відстанню Хеммінга до невідомого вхідного сигналу, у результаті чого буде активізований тільки один вихід мережі, що відповідає цьому зразкові.

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

, i=0...n-1, k=0...m-1, (3.28)

Tk = n / 2, k = 0...m-1. (3.29)

Тут xik – i-ий елемент k-ого зразку.

Вагові коефіцієнти гальмуючих синапсів у другому шарі беруть рівними деякої величини 0 < ε < 1/m. Синапс нейрона, зв'язаний з його ж аксоном має вагу +1.

Алгоритм функціонування мережі Хеммінга наступний:

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

, j=0...m-1. (3.30)

Після цього отриманими значеннями іниціалізуються значення аксонів другого шару:

yj(2) = yj(1), j = 0...m-1. (3.31)

2. Обчислити нові стани нейронів другого шару:

 (3.32)

і значення їх аксонів:

. (3.33)

Активаційна функція f має вигляд порога (рис. 6б), причому величина F повинна бути досить великою, щоб будь-які можливі значення аргументу не призводили до насичення.

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

З оцінки алгоритму видно, що роль першого шару досить умовна: скориставшись один раз на кроці 1 значеннями його вагових коефіцієнтів, мережа більше не звертається до нього, тому перший шар може бути узагалі виключений з мережі (замінений на матрицю вагових коефіцієнтів).

Збільшення числа і складності розпізнаваємих образів обмежується фактично тільки обсягом ОЗУ. Слід зазначити, що навчання мережі Хеммінга є одним із найпростіших серед інших алгоритмів.

Обговорення мереж, що реалізують асоціативну пам'ять, було б неповним без хоча б короткого згадування про двонаправлену асоціативну пам'ять (ДАП). Вона є логічним розвитком парадигми мережі Хопфилда, до якої для цього досить додати другий шар. Структура ДАП представлена на рис. 8.

