ЛАБОРАТОРНА РОБОТА 2
НЕЙРОННА МЕРЕЖА КОХОНЕНА

2.1. Мета лабораторної роботи

Отримання й закріплення знань, формування практичних навичок роботи з нейронною мережею Кохонена.

2.2. Короткі теоретичні відомості

2.2.1. Нейронна мережа Кохонена

На відміну від нейронної мережі Хеммінга, дослідженої в попередній лабораторній роботі і віднесеної до класу мереж, що змагаються, у яких ваги зв'язків залишаються фіксованими в процесі їхнього функціонування, мережа Кохонена – представник іншого класу конкуруючих мереж або мереж, що змагаються, у яких ваги зв'язків змінюються в ході ітераційного процесу виділення нейронів-переможців.

Структура цієї нейронної мережі, що має й інші назви – відображення, що самоорганізується, Кохонена, топологічно зберігаюче перетворення та карти ознак Кохонена, що самоорганізуються, була запропонована Кохоненом у 1982 році. Відмінна риса цієї мережі – відображати вхідну інформацію, зберігаючи відношення сусідніх вхідних елементів, тобто зберігаючи її топологічну структуру. Ця властивість характерна мозку, але використовується лише в деяких нейронних мережах, хоча вона часто буває необхідною при встановленні характеру взаємозв'язків, які людським оком важко вловлюються. У цьому випадку можливе застосування відображення Кохонена, що дозволяє від важко сприйманих людським зором взаємозв'язків елементів перейти, наприклад, до впорядкованого розташування елементів на прямокутній, гексагональній або будь-який іншій відповідній сітці. Це широко використовується для перетворення багатовимірних вихідних даних в одно- або двовимірні карти ознак Кохонена (або карти Кохонена, що самоорганізуються).

Розглянемо інший можливий ефективний випадок застосування перетворення Кохонена. Є безліч об'єктів, частина з яких дуже схожа одна

на одну, а інша їх частина не схожа або схожа відносно. Необхідно виконати класифікацію таким чином, щоб було наочно видно, яким образом групуються об'єкти й наскільки вони близькі або далекі один від одного.

Можливе також застосування відображення Кохонена для перетворення багатовимірних вихідних даних в одно- або двовимірні "карти ознак Кохонена", які можна розглядати як результат передобробки для інших нейронних мереж.

Структура мережі Кохонена наведена на рис. 2.1.

