Зміст

Вступ 3

1. Мережа Кохонена 4

2. Можливості Емулятора мережі Кохонена 12

3. Експерименти на емуляторі мережі Кохонена 15

Висновок 21

Список використаної літератури 22

# Вступ

Нейронні мережі Кохонена — клас нейронних мереж, основним елементом яких є шар Кохонена. Шар Кохонена складається з адаптивних лінійних суматорів («лінійних формальних нейронів»). Як правило, вихідні сигнали шару Кохонена обробляються за правилом «переможець забирає все»: найбільший сигнал перетворюється в одиничний, решта звертаються в нуль.

Дана програма реалізує розпізнавання десяти цифр 0-9 і показує основні дії, що виконуються для розпізнавання. Має розділи «Допомога» та «Теорія».

**1.Мережа Кохонена**

**Задача класифікації**

Задача класифікації полягає в розбитті об'єктів на класи, коли основою розбиття слугує вектор параметрів об'єкта. Об’єкти в межах одного класу вважаються еквівалентними з точки зору критерію розбиття. Самі класи часто бувають невідомі заздалегідь, а формуються динамічно (як, наприклад, в мережах Кохонена). Класи залежать від пропонованих об'єктів, і тому додавання нового об'єкта вимагає коректування системи класів.

Будемо характеризувати об'єкти, що підлягають класифікації, вектором параметрів хр є X1що має N компонент, компоненти позначаємо нижнім індексом: . Вектор параметрів - єдина характеристика об'єктів при їх класифікації. Введемо множину класів С1,..,СМ = {СМ} в просторі класів С:(C1UC2 ... UСМ) с С,

Простір класів може не збігатися з простором об'єктів X і мати іншу розмірність. У найпростішому випадку, коли простір класів і об'єктів збігаються, X = С, класи представляют собою області простору X, і об'єкт хр буде віднесений до одного з класів m0, якщо ср є Сm0. У загальному випадку X є С різні.

Визначимо ядра класів {сm} = с1, ..,ст в просторі класів С, як об'єкти, типові для свого класу. Наприклад, якщо для класифікації за національністю вибрати параметри {колір очей, зріст, колір волосся}, то ядро класу "українець" може мати параметри {блакитноокий, зріст 185, волосся русяве}, і до цього класу можна віднести об'єкт з параметрами {світло-блакитноокий, зріст 182, волосся темно-русяве}, тому з ядер "українець", "естонець", "киргиз" параметри об'єкта найбільш схожі на ядро "українець".

Очевидно, що близькість об'єкта до ядра необхідно оцінювати чисельно. Введемо міру близькості d(xp,cm) - скалярну функцію від об'єкта і ядра класу, яка тим менше, чим більше об'єкт схожий на ядро класу. Можуть знадобитися допоміжні міри близькості, визначені для двох об'єктів, d(xP1, хР2), і для двох ядер класів, d(cm1, сm2).

Найчастіше застосовується евклідова міра: d(x,y) = [Σ](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B3%D0%BC%D0%B0_(%D0%B1%D1%83%D0%BA%D0%B2%D0%B0))i(xi – уi)2, або «city block»: d(x,y) = [Σ](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B3%D0%BC%D0%B0_(%D0%B1%D1%83%D0%BA%D0%B2%D0%B0))i| xi – уi |.

Задавшись числом класів М, можна поставити завдання класифікації: знайти М ядер класів {сm} і розбити об'єкти {хр} на класи {Сm}, тобто побудувати функцію т(р) таким чином, щоб мінімізувати суму мір близькості:

min {D=[Σ](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B3%D0%BC%D0%B0_(%D0%B1%D1%83%D0%BA%D0%B2%D0%B0))p d(xp,cm(p))}.

У найпростішому випадку X = С, простір об'єктів X розбивається на області {Сm}, і якщо хРо є сm0, то m(р0) = m0, і об’єкт відносять до класу m0.