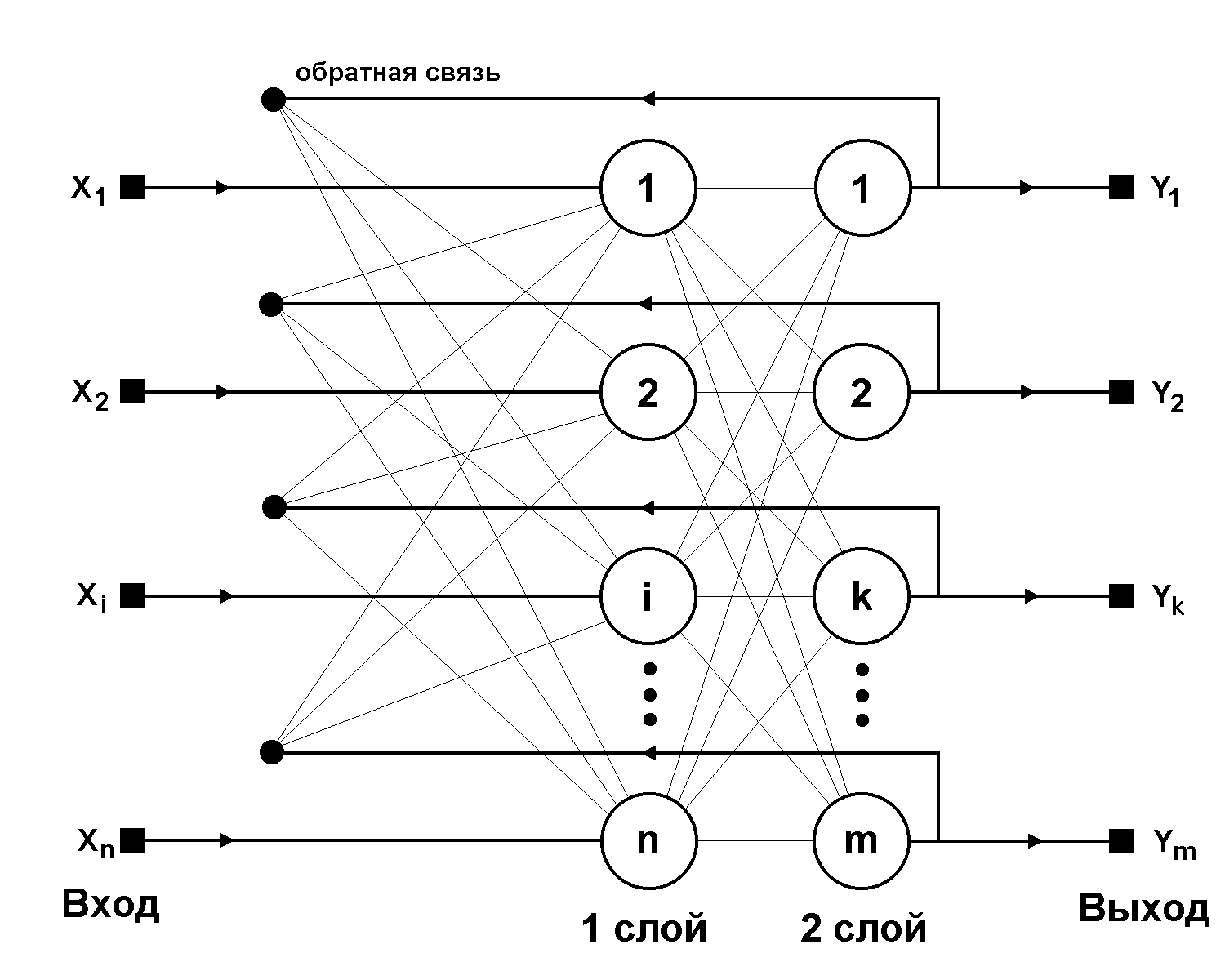


Рис. 8 Структурна схема ДАП

Мережа здатна запам'ятовувати пари асоційованих один з одним образів. Нехай пари образів записуються у вигляді векторів Xk = {xіk:і=0...n-1} і Yk = {yjk: j=0...m-1}, k=0...r-1, де r – число пар. Подача на вхід першого шару деякого вектора P = {pі:і=0...n-1} викликає утворення на вході другого шару якогось іншого вектора Q ={qj:j=0...m 1}, що потім знову надходить на вхід першого шару. При кожному такому циклі вектори на виходах обох шарів наближаються до пари зразкових векторів, перший з яких – X – найбільше походить на P, що був поданий на вхід мережі на самому початку, а другий – Y – асоційований з ним. Асоціації між векторами кодуються у ваговій матриці W(1) першого шару. Вагова матриця другого шару W(2) дорівнює транспонованої першої (W(1))T. Процес навчання, також як і у випадку мережі Хопфилда, полягає в попередньому розрахунку елементів матриці W (і відповідно WT) по формулі:

. (3.34)

Ця формула є розгорнутим записом матричного рівняння

 (3.35)

для окремого випадку, коли образи записані у вигляді векторів, при цьому добуток двох матриць розміром відповідно [n\*1] і [1\*m] приводить до (3.34).

У висновку можна зробити наступні узагальнення. Мережі Хопфилда, Хеммінга і ДАП дозволяють просто й ефективно вирішити задачу відтворення образів по неповній і перекрученій інформації. Невисока ємність мереж (число образів, що запам'ятовуються) пов’язана з тим, що, мережі не просто запам'ятовують образи, а дозволяють проводити їх узагальнення, наприклад, за допомогою мережі Хеммінга можлива класифікація за критерієм максимальної правдоподібності. Разом з тим, легкість побудови програмних і апаратних моделей роблять ці мережі привабливими для багатьох застосувань.

1. **Мережа Хебба. Алгоритм Хебба навчання нейронних мереж**

У главі 3.2 "Алгоритм навчання без вчителя (алгоритм прямого поширення нейронних мереж)" був розглянутий алгоритм Хебба як один з найбільш простих для програмної реалізації і досить ефективний для навчання НМ. Розглянемо його застосування для навчання найпростішої нейронної мережі.

Штучні нейронні мережі, призначені для вирішення різноманітних конкретних задач, можуть містити від декількох нейронів до тисяч і навіть мільйонів елементів. Однак вже окремий нейрон (рис. 1) з біполярною або бінарною функцією активації може бути використаний для вирішення простих задач розпізнавання і класифікації зображень. Вибір біполярного (1, -1) або бінарного (1, 0) представлення сигналів у нейронних мережах здійснюється виходячи з розв'язуваної задачі й у багатьох випадках він рівноцінний. Мається спектр задач, у яких бінарне кодування сигналів більш зручно, однак у більшості випадків біполярне представлення інформації має більше переваг.

Оскільки вихідний сигнал у двійкового нейрона приймає тільки два значення, то нейрон можна використовувати для класифікації запропонованих зображень на два класи.

Нехай є множина М зображень, для яких відома коректна класифікація на два класи X 1 = {X 11, X 12, …, X 1q}, X 2 = {X 21, X 22, …, X 2р},

X 1X 2 = M, X 1X 2 = Ø, та нехай першому класу X1 відповідає вихідний сигнал у = 1, а класу X 2 – сигнал у = –1. Якщо, наприклад, пред’явлено деяке зображення  і його зважена сума вхідних сигналів перевищує нульове значення:



тоді вихідний сигнал у = 1 та, відповідно, вхідне зображення X α належить класу X 1. Якщо S ≤ 0, тоді у = –1 і зображення належить другому класу.

Можливо використання окремого нейрона і для виділення з множини класів М = {X 1 = {X 11,…, X 1k}, …, X i= {X i1,…, X iq}, …, X p = {Xp1,…, X pm}} зображень єдиного класу X i. В цьому випадку вважають, що один з двох можливих вихідних сигналів нейрону (наприклад, 1) відповідає класу X i, а другий – усім іншим класам. Тому, якщо вхідне зображення Х α призводить до появи сигналу у = 1, тоді Х α ∈ X i, якщо у = –1 (або у = 0, якщо використовується бінарне кодування), то це означає, що пред’явлене зображення не належить класу, що виділяється.

Система розпізнавання на основі єдиного нейрона поділяє увесь простір можливих рішень на дві ділянки за допомогою гіперплощини

x1w1 + x2w2 + … + xnwn+w0 = 0.

Для двовимірних вхідних векторів межею між двома класами зображень є пряма лінія: вхідні вектори, розташовані вище цієї прямої, належать до одного класу, а нижче – до іншого.

Для адаптації, настроювання або навчання ваг зв'язків нейрона скористаємося "правилом Хебба".

Хебб, досліджуючи механізми функціонування центральної нервової системи, припустив, що навчання відбувається шляхом посилення зв'язків між нейронами, активність яких збігається за часом. Хоча в біологічних системах це припущення виконується далеко не завжди і не вичерпує усіх видів навчання, однак при навчанні одношарових нейромереж з біполярними сигналами воно досить ефективно.

Відповідно до правила Хебба, якщо пред'явленому біполярному зображенню X = (x1, ..., xn) відповідає неправильний вихідний сигнал у, тоді ваги wi  зв'язків нейрона адаптуються по формулі:

wi (t + 1) = wi (t) + xi у,  (4.1)

де wi (t), wi(t + 1) відповідно вага i- ого зв’язку нейрона до и після адаптації; xi () – компоненти вхідного зображення; х0 ≡ 1 – сигнал зміщення; у – вихідний сигнал нейрона.