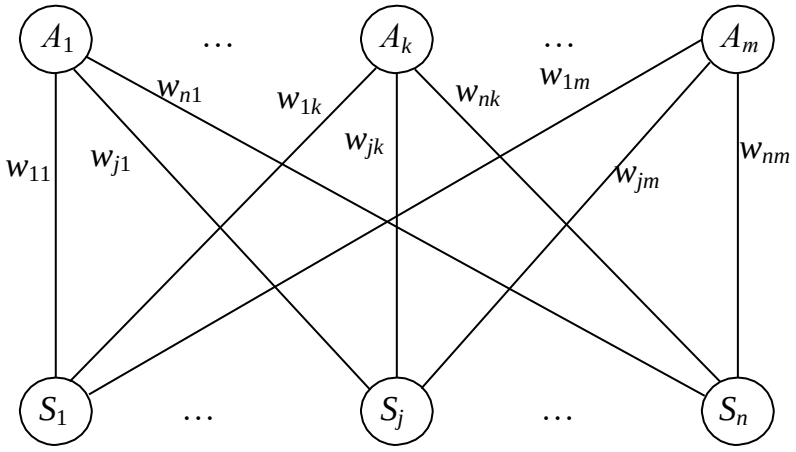

Рис. 2.1. Мережа Кохонена

Мережа має два шари нейронів. Нейрони першого шару сприймають вхідну інформацію у вигляді *n*-мірних безперервних векторів та передають її *A*-нейронам, які впорядковані в одно- або двовимірному масиві як елементи деяких кластерів. Під час процесу самоорганізації при пред'явленні вектора деякого вхідного зображення виділяється *A*-елемент кластера, що у сенсі мінімуму заданої відстані (наприклад, квадрата евклідової відстані) найбільше відповідає цьому вхідному вектору. Цей виділений або перемігший елемент та його найближчі сусіди (або в термінах топології – покриття елемента) за заданим правилом змінюють свої ваги, щоб ще краще відповідати вхідному вектору, тобто елемент переможець та його покриття в деякому сенсі близькі до вхідного вектора й прагнуть цю близькість збільшити.

Приклад покриття елемента A_j для одновимірного випадку наведено на рис. 2.2, а для двовимірного випадку – на рис. 2.3.