Кількість класів М може динамічно мінятися. При цьому часто виникають ситуації, коли об'єкти розподілені за класами нерівномірно. Необхідно контролювати рівномірність щільності ядер ст в просторі С і частку об'єктів, які відносяться до кожного класу т0. Конкретні вимоги повинні вибиратися в залежності від завдання. При необхідності можна коригувальних - вати щільність злиттям і поділом ядер. Критерії для цього можуть використовуватися різні, і, в основному, емпіричні. Наприклад, два класи можуть зливатися в один, якщо міра близькості їх ядер менше, ніж середня міра близькості ядер і всіх об'єктів в цих двох класах. Застосовуючи аналогічні правила, треба пам'ятати про "побічні ефекти". Наведений критерій злиття може працювати некоректно (рис. 1).

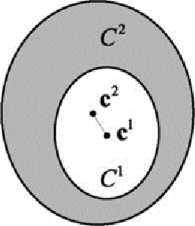


Рис. 1 Приклад некоректної роботи критерію злиття

Міра близькості ядер в цьому випадку набагато менше, аніж середня міра близькості між об'єктами і ядрами. У відповідності з правилом, класи будуть об'єднані, хоча в багатьох задачах таке об'єднання є необгрунтованим.

Корисно визначити діаметр класу - максимальна міра близькості між об'єктами даного класу.

**Алгоритм класифікації**

Велика частина алгоритмів - ітераційні. Серйозний недолік багатьох алгоритмів класифікації - теоретична необґрунтованість, відсутність доказів, що класифікація буде правильною, не кажучи вже про доведення оптимальності конкретного алгоритму.

Розглянемо загальний алгоритм з фіксованою кількістю ядер М. Кількість ядер вибираєся заздалегідь, виходячи з конкретної задачі.

Початкові значення ядер с1,...,ст можуть вибиратися випадковими, однаковими або за іншими евристичним правилам.

Кожна ітерація алгоритму складається з двох етапів:

1. При незмінних ядрах {сm} = const шукаємо таке розбиття т(р) об'єктів {xр} на класи, щоб мінімізувати сумарну міру близькості між об'єктами і ядрами їх класів:

min {D = [Σ](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B3%D0%BC%D0%B0_(%D0%B1%D1%83%D0%BA%D0%B2%D0%B0))pd(xp,cm(p))}.

Результат етапу створення функції m(р), розбиття об'єктів на класи.

1. При незмінному розбитті т(р) налаштовуємо ядра {сm} так, щоб в межах кожного класу т0 сумарна міра близкості ядра цього класу і об'єктів, що належать йому, була мінімальною:

min{D = [Σ](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B3%D0%BC%D0%B0_(%D0%B1%D1%83%D0%BA%D0%B2%D0%B0))p:m(p)=m0 d(xP,cm(p)},

Для всіх m0 = 1 ...М.

Результат цього етапу - новий набір ядер {сm}.

**Мережа Кохонена**

Якщо для класифікації застосовувати нейронні мережі, необхідно формалізувати задачу. Найбільш очевидний спосіб: виберемо в якості вхідних даних вектор параметрів єдиного об'єкту. Результатом роботи мережі буде код класу, до якого належить пред'явлений на вході об'єкт. У нейромережах прийнято кодування номером каналу. Тому мережа матиме М виходів, по числу класів, і чим більше значення приймає вихід номер т0, тим більше "впевненість" мережі в тому, що вхідний об'єкт належить до класу т0. Корисно застосувати функцію активації SOFTMAX, тоді сума виходів завжди буде дорівнювати одиниці. Кожен вихід можна буде трактувати як ймовірність того, що об'єкт належить даному класу. Всі виходи утворюють повну групу, тому сума виходів рівна одиниці, і об'єкт свідомо відноситься до одного з класів.

Виберемо евклідову міру близькості. У цьому випадку ядро класу, що мінімізує суму мір близькості для об'єктів цього класу, збігається з центром ваги об'єктів:

де N(m0) - кількість об'єктів хр в класі m0.

При розбитті на класи повинна бути мінімізована сумарна міра близькості для всієї множини {хр} вхідних об'єктів:



- розписано скалярний добуток. У цій сумі два доданки не залежать від способу розбиття і є постійні:



Тому задача пошуку мінімуму D еквівалентна пошуку максимуму виразу: minD --> max [Σ](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B3%D0%BC%D0%B0_(%D0%B1%D1%83%D0%BA%D0%B2%D0%B0))p [Σ](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B3%D0%BC%D0%B0_(%D0%B1%D1%83%D0%BA%D0%B2%D0%B0))i xip xim(p).