У більш повній і строгій формі алгоритм настроювання ваг зв'язків нейрона з використанням правила Хебба виглядає таким чином:

Крок 1. Задається множина M = {(Х 1, t1), …, (X m, tm)} яка складається з пар вхідне зображення Xk =, і необхідний вихідний сигнал нейрона tk, . Ініціюються ваги зв’язків нейрона:

wi = 0, .

### Крок 2. Для кожної пари (X k, t k), доки не дотримуються умови зупинки, виконуються кроки 3 – 5.

### Крок 3. Ініціюється множина входів нейрона:

x0 = 1, xi = xik, .

### Крок 4. Ініціюється вихідний сигнал нейрона: y =t k.

### Крок 5. Корегуються ваги зв’язків нейрона по правилу

.

### Крок 6. Перевірка умов зупинки.

### Для кожного вхідного зображення X k обчислюється відповідний йому вихідний сигнал y k:



де



Якщо вектор (y1, …, ym) обчислених вихідних сигналів дорівнює вектору (t1, …, tm) заданих сигналів нейрона, тобто кожному вхідному зображенню відповідає заданий вихідний сигнал, тоді обчислення припиняються (перехід до кроку 7), якщо ж (y1, …, ym) ≠ (t1, …, tm), то перехід до кроку 2 алгоритму.

Крок 7. Зупинка.

Покажемо процес навчання нейронної мережі за Правилом Хебба на конкретних прикладах.

Приклад 1. Нехай треба навчити біполярний нейрон розпізнаванню зображень X1, та Х2, наведених на рис. 9.

При цьому зажадаємо, щоб зображенню Х1 відповідав вихідний сигнал нейрона "+1", а зображенню Х2 – сигнал "-1".

|  |  |
| --- | --- |
| X 1 | X 2 |
| |  |  |  | | --- | --- | --- | | 1 | 2 | 3 | | 4 | 5 | 6 | | 7 | 8 | 9 | | |  |  |  | | --- | --- | --- | | 1 | 2 | 3 | |  | 5 | 6 | | 7 | 8 | 9 | |
| Рис. 9. Вхідні зображення | |

Застосування алгоритму Хебба дає наступні результати:

Крок 1. Задається множина

М = {(Х1 = (1, -1, 1, 1, 1, 1,-1, -1, 1), 1), (Х2 = ( 1, 1, 1, 1, -1, 1, 1, -1, 1 ), -1)};

і ініціюються ваги зв'язків нейрона: wi = 0, 

Крок 2. Для кожної з двох пар (Х1, 1), (Х2, –1), виконуються кроки 3 – 5.

Крок 3. Ініціюється множина входів нейрона для зображення першої пари:

х0 = 1, хi = xi1, 

Крок 4. Ініціюється вихідний сигнал нейрона для зображення першої пари:

у = t1 = 1.

Крок 5. Корегуються ваги зв'язків нейрона за правилом Хебба

### 

;

;

;

;

.

Крок 3. Ініціюється множина входів нейрона для зображення Х2 другої пари:

х0 = 1, хi = хi2, 

Крок 4. Ініціюється вихідний сигнал нейрона для зображення другої пари:

(Х2, t2): у = t2 = –1.

Крок 5. Корегуються ваги зв'язків нейрона:

### ;

### ;

### ;

### ;

### ;

;

.

### Крок 6. Перевіряються умови зупинки.

### Обчислюються вхідні та вихідний сигнали нейрону при пред’явлені зображення X 1:

### y1 = 1, тому що S1 > 0.

### Обчислюються вхідні та вихідний сигнали нейрону при пред’явлені зображення Х 2



y2 = –1, тому що S2 < 0.

Оскільки вектор (y1, y2) = (1, –1) дорівнює вектору (t1, t2), то обчислення припиняються, тому що мета досягнута – нейрон правильно розпізнає задані зображення.

Крок 7. Зупинка.

Основна ідея правила (4.1) – підсилювання зв'язку, що з'єднує нейрони з однаковою за часом активністю, і послаблення зв'язку, що з'єднує елементи з різною активністю, може бути використана і при настроюванні нейромереж з бінарними елементами. Правило Хебба (4.1) для одношарових бінарних мереж можна записати у вигляді:

wi (t + 1) = wi (t) + wi , (4.2)

де

 (4.3)

Приклад 2. Нехай потрібно навчити бінарний нейрон розпізнаванню зображень Х 1 і Х 2 приклада 1. При цьому зображенню Х 1 нехай відповідає вихідний сигнал нейрона "+1", а зображенню Х2 – “0”. Застосування правила Хебба в цьому випадку дає наступні результати:

Крок 1. Задається множина

М = {( Х 1 = (1, 0, 1, 1, 1, 1, 0, 0, 1), 1), (Х2 = ( 1, 1, 1, 1, 0, 1, 1, 0, 1 ), 0)},

і ініціюються ваги зв'язків нейрона wi = 0, 

Крок 2. Для пар (Х 1,1), (Х 2, 0), виконуються кроки 3 - 5.

Крок 3. Ініціюється множина нейрона елементами зображення Х 1:

Крок 4. Ініціюється вихідний сигнал нейрона для зображення Х1:

у = t1 = 1.