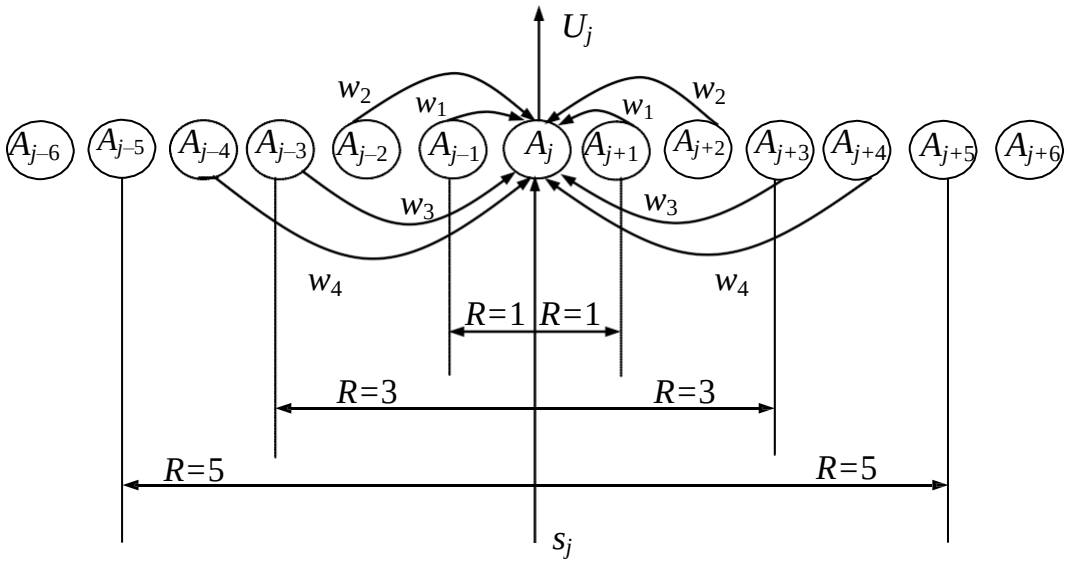

Рис. 2.2. Схема взаємодії нейронів у одновимірному випадку

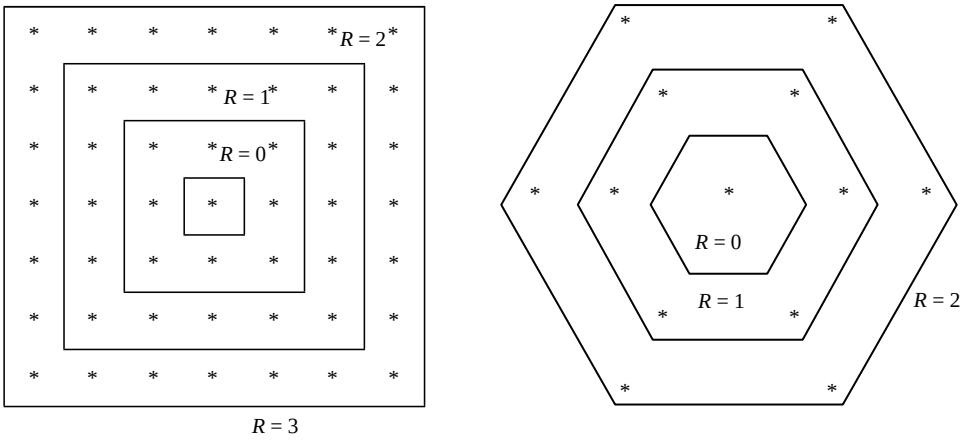

Рис. 2.3. Топологія оточення нейрона у двовимірному випадку

На рис. 2.2 покриття елемента A_j з радіусом R = 1 включає, крім A_j, усього два елементи: A_{j-1} та A_{j+1}, а покриття з R = 5 – десять елементів: A_{j-5}, ..., A_{j-1}, A_{j+1}, ..., A_{j+5}. У двовимірному випадку (рис. 2.3) при прямокутних ґратах та R = 1 у покриття входять вісім елементів, найближчих до виділеного, а у випадку гексагональних ґрат – шість. Перемігші елементи, що розташовані близько до краю ґрат, мають менше число найближчих сусідів, відсутні сусіди в процесі функціонування мережі просто ігноруються.

2.2.2. Алгоритм навчання мережі Кохонена

Крок 1. Ініціюються ваги w_{jk} $(j=\overline{1,n},\ k=\overline{1,m})$ зв'язків мережі випадковими числами з інтервалу [0, 1]. Задається множина, що визначає топологічну близькість елементів, і навчальний коефіцієнт α.

Крок 2. Поки не виконуються умови зупину, реалізуються кроки 3 – 9 алгоритму.

Крок 3. Для кожного вхідного вектора $S^p=(s_1^p,\ ...,s_n^p),\ p=\overline{1,L}$ виконуються кроки 4 – 6.

Крок 4. Для кожного нейрона A_j обчислюється відстань

$$D(A_j)=\sum_{i=1}^{n}(w_{ij}-s_i^p)^2.$$

Крок 5. Визначається нейрон A_j, для якого відстань $D(A_j)$ мінімальна.

Крок 6. Для всіх A-елементів у межах заданого окілу R елемента A_j та для всіх i виконується зміна ваг зв'язків:

$$w_{ij}(new)=w_{ij}(old)+\alpha(s_i^p-w_{ij}(old)),\quad i=\overline{1,n}.$$

Крок 7. Модифікується навчальний коефіцієнт α.

Крок 8. Зменшується радіус топологічної близькості як функція часу.

Крок 9. Перевіряються умови зупинки.

Крок 10. Зупинка.

Зауваження 2.1. Розглянемо деякі можливі альтернативні зміни в структурі алгоритму.

Вихідним вагам у наведеному алгоритмі задаються випадкові значення. Однак якщо є деяка апріорна інформація про розташування або розподіл кластерів, що стосується конкретної досліджуваної проблеми, то її можна й бажано врахувати у вихідних вагах зв'язків нейронів.

Можливі структури алгоритму з різними законами зміни радіуса R окілу найближчих елементів, проте кожного разу радіус повинен зменшуватися в міру прогресу процесу кластеризації. Аналогічно навчальний коефіцієнт α повинен бути спадною функцією часу або

періодів навчання. Кохонен показав, що лінійного зменшення α досить для практичних додатків, однак сам процес навчання мережі може бути досить тривалим.

Можливими умовами закінчення ітераційного процесу можуть бути: задане число ітерацій (пред'явлень вхідної множини зображень) або зменшення навчального коефіцієнта α до наперед заданого значення, або зміни величин елементів матриці ваг між двома ітераціями, що менші від заданого значення ε й т. д.

Приклад 2.1. Застосування перетворення Кохонена, що самоорганізується, для кластеризації чотирьох векторів (0 0 0 1), (0 0 1 1), (1 0 0 0), (1 1 0 0) на два класи.