Запишемо варіант алгоритму класифікації для пошуку максимуму цієї функції:

1. Цикл: для кожного вектора хр{

2. Цикл: для кожного т {

3. Розрахувати [Σ](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B3%D0%BC%D0%B0_(%D0%B1%D1%83%D0%BA%D0%B2%D0%B0))i xip cim = Dm,p.

}// кінець циклу

1. Знаходимо m0, для якого m0: maxm{Dm,p}
2. Відносимо об'єкт до класу т0.

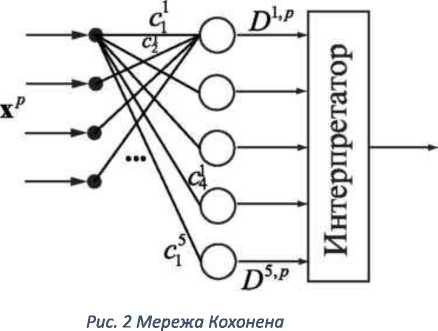
} / / Кінець циклу

Такий алгоритм легко реалізується у вигляді нейронної мережі. Для цього потрібно М суматорів, що знаходять усі Dm,p, і інтерпретатора, що знаходить суматор з максимальним виходом.

Сума [Σ](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B3%D0%BC%D0%B0_(%D0%B1%D1%83%D0%BA%D0%B2%D0%B0))i xip cip дуже нагадує зважену суму NETjt =[Σ](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%B3%D0%BC%D0%B0_(%D0%B1%D1%83%D0%BA%D0%B2%D0%B0))i wijl xijl,що розраховується формальним нейроном. Виберемо xi p в якості вхідних сигналів (що ми, втім, вже зробили) і компоненти ядер ci m в якості вагових коефіцієнтів w ijl. Тоді кожен формальний нейрон з числом входів, рівним числу компоненту вхідному векторі, буде давати на виході одну із сум Dm,p.

Щоб визначити клас, до якого належить об'єкт, потрібно вибрати серед всіх нейронів даного шару один з максимальним виходом - це здійснює інтерпретатор. Інтерпретатор - або програма, що вибирає нейрон з максимальним виходом, або шар нейронів з латеральним гальмуванням, що складається з нейронів із зворотними зв'язками. На звичайних ЕОМ програмний інтерпретатор ефективніше, тому що латеральне гальмування вимагає моделі гальмування процесу в часі, що вимагає багатьох ітерацій.

Розглянута мережа нейронів, що використовує евклідову міру близькості для класифікації об'єктів, називається мережею Кохонена (рис. 2).



Нейрони прошарку Кохонена генерують сигнали Dm,p. Інтерпретатор вибирає максимальний сигнал шару Кохонена та видає номер класу ш, що відповідає номеру входу, по якому інтерпретатором отриманий максимальний сигнал. Це відповідає номеру класу об'єкта, який був пред'явлений на вході, у вигляді вектора хр.

Ядра ст є ваговими коефіцієнтами нейронів. Кожен нейрон Кохонена запам'ятовує одне ядро класу, і відповідає за визначення об'єктів у своєму класі, тобто величина виходу нейрона тим більше, чим ближче об'єкт до даного ядра класу.

Загальна кількість класів співпадає з кількістю нейронів Кохонена. Міняючи кількість нейронів, можна динамічно змінювати кількість класів.

Нейрони Кохонена мають лінійну функцію активації. Якщо застосувати функцію SOFTMAX, то вихід шару Кохонена можна трактувати як ймовірність приналежності об'єкта до кожного з класів.

Але застосування SOFTMAX некоректно з точки зору принципу локальності, тому обчислення цієї функції активації вимагає знання всіх виходів мережі кожним з нейронів, а в реальної мережі це не виконується.

Вхідні вектори мережі найчастіше нормуються:



Можливі інші способи нормування.

**Навчання прошарку Кохонена**

Завдання навчання - навчити мережу активувати один і той же нейрон для схожих векторів хр на вході. Не важливо, який конкретно нейрон буде активований.

**Присвоєння початкових значень**

Зазвичай початкові значення в нейронних мережах вибираються малими випадковими числами. Для шару Кохонена такий вибір можливий, але має недоліки. Зрозуміло, якщо ядра класів нормовані, то й початкові значення потрібно нормувати.