Крок 5. Корегуються ваги зв'язків нейрона за допомогою співвідношень (4.2), (4.3):

;

;

;

;



Крок 3. Ініціюється множина нейрона елементами зображення Х2:

х0 = 1, хi = хi2, 

Крок 4. Ініціюється вихідний сигнал нейрона для зображення Х2:

y =t 2 = 0

Крок 5. Корегуються ваги зв'язків нейрона за допомогою співвідношень (4.2), (4.3):

### ;

### ;

;

### ;

;

;

### .

### Крок 6. Перевірка умов зупинки.

### Обчислюються вхідні і вихідні сигнали нейрону при пред’явлені зображень Х 1, Х 2:



y1 = 1, тому що S1 > 0.



y2 = –1, тому що S 2 < 0.

Оскільки вектор (y1, y2) = (1, 0) дорівнює заданому вектору (t1, t2) = (1, 0), то мета досягнута та обчислення припиняються.

Крок 7. Зупинка.

Використання групи з m біполярних або бінарних нейронів A1, ..., Am (рис. 10) дозволяє істотно розширити можливості нейронної мережі і розпізнавати до 2m різних зображень. Правда, застосування цієї мережі для розпізнавання 2m (або близьких до 2m чисел) різних зображень може призводити до нерозв'язних проблем адаптації ваг зв'язків нейромережі. Тому часто рекомендують використовувати дану архітектуру для розпізнавання тільки m різних зображень, задаючи кожному з них одиничний вихід тільки на виході одного А-елемента (виходи інших при цьому повинні приймати значення "-1" для біполярних нейронів або "0" – для бінарних).

*wni*

⋅ ⋅ ⋅

⋅ ⋅ ⋅

⋅ ⋅ ⋅

⋅ ⋅ ⋅

*y*1

*Ат*

*Аj*

*A*1

*х*1

*w*11

*w*1*i*

*wjm*

*wnm*

Рис 10. Нейронна мережа з *m* елементів

*w*1*m*

*wj*1

*wji*

*yi*

*ym*

*wn*1

*xj*

*xn*

Одношарова нейронна мережа з двійковими нейронами, приведена на рис. 10, може бути навчена за допомогою алгоритму на основі правила Хебба. У цьому випадку вона називається мережею Хебба. Використання інших алгоритмів навчання цієї ж мережі призводить і до зміни назви нейронної мережі. Використання в назві мереж їхніх алгоритмів навчання характерно для теорії нейронних мереж. Для біполярного представлення сигналів можливе навчання нейромережі за допомогою наступного алгоритму:

Крок 1. Задається множина M = {(Х 1, t1), …, (Х m, t m)}, яка складається з пар вхідне зображення X k = , і необхідний вихідний сигнал нейрона t k, . Ініціюються ваги зв’язків нейрона:

wji = 0, , 

Крок 2. Кожна пара (Х k, t k), перевіряється на правильність реакції нейронної мережі на вхідне зображення. Якщо отриманий вихідний вектор мережі   
(, …, ), відрізняється від заданого t1 = (, …, ), то виконують кроки 3–5.

Крок 3. Ініціюється множина входів нейронів: x0 = 1, xj = xjk, 

Крок 4. Ініціюються вихідні сигнали нейронів: yi =  .

Крок 5. Корегуються ваги зв'язків нейронів по правилу:

.

Крок 6. Перевіряються умови зупинки, тобто правильності функціонування мережі при пред’явлені кожного вхідного зображення. Якщо умови не виконуються, то перехід до кроку 2 алгоритму, інакше – припинення обчислень (перехід до кроку 7).

Крок 7. Зупинка.

Таким чином у даній главі докладно продемонстрований порядок створення і навчання нейронної мережі Хебба на конкретних елементарних прикладах. Показано, що навіть найпростіша мережа, що складається з одного нейрона здатна виконувати корисну роботу по розпізнаванню одного з двох зображень.

1. **Області вживання і задачі РОЗВ'ЯЗУВАНІ за допомогою нейронних мереж**

У літературі зустрічається значне число ознак, якими повинна володіти задача, щоб застосування НМ було виправдано і НМ могла б її вирішити:

* відсутній алгоритм або не відомі принципи вирішення задачі, але накопичене достатнє число прикладів;
* проблема характеризується великими обсягами вхідної інформації;
* дані неповні або надлишкові, зашумлені, частково суперечливі.

Таким чином, НМ добре підходять для розпізнавання образів і вирішення задач класифікації, оптимізації і прогнозування. Нижче приведений перелік можливих промислових застосувань нейронних мереж, на базі яких або вже створені комерційні продукти, або реалізовані демонстраційні прототипи.

Банки і страхові компанії:

* автоматичне зчитування чеків і фінансових документів;
* перевірка вірогідності підписів;
* оцінка ризику для позик;
* прогнозування змін економічних показників.

Адміністративне обслуговування:

* автоматичне зчитування документів;
* автоматичне розпізнавання штрихових кодів.

Нафтова і хімічна промисловість:

* аналіз геологічної інформації;
* ідентифікація несправностей устаткування;
* розвідка покладів мінералів за даними аерофотознімань;
* аналіз складів домішок;
* керування процесами.

Військова промисловість і аеронавтика:

* обробка звукових сигналів (поділ, ідентифікація, локалізація, усунення шуму, інтерпретація);
* обробка радарних сигналів (розпізнавання цілей, ідентифікація і локалізація джерел);
* обробка інфрачервоних сигналів (локалізація);
* узагальнення інформації;
* автоматичне пілотування.

Промислове виробництво:

* керування маніпуляторами;
* керування якістю;
* керування процесами;
* виявлення несправностей;
* адаптивна робототехніка;
* керування голосом.

Служба безпеки:

* розпізнавання осіб, голосів, відбитків пальців.