Оскільки вектори 4-мірні, то число вхідних *S*-нейронів дорівнює 4, а оскільки задано два класи, то $m = 2$ та число *A*-нейронів дорівнює 2. Звідси також виходить, що покриття нейронів складається тільки із самого нейрона, тобто радіус околу нейронів R дорівнює нулю. Припустимо, що навчальний коефіцієнт α змінюється як функція часу:

$$\alpha(t+1) = k\alpha(t),$$

де $k = 0{,}5$ та $\alpha(0) = 0{,}6$.

Використовуємо наведений вище алгоритм для розв'язання данної задачі кластеризації.

Крок 1. Ініціюється матриця M_q ваг зв'язків випадково обраними числами з інтервалу [0, 1]:

$$\begin{vmatrix} 0{,}7 & 0{,}6 \\ 0{,}4 & 0{,}1 \\ 0{,}5 & 0{,}5 \\ 0{,}2 & 0{,}9 \end{vmatrix}.$$

Ініціюється радіус та навчальний коефіцієнт: $R = 0$, $\alpha = 0{,}6$.

Задаються умови закінчення ітераційного процесу: процес закінчується, якщо виконуються вісім нерівностей:

$$\left| a_{ij}^{q} - a_{ij}^{q-1} \right| \le 0{,}0005, \quad i = \overline{1,4}, \ j = \overline{1,2}, \tag{5.1}$$

де a_{ij}^{q}, a_{ij}^{q-1} – елементи матриці ваг M_q після завершення поточної й попередньої ітерацій.

Крок 2. Починаються обчислення, що реалізують кроки 3 – 9 алгоритму.

Крок 3. Для першого вектора S^1 = (0 0 0 1) виконуються кроки 4 – 6.

Крок 4. Для нейронів A_1 і A_2 обчислюються відстані:

$$D(A_1) = (0,7 - 0)^2 + (0,4 - 0)^2 + (0,5 - 0)^2 + (0,2 - 1)^2 = 1,54,$$
$$D(A_2) = (0,6 - 0)^2 + (0,1 - 0)^2 + (0,5 - 0)^2 + (0,9 - 1)^2 = 0,63.$$

Крок 5. Визначається, що $D(A_1) > D(A_2)$, а отже, нейрон A_2 є переможцем.

Крок 6. Обчислюються нові ваги перемігшого нейрона A_2:

$$w_{i2}(new) = w_{i2}(old) + 0,6(S_i^1 - w_{i2}(old)) = 0,4w_{i2}(old) + 0,6S_i^1.$$

Після розрахунків виходить нова матриця ваг:

$$\begin{vmatrix} 0,7 & 0,24 \\ 0.4 & 0,04 \\ 0,5 & 0,20 \\ 0,2 & 0,96 \end{vmatrix}.$$

Крок 3. Для другого вектора S^2 = (0 0 1 1) виконуються кроки 4 – 6.

Крок 4. Для нейронів A_1 і A_2 обчислюються відстані:

$$D(A_1) = (0,7 - 0)^2 + (0,4 - 0)^2 + (0,5 - 1)^2 + (0,2 - 1)^2 = 1,54,$$
$$D(A_2) = (0,24 - 0)^2 + (0,04 - 0)^2 + (0,2 - 0\ )^2 + (0,96 - 1)^2 = 0,70.$$

Крок 5. Визначається, що нейрон A_2 є переможцем.

Крок 6. Адаптуються ваги зв'язків перемігшого нейрона й виходить матриця ваг з новим другим стовпцем:

$$\begin{vmatrix} 0,7 & 0,096 \\ 0,4 & 0,016 \\ 0,5 & 0,680 \\ 0,2 & 0,984 \end{vmatrix}.$$

Крок 3. Для третього вектора S^3 = (1 0 0 0) виконуються кроки 4 – 6.