Якщо ваги ініціалізуються випадковими значеннями з рівномірним розподілом, то виникає проблема. Коли ядра розподіляються рівномірно, то в областях простору X, де мало вхідних векторів, ядра будуть використовуватися рідко, тому мало буде схожих векторів. У тих областях, де вхідних векторів багато, щільність ядер виявиться недостатньою, і несхожі об'єкти будуть активувати один і той же нейрон, тому що більше схожого ядра не знайдеться. Для усунення проблеми можна виділяти ядра відповідно до щільністю вхідних векторів. Але розподіл вхідних векторів часто буває заздалегідь невідомим. У цьому випадку допомагає метод опуклою комбінації.

**Навчання мережі**

Якщо число вхідних векторів дорівнює числу ядер (тобто нейронів), то навчання не потрібно. Досить привласнити ядрам значення вхідних векторів, і кожен вектор буде активувати свій нейрон Кохонена. Але найчастіше кількість класів менше числа вхідних векторів. У цьому випадку ваги мережі настроюються ітеративним алгоритмом.

Алгоритм аналогічний вихідному алгоритму класифікації, але корекції ваг проводяться після пред'явлення кожного вхідного вектора, а не після пред'явлення всіх, як вимагає вихідний алгоритм. Збіжність при цьому зберігається.

1. Присвоюємо початкові значення ваговим коефіцієнтам.
2. Подаємо на вхід один з векторів хр.
3. Розраховуємо вихід шару Кохонена, Dm,p, і визначаємо номер нейрона-переможця m0, вихід якого максимальний, m0: mах0 Dm,p.
4. Коректуємо ваги тільки виграв нейрона т0: wm0 = wm0 + а(хр — wm0) - корекція записана у виді векторної вирази (вектор ваг wm0 нейрона т0 має стільки компонент, скільки їх у вхідного вектора хр). а - швидкість навчання, мала позитивна величина. Часто використовують розклад з навчанням, коли а = a(t) монотонно спадає. Вимоги до a(t) ті ж, що і у випадку багатошарового перцептрона.

Ваги коректуються так, що вектор ваг наближається до поточного вхідного вектора. Швидкість навчання управляє швидкістю наближення ядра класу (вектора ваг) до вхідного вектору хр.

Алгоритм виконується до тих пір, поки ваги не перестануть мінятися.

**Приклад навчання**

Розглянемо приклади навчання мережі Кохонена звичайним методом. На рис. З представлений приклад навчання. Точками позначені вектори хр навчальної множини, кругами - вектора вагових коефіцієнтів.

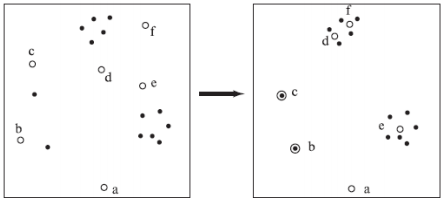


Рис. З Навчання мережі Кохонена

Вектор ваг нейрона а не навчається, тому що ні для одного з векторів навчальної множини цей нейрон не отримує максимального виходу. Крім того, в області з шести навчальних векторів (справа внизу) виявляється всього один вектор ваг нейрона е. Це не відповідає високій щільності навчальних векторів в цій області. Ці недоліки властиві звичайному методу навчання мережі Кохонена.

**2. Можливості Емулятора мережі Кохонена**

Дана програма реалізує розпізнавання десяти цифр 0-9 і показує основні дії, що виконуються для розпізнавання. Має розділи «Допомога» та «Теорія».