Біомедична промисловість:

* аналіз рентгенограм;
* виявлення відхилень у ЕКГ

Телебачення і зв'язок:

* адаптивне керування мережею зв'язку;
* стиск і відновлення зображення.

Представлений перелік далеко не повний. Щомісяця західні засоби масової інформації повідомляють про нові комерційні продукти на базі нейронних мереж. Так, фірма LІAC випускає апаратуру для контролю якості води. Нейросистеми фірми SAІ знаходять пластикові бомби в багажі авіапасажирів. Фахівці інвестиційного банку Cіtіcomp (Лондон) за допомогою програмного нейропакету роблять короткострокові прогнози коливань курсів валют.

1. **Сучасні проекти і вироби, засновані на нейронних мережах**

Нейронні мережі можуть бути реалізовані двома шляхами: перший це програмна модель НМ, другий - апаратна. На сучасному ринку виробу, засновані на використанні механізму дії НМ, спочатку з'явилися у вигляді нейроплат. Як типовий приклад нейроплати можна назвати плату МВ 86232 японської фірми Fujіtsu. На платі розміщений процесор цифрової обробки сигналів та оперативна пам'ять ємністю 4 Мбайт, що дозволяє використовувати таку плату для реалізації НМ, що містять до тисячі нейронів.

Основними комерційними апаратними виробами на основі НМ є і, імовірно, найближчим часом будуть залишатися нейробіс. Зараз випускаються більш 20 типів нейробісів, параметри яких часом розрізняються на декілька порядків. Серед них – модель ETANN фірми Іntel. Ця БІС, виконана за мікронною технологією, є реалізацією НМ із 64 нейронами і 10240 синапсами. Її ціна 2000 дол. До числа найдешевших нейробис (41 дол.) відноситься модель MD 1220 фірми Mіcro Devіces. Ця БІС реалізує НМ із 8 нейронами і 120 синапсами.

Серед розроблювальних у даний час нейробіс виділяються моделі фірми Adaptіve Solutіons (США) і Hіtachі (Японія). Нейробіс фірми Adaptіve Solutіons, імовірно, стане однією із самих швидкодіючих: оголошена швидкість обробки складає 1,2 млрд. з'єднань / с. (НМ містить 64 нейрона і 262144 синапса). Нейробіс фірми Hіtachі дозволяє реалізувати НМ, що містить до 576 нейронів. Ці нейробіс, безсумнівно, стануть основою нових нейрокомп’ютерів і спеціалізованих багатопроцесорних виробів.

Більшість сьогоднішніх нейрокомп’ютерів являють собою просто персональний комп'ютер або робочу станцію, до складу яких входить додаткова нейроплата. До їхнього числа відносяться, наприклад, комп'ютери серії FMR фірми Fujіtsu. Такі системи мають безперечне право на існування, оскільки їхніх можливостей цілком достатньо для розробки нових алгоритмів і вирішення великого числа прикладних задач методами нейроматематики. Однак найбільший інтерес представляють спеціалізовані нейрокомп’ютери, що безпосередньо реалізують принципи НМ. Типовими представниками таких систем є комп'ютери сімейства Mark фірми TRW (перша реалізація персептрона, розроблена Розенблатом, називалася Mark І). Модель Mark ІІІ фірми TRW являють собою робочу станцію, що містить до 15 процесорів сімейства Motorola 68000 з математичними співпроцесорами. Усі процесори об'єднані шиною VME. Архітектура системи, що підтримує до 65 000 віртуальних процесорних елементів з більш ніж 1 млн. з'єднань, що набудовуються, дозволяє обробляти до 450 тис. міжз’єднань / с. Mark ІV – це однопроцесорний суперкомп'ютер з конвеєрною архітектурою. Він підтримує до 236 тис. віртуальних процесорних елементів, що дозволяє обробляти до 5 млн. міжз’єднань / с. Комп'ютери сімейства Mark мають загальну програмну оболонку ANSE (Artіfіcіal Neural System Envіronment), що забезпечує програмну сумісність моделей. Крім зазначених моделей фірми TRW пропонує також пакет Mark ІІ - програмний эмулятор НМ.

Іншою цікавою моделлю є нейронкомп'ютер NETSІ, створений фірмою Texas Іnstruments на базі розробок Кембріджського університету. Його топологія являє собою тривимірні ґрати стандартних обчислювальних вузлів на базі процесорів 80188. Комп'ютер NETSІ використовується для моделювання таких моделей НМ, як мережа Хопфилда - Кохонена і НМ зі зворотнім поширенням. Його продуктивність досягає 450 млн. міжз’єднань / с.

Фірма Computer Recognіtііon Systems (CRS) продає серію нейрокомп’ютерів WІZARD/CRS 1000, призначених для обробки відеозображень. Розмір вхідного зображення 512 x 512 пікселей. Модель CRS 1000 вже знайшла застосування в промислових системах автоматичного контролю.

Сьогодні на ринку представлено багато моделей нейрокомп’ютерів. Насправді їх, очевидно, набагато більше, але найбільш могутні і перспективні моделі як і раніше створюються за замовленнями військових. Нажаль, не маючи достатньої інформації про моделі спеціального призначення, важко скласти уявлення про істинні можливості сучасних комп'ютерів.