Крок 4. Для нейронів A_1 і A_2 обчислюються відстані:

$$D(A_1) = (0,7 - 1)^2 + (0,4 - 0)^2 + (0,5 - 0)^2 + (0,2 - 0)^2 = 0,54,$$
$$D(A_2) = (0,096 - 1)^2 + (0,016 - 0)^2 + (0,680 - 0)^2 + (0,984 - 0)^2 = 1,888.$$

Крок 5. Визначається, що нейрон A_1 є переможцем.

Крок 6. Адаптуються ваги зв'язків перемігшого нейрона й виходить матриця ваг з новим першим стовпцем:

$$\begin{vmatrix} 0,880 & 0,096 \\ 0,160 & 0,016 \\ 0,200 & 0,680 \\ 0,080 & 0,984 \end{vmatrix}.$$

Крок 3. Для четвертого вектора S^4 = (1 1 0 0) виконуються кроки 4 – 6.

Крок 4. Обчислюються відстані:

$$D(A_1) = (0,880 - 1)^2 + (0,160 - 1)^2 + (0,200 - 0)^2 + (0,080 - 0)^2 = 0,766,$$
$$D(A_2) = (0,096 - 1)^2 + (0,016 - 1)^2 + (0,680 - 0)^2 + (0,984 - 0)^2 = 3,216.$$

Крок 5. Визначається, що переможцем є нейрон A_1.

Крок 6. Адаптуються ваги зв'язків перемігшого нейрона й виходить нова матриця ваг:

$$\begin{vmatrix} 0,952 & 0,096 \\ 0,664 & 0,016 \\ 0,080 & 0,680 \\ 0,032 & 0,984 \end{vmatrix}.$$

Крок 7. Модифікується навчальний коефіцієнт α:

$$\alpha(t = 1) = 0{,}5 \cdot \alpha(t = 0) = 0{,}5 \cdot 0{,}6 = 0{,}3.$$

Крок 8. Оскільки $R = 0$, то цей крок не виконується.

Крок 9. Перевіряються умови зупину, й оскільки вони не виконуються, то здійснюється перехід на крок 2 алгоритму (у противному випадку – зупин).

Аналогічно першій виконуються друга й наступна ітерації алгоритму. У результаті цих ітерацій маємо такі матриці ваг M_q $(q = \overline{1,9})$:

$$M_2 = \begin{vmatrix} 0{,}9760 & 0{,}0470 \\ 0{,}6250 & 0{,}0080 \\ 0{,}0330 & 0{,}6330 \\ 0{,}0150 & 0{,}9920 \end{vmatrix}, \qquad M_3 = \begin{vmatrix} 0{,}982 & 0{,}034 \\ 0{,}601 & 0{,}006 \\ 0{,}028 & 0{,}607 \\ 0{,}011 & 0{,}994 \end{vmatrix}, \ \ldots,$$

$$M_8 = \begin{vmatrix} 0{,}98642 & 0{,}02516 \\ 0{,}57916 & 0{,}00448 \\ 0{,}02080 & 0{,}58263 \\ 0{,}00882 & 0{,}99562 \end{vmatrix}, \qquad M_9 = \begin{vmatrix} 0{,}98648 & 0{,}02504 \\ 0{,}57879 & 0{,}00448 \\ 0{,}02070 & 0{,}58224 \\ 0{,}00878 & 0{,}99564 \end{vmatrix}.$$

Після виконання дев'ятої ітерації виконуються умови зупину (5.1) і алгоритм припиняє свою роботу.

Для векторів (0 0 0 1), (0 0 1 1), (1 0 0 0), (1 1 0 0), у силу того, що для кожного із двох класів тільки один з одиничних компонентів є в обох представниках класу, а інший – тільки в одному із двох, при цьому інші компоненти векторів нульові, ідеальна матриця M_u ваг має вигляд

$$M_u = \begin{vmatrix} 1 & 0 \\ 0{,}5 & 0 \\ 0 & 0{,}5 \\ 0 & 1 \end{vmatrix}.$$

Якщо порівнювати елементи w_{21}, w_{32} матриць M_u й M_9, то максимальна погрішність елементів, отриманих у результаті ітераційної процедури, становить 16,5 %. Така відносно велика погрішність

розрахунків пояснюється занадто швидкою зміною навчального коефіцієнта α. Збільшимо у виразі $\alpha(t+1) = k\alpha(t)$ коефіцієнт k з 0,5 до 0,75 і повторимо весь ітераційний процес при тих же вихідних даних та умовах закінчення ітерацій. У результаті одержимо, що число ітерацій до виконання умов зупину зросте з дев'яти до сімнадцяти, а погрішність визначення елементів w_{21}, w_{32} матриці ваг зменшиться до 8 %:

$$M_1' = \begin{vmatrix} 0{,}952 & 0{,}096 \\ 0{,}664 & 0{,}016 \\ 0{,}080 & 0{,}680 \\ 0{,}032 & 0{,}984 \end{vmatrix}, \quad M_2' = \begin{vmatrix} 0{,}98548 & 0{,}02904 \\ 0{,}65086 & 0{,}00484 \\ 0{,}02420 & 0{,}65570 \\ 0{,}00968 & 0{,}99516 \end{vmatrix}, \quad M_3' = \begin{vmatrix} 0{,}99363 & 0{,}01275 \\ 0{,}62316 & 0{,}00213 \\ 0{,}01061 & 0{,}62529 \\ 0{,}00425 & 0{,}99787 \end{vmatrix},$$

……,

$$M_{16}' = \begin{vmatrix} 0{,}99930 & 0{,}00141 \\ 0{,}54003 & 0{,}00024 \\ 0{,}01173 & 0{,}54022 \\ 0{,}00046 & 0{,}99976 \end{vmatrix}, \quad M_{17}' = \begin{vmatrix} 0{,}99931 & 0{,}00139 \\ 0{,}53957 & 0{,}00024 \\ 0{,}01159 & 0{,}53975 \\ 0{,}00046 & 0{,}99976 \end{vmatrix}.$$

Із зіставлення елементів матриць M_9 і M_{17}' виходить, що покращилися не тільки елементи w_{21}, w_{32}, але й всі інші компоненти матриці. При збільшенні числа ітерацій до 70 вдається одержати погрішність елементів w_{21}, w_{32} у межах одного відсотка.

2.3. Індивідуальні завдання

1. Виконати кластеризацію дев'яти двійкових векторів за допомогою нейронної мережі Кохонена на три класи. Двійкові вектори одержати в такий спосіб.

Записати своє прізвище, ім'я та по батькові. Вибрати із цих трьох слів слово з найбільшим числом букв (якщо два або всі слова мають однакове найбільше число букв, то вибирається одне з них). Закодувати голосні й приголосні букви цього слова відповідно одиницями й нулями. Отриманий двійковий код стане першим вектором (v_1) з дев'яти векторів, кластеризацію яких необхідно виконати за допомогою нейронної мережі Кохонена. Другий і третій вектори (v_2, v_3) одержати з вектора v_1 шляхом інверсії відповідно другого й третього розряду вектора v_1. Потім аналогічним чином закодувати нулями й одиницями букви наступних двох

слів. Якщо отримані двійкові вектори (v_4, v_5) коротші від векторів v_1, v_2, v_3, то їхню довжину слід збільшити до довжини цих векторів, додаючи ліворуч або праворуч необхідне число двійкових розрядів. Вектори v_6, v_7 й v_8, v_9 одержати відповідно з векторів v_4, v_5 шляхом інверсії відповідно другого й третього розрядів цих векторів.

2. Розробити архітектуру нейронної мережі для ваших вхідних векторів.

3. Обґрунтувати попередній вибір параметрів α і R, ваг зв'язків нейронної мережі для вашого випадку.

4. Дослідити працездатність мережі при різних значеннях α.

5. Зробити висновки.