{Програмна реалізація нейронної мережі Хебба}

unit rou\_main;

interface

uses

Windows, Messages, SysUtils, Variants, Classes,

Graphics, Controls, Forms,Dialogs, ExtCtrls,

StdCtrls, Grids, Buttons;

const

yes='y'; //Стан активності/не активності

no='n'; //комірки елемента TStringGrid

str=10; //Розміри нейрону

stol=10;

type

{Користувацькі типи даних}

one\_element= array [0..str\*stol+4] of integer; //Опис нейрона

W\_type= array [0..3,0..str\*stol] of integer; //Опис нейронної мережі

S\_type=array [0..3] of integer; //Масив сумарних

//вихідних результатів

TForm1 = class(TForm)

{Опис елементів форми}

Panel1: TPanel;

s1: TStringGrid; //Поле першого нейрону

l1: TLabel;

Bevel1: TBevel;

Memo1: TMemo; //Інформаційне поле

Label2: TLabel;

s2: TStringGrid; //Поле другого нейрону

l2: TLabel;

s3: TStringGrid; //Поле третього нейрону

l3: TLabel;

s4: TStringGrid; //Поле четвертого нейрону

l4: TLabel;

s5: TStringGrid; //Поле для символів, що розпізнаються

Label6: TLabel;

BitBtn1: TBitBtn; //Кнопка "Teaching"

BitBtn2: TBitBtn; //Кнопка "Find"

BitBtn3: TBitBtn; //Кнопка "Exit"

BitBtn14: TBitBtn; //Кнопка "Clean"

BitBtn8: TBitBtn; //Кнопка "Clean"

BitBtn11: TBitBtn; //Кнопка "Clean"

BitBtn15: TBitBtn; //Кнопка "Clean"

BitBtn18: TBitBtn; //Кнопка "Clean"

BitBtn4: TBitBtn; //Кнопка "Load"

BitBtn5: TBitBtn; //Кнопка "Save"

{Процедури і функції основної форми}

procedure s1DrawCell(Sender: TObject; ACol, ARow: Integer;

Rect: TRect; State: TGridDrawState);

procedure FormCreate(Sender: TObject);

procedure s1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

procedure BitBtn1Click(Sender: TObject);

procedure Initialization\_M;

procedure Vivod\_W (W:W\_type);

procedure BitBtn2Click(Sender: TObject);

procedure BitBtn4Click(Sender: TObject);

procedure BitBtn5Click(Sender: TObject);

procedure BitBtn14Click(Sender: TObject);

procedure BitBtn8Click(Sender: TObject);

procedure BitBtn11Click(Sender: TObject);

procedure BitBtn15Click(Sender: TObject);

procedure BitBtn18Click(Sender: TObject);

end;

{Допоміжні процедури і функції}

procedure Raschet\_b1;

function convert (s: string): integer;

procedure Read\_str\_grd (str\_grd:TStringGrid;var M:one\_element);

function Check\_result (num\_letter,num\_nero:integer):boolean;

procedure Obuchenie(num\_letter,num\_nero:byte);

function Check\_Learning :boolean;

procedure checkFillM5(Y:integer);

procedure clear\_stringgrid (Sender:Tobject);

function mask\_el: W\_type;

procedure re\_mask\_el(mask\_elem: W\_type);

{Основні змінні}

var

Form1 :TForm1; //Змінна форми

M :array [0..4] of one\_element; //Масив, що описує нейрони мережі

W :W\_type; //Опис нейронної мережі

prizn :boolean;

S :S\_type;

i :integer;

f :file of W\_type; //Змінна файлу для збереження

//стану нейронної мережі

implementation

{$R \*.dfm}

{Процедура малювання форми}

procedure TForm1.s1DrawCell(Sender: TObject; ACol, ARow: Integer;

Rect: TRect; State: TGridDrawState);

begin

(sender as tstringgrid).canvas.Brush.Color:=$a08000;

if (sender as tstringgrid).Cells[Acol,Arow]=yes then

(sender as tstringgrid).canvas.FillRect(Rect)

else begin

(sender as tstringgrid).canvas.Brush.Color:=clCream;

(sender as tstringgrid).canvas.FillRect(Rect)

end;

end;

{Процедура створення форми}

procedure TForm1.FormCreate(Sender: TObject);

begin

prizn:=false;

clear\_stringgrid (s1); //Очищення поля нейрону 1

clear\_stringgrid (s2); //Очищення поля нейрону 2

clear\_stringgrid (s3); //Очищення поля нейрону 3

clear\_stringgrid (s4); //Очищення поля нейрону 4

clear\_stringgrid (s5); //Очищення поля нейрону 5

end;

{Процедура активації комірки нейрону}

procedure TForm1.s1MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: Integer);

begin

if (sender as tstringgrid).Cells[(sender as tstringgrid).Col,(sender as tstringgrid).Row]=yes then

(sender as tstringgrid).Cells[(sender as tstringgrid).Col,(sender as tstringgrid).Row]:=no

else (sender as tstringgrid).Cells[(sender as tstringgrid).Col,(sender as tstringgrid).Row]:=yes;

end;

{Процедура навчання нейронної мережі}

procedure TForm1.BitBtn1Click(Sender: TObject);

var i,j:integer;

begin

Memo1.Clear; //Очищення інформаційного поля

Initialization\_M; //Ініціалізація масиву W для навчання

Memo1.Lines.Add('Initialization of the weight ratio of neuron connections');

Vivod\_W (W); //Вивід масиву W

Raschet\_b1; //Розрахунок важелів зв’язків нейронів

//для першого нейрону

Memo1.Lines.Add('Calculation the firs simbol:');

Vivod\_W (W); //Вивід масиву W

{Навчання першого нейрону}

Check\_result (0,0);

Check\_result (0,1);

Check\_result (0,2);

Check\_result (0,3);

{Навчання інших нейронів}

for i:=1 to 3 do

for j:= 0 to 3 do

begin

while not Check\_result (i,j) do

Obuchenie (i,j); //Навчання нейрону доки не отримано

//жаданий результат

end;

Memo1.Lines.Add('Result weight ratio ');

Vivod\_W (W); //Вивід масиву W

if Check\_Learning then Memo1.Lines.Add('Teaching is successful.')

else Memo1.Lines.Add('Teaching is impossible.');

BitBtn1.Enabled:=false; //Кнопка "Teaching" неактивна

BitBtn2.enabled:=true; //Кнопка "Find" активна

BitBtn5.enabled:=true; //Кнопка "Save" активна

end;

{Функція інтерпретації активності комірки нейрона}

function convert (s: string): integer;

begin

if s=yes then Result:=1

else Result:=-1;

end;

{Функція початкової ініціалізації масиву W}

procedure TForm1.Initialization\_M;

var

str1 :string;

j,i :byte;

begin

{Зчитуємо зображення навчальних символів}

Read\_str\_grd (s1,M[0]);

Read\_str\_grd (s2,M[1]);

Read\_str\_grd (s3,M[2]);

Read\_str\_grd (s4,M[3]);

M[0,str\*stol+1]:=1;

M[1,str\*stol+2]:=1;

M[2,str\*stol+3]:=1;

M[3,str\*stol+4]:=1;

Memo1.Lines.Add('Giving of the set M:');

{Вивід на екран зчитаних даних}

for j:= 0 to length (M)-2 do

begin

for i:= 0 to length(M[j])-1 do str1:=str1+' '+inttostr(m[j,i]);

Memo1.Lines.Add(str1);

str1:='';

end;

end;

{Конвертація зображення на нейроні в числові дані}

procedure Read\_str\_grd (str\_grd:TStringGrid;var M:one\_element);

var i,j:integer;

begin

for i:=0 to str-1 do

for j:=0 to stol-1 do

M[i\*stol+j+1]:=convert(str\_grd.Cells[j,i]); //Інтерпретація значення

//комірок нейрона

M[0]:=1;

for i:= str\*stol+1 to str\*stol+4 do

M[i]:=-1;

end;

{Процедура виводу на екран вмісту масиву W}

procedure TForm1.Vivod\_W (W:W\_type);

var

i,j :byte;

str :string;

begin

for j:=0 to length (W)-1 do

begin

Form1.Memo1.Lines.Add('W'+inttostr(j+1)+':');

for i:=0 to length (W[j])-1 do str:=str+ ' '+inttostr (W[j,i]);

Form1.Memo1.Lines.Add(str);

str:='';

end;

end;

{Процедура розрахунку важелів зв’язків нейронів для першого нейрону}

procedure Raschet\_b1;

var

i,j:byte;

begin

for j:=0 to length(W)-1 do

begin

for i:=0 to str\*stol do

W[j,i]:=W[j,i]+M[0,i]\*M[0,str\*stol+J+1];

end;

end;

{Функція визначення правильності навчання}

function Check\_result (num\_letter,num\_nero:integer):boolean;

var

i :integer;

stroke :string;

begin

s[num\_nero]:=0;

for i:=0 to str\*stol do

begin

s[num\_nero]:=s[num\_nero]+W[num\_nero,i]\*M[num\_letter,i];

end;

stroka:='Sending the char '+inttostr(num\_letter+1)+' to the

'+inttostr(num\_nero+1)+' neuron';

Form1.Memo1.Lines.Add(stroka);

Form1.Memo1.Lines.Add('S'+inttostr(num\_nero+1)+'='+inttostr(s[num\_nero]));

if s[num\_nero]<=0 then s[num\_nero]:=-1

else s[num\_nero]:=1;

if s[num\_nero]= M[num\_letter,str\*stol+num\_nero+1] then Result:=true

else begin

Result:=false;

Form1.Memo1.Lines.Add ('Incorrect result. Reteaching.');

end;

end;

{Процедура навчання нейрона}

procedure Obuchenie(num\_letter,num\_nero:byte);

var

p: integer;

begin

for p:=0 to str\*stol do

W[num\_nero,p]:=W[num\_nero,p]+

M[num\_letter,p]\*M[num\_letter,str\*stol+num\_nero+1];

end;

{Функція визначення правильності навчання нейронної мережі}

function Check\_Learning :boolean;

var

res :array[0..3] of boolean;

s :array[0..3] of integer;

i,j,max,Y :integer;

s1 :string;

begin

{Подання на вхід мережі навчальних символів

і перевірка вірності реакції мережі на них}

for y:=0 to 3 do

begin

res[Y]:=false;

checkFillM5(Y);

for j:=0 to 3 do

begin

s[j]:=0;

for i:=0 to str\*stol do

begin

s[j]:=s[j]+W[j,i]\*M[4,i];

end;

end;

s1:='Identification of the Char ';

max:=s[0];

for i:=1 to 3 do

begin

s1:='S= '+inttostr(s[i]);

if s[i]>max then

begin

max:=s[i];

end;

end;

if max>0 then res[Y]:=true;

end;

result:=true;

{Перевірка чи на всі навчальні зображення мережа реагує вірно}

for i:=0 to 3 do

if not res[i] then

begin

result:=false;

break;

end;

end;

{Процедура вибору символа}

procedure checkFillM5(Y:integer);

var

i: integer;

begin

for i:=1 to str\*stol do M[4,i]:=M[y,i]

end;

{Процедура визначення символу відповідно до навчальних зображень}

procedure TForm1.BitBtn2Click(Sender: TObject);

var

s: array [0..3] of integer;

i,j,k: byte;

max: integer;

strng: string;

begin

Memo1.Lines.Add('Identification of the Char ');

l1.Color:=clSilver;

l2.Color:=clSilver;

l3.Color:=clSilver;

l4.Color:=clSilver;

Read\_str\_grd (s5,M[4]); //Зчитуємо зображення символу

for j:=0 to 3 do

begin

s[j]:=0;

for i:=0 to str\*stol do

begin

s[j]:=s[j]+W[j,i]\*M[4,i]; //Подання символу на нейрони і

//визначення результуючих сум

end;

end;

max:=s[0];

k:=0;

for i:=0 to 3 do

begin

strng:=strng+'S'+inttostr(i+1)+'='+inttostr(s[i])+' ';

if s[i]>max then

begin

max:=s[i]; //Визначення максимальної суми

k:=i; //Визначення індексу вірного зображення

end;

end;

Memo1.Lines.Add(strng);

Memo1.Lines.Add('The char '+inttostr (K+1)+' has been identified.');

case (k+1) of //Позначення червоним кольором

//символу, що визначений як вірний

1:l1.Color:=clRed;

2:l2.Color:=clRed;

3:l3.Color:=clRed;

4:l4.Color:=clRed;

end;

BitBtn1.Enabled:=true; //Кнопка "Teaching" активна

end;

{Функція отримання маски навчальних символів}

function mask\_el: W\_type;

var

i,j :integer;

mask\_elem :W\_type;

begin

for i:=0 to str-1 do

for j:=0 to stol-1 do

begin

{Конвертація зображень навчальних символів у числові значення}

mask\_elem[0,i\*stol+j]:=convert(Form1.s1.Cells[j,i]);

mask\_elem[1,i\*stol+j]:=convert(Form1.s2.Cells[j,i]);

mask\_elem[2,i\*stol+j]:=convert(Form1.s3.Cells[j,i]);

mask\_elem[3,i\*stol+j]:=convert(Form1.s4.Cells[j,i]);

end;

Result:=mask\_elem;

end;

{Процедура збереження в файлі "symbol.rou" стану нейронної мережі}

procedure TForm1.BitBtn5Click(Sender: TObject);

var

W1 :W\_type;

begin

w1:=mask\_el;

AssignFile(f, 'symbol.rou');

Rewrite(f); //Відкриття файлу для запису даних

write(f,W); //Запис стану нейронної мережі

write(f,W1); //Запис зображення навчальних символів

CloseFile(f);

Form1.Memo1.Lines.Add('Teaching Configuration data has been saved'+

' in data file symbol.rou');

MessageDlg('Teaching Configuration data has been saved'+

' in data file symbol.rou', mtInformation, [mbOk], 0);

BitBtn4.enabled:=true; //Кнопка "Load" активна

end;

{Процедура активації нейронів мережі згідно із зображенням}

procedure re\_mask\_el(mask\_elem: W\_type);

var

i,j :integer;

begin

{Очищення нейронів}

clear\_stringgrid(Form1.s1);

clear\_stringgrid(Form1.s2);

clear\_stringgrid(Form1.s3);

clear\_stringgrid(Form1.s4);

for i:=0 to str-1 do

for j:=0 to stol-1 do

begin

{Запис нового зображення в нейрони}

if mask\_elem[0,i\*stol+j]=1 then Form1.s1.Cells[j,i]:=yes;

if mask\_elem[1,i\*stol+j]=1 then Form1.s2.Cells[j,i]:=yes;

if mask\_elem[2,i\*stol+j]=1 then Form1.s3.Cells[j,i]:=yes;

if mask\_elem[3,i\*stol+j]=1 then Form1.s4.Cells[j,i]:=yes;

end;

end;

{Процедура завантаження стану нейронної мережі із файлу "symbol.rou"}

procedure TForm1.BitBtn4Click(Sender: TObject);

var

W1 :W\_type;

begin

AssignFile(f, 'symbol.rou');

FileMode := 0; //Режим читання

{$I-}

Reset(f); //Відкриття файлу для читання

{$I+}

if IOResult = 0 then //Файл знайдено

begin

MessageDlg('Data from data file symbol.rou has been just loaded',

mtInformation, [mbOk], 0);

read(f,w); //Читання стану нейронної мережі

read(f,W1); //Читання зображень навчальних символів

re\_mask\_el(W1); //Активація нейронів, згідно із

//зображенням навчальних символів

CloseFile(f);

end

else //Файл не знайдено

begin

MessageDlg('File access error: Data File symbol.rou is epcent',

mtWarning, [mbOk], 0);

Memo1.Lines.Add('Data File symbol.rou is epcent');

end;

Form1.BitBtn2.Enabled:=true; //Кнопка "Find" активна

end;

{Процедура очищення поля нейрона}

procedure clear\_stringgrid (Sender:Tobject);

var

i,j :byte;

begin

for i:=0 to (sender as TStringGrid).ColCount-1 do

for j:=0 to (sender as TStringGrid).RowCount-1 do

(sender as TStringGrid).Cells[i,j]:=no;

end;

{Очищення нейрону 1}

procedure TForm1.BitBtn14Click(Sender: TObject);

begin

clear\_stringgrid (s1);

end;

{Очищення нейрону 2}

procedure TForm1.BitBtn8Click(Sender: TObject);

begin

clear\_stringgrid (s2);

end;

{Очищення нейрону 3}

procedure TForm1.BitBtn11Click(Sender: TObject);

begin

clear\_stringgrid (s3);

end;

{Очищення нейрону 4}

procedure TForm1.BitBtn15Click(Sender: TObject);

begin

clear\_stringgrid (s4);

end;

{Очищення поля символу, що розпізнається}

procedure TForm1.BitBtn18Click(Sender: TObject);

begin

clear\_stringgrid (s5);

end;

end.