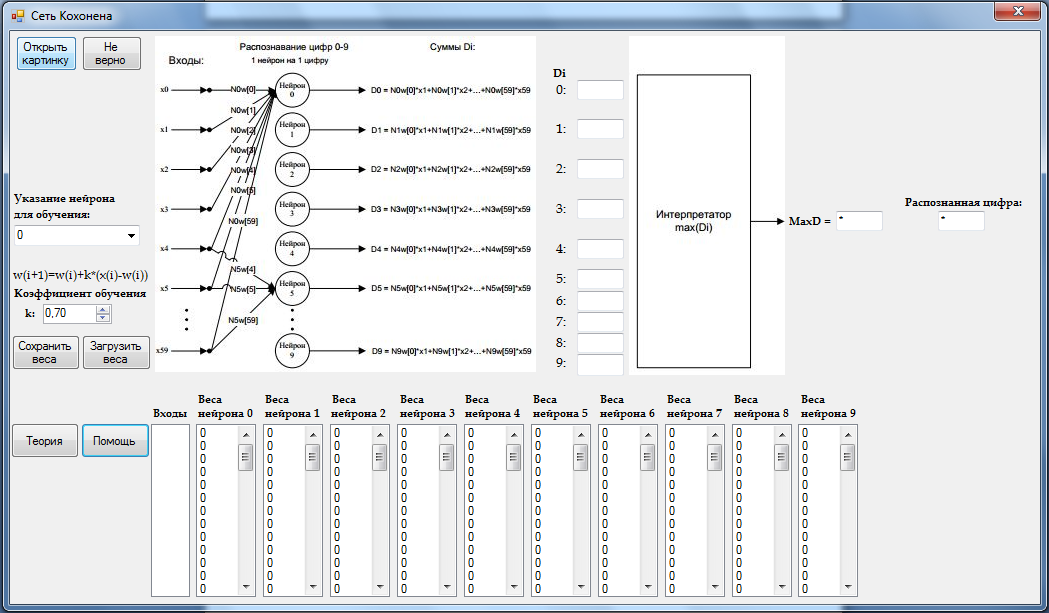
Вхідне зображення має розміри 6 на 10 пікселів. Основне вікно має вигляд:

Рис. 4. Основне вікно.

Містить у собі кнопки завантаження/збереження навченого стану мережі, кнопку відкриття зображення цифри. Показує стан Di для кожного нейрона, так що можна наглядно подивитись стан усіх нейронів після подання зображення. Для навчання передбачено зручну зміну коефіцієнта навчання у головному вікні програми.

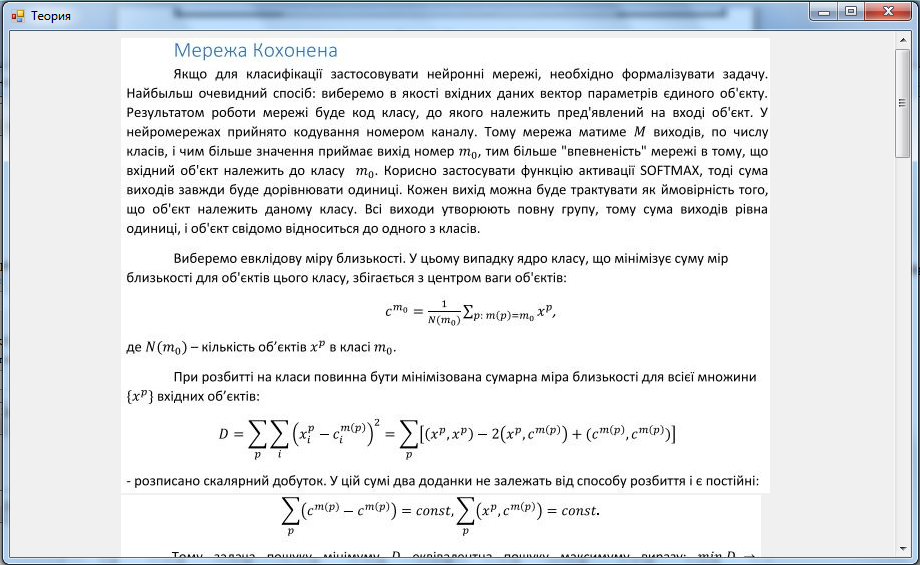
Додаткове вікно «Теорія», представляє собою нову форму, яка має полосу прокрутки і містить основну теорію:

Рис. 5. Вікно теорії

Додаткове вікно «Допомога», представляє собою нову форму, яка допомагає розібратися із програмою:

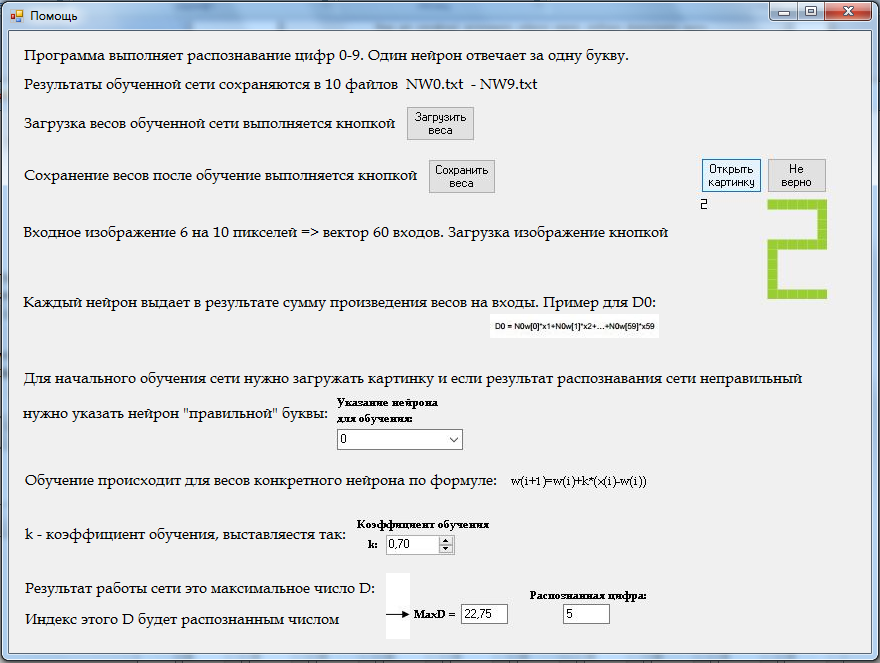


Рис. 6. Вікно Допомоги

**3.Експерименти на емуляторі мережі Кохонена**

Навчання мережі із коефіцієнтом навчання 0,7.

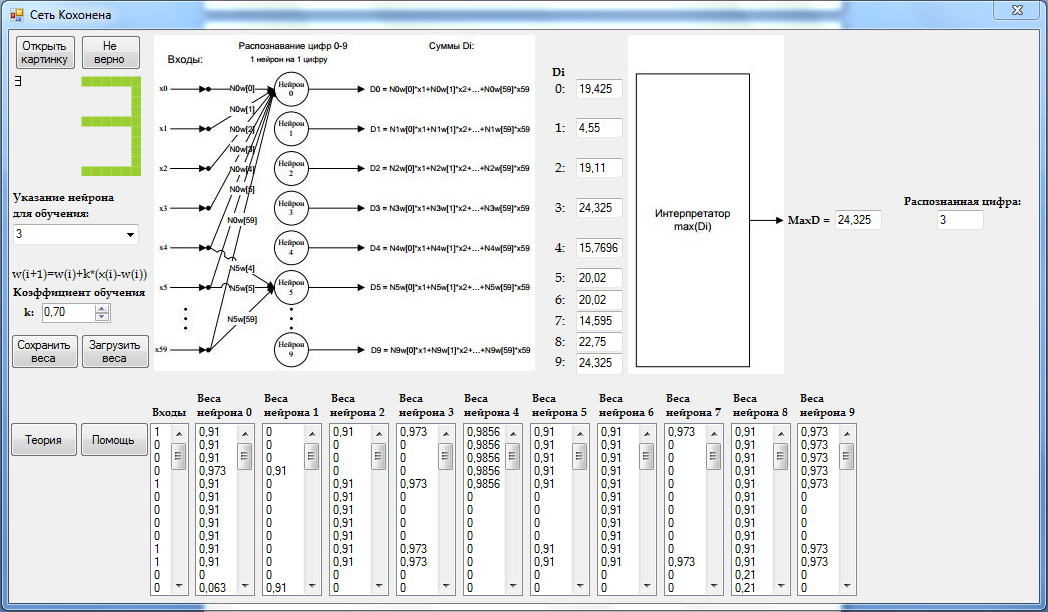


Рис. 7. Навчання з k=0.7

Кількість ітерацій навчання для «однорядних» цифр дорівнює 20. При цьому такі цифри як 9 та 3 можуть інколи співпадати при розпізнанні, так як вони дуже схожі і відрізняються усього на 3 пікселі.

Щоб уникнути такої ситуації, коли дві різних цифри розпізнаються як одна, зменшимо коефіцієнт до 0.5 та донавчимо мережу по цим декільком цифрам. Кількість ітерацій навчання зміниться до 27 і мережа буде розпізнавати всі цифри. На рисунку 8 можна побачити ваги після навчання.

Ваги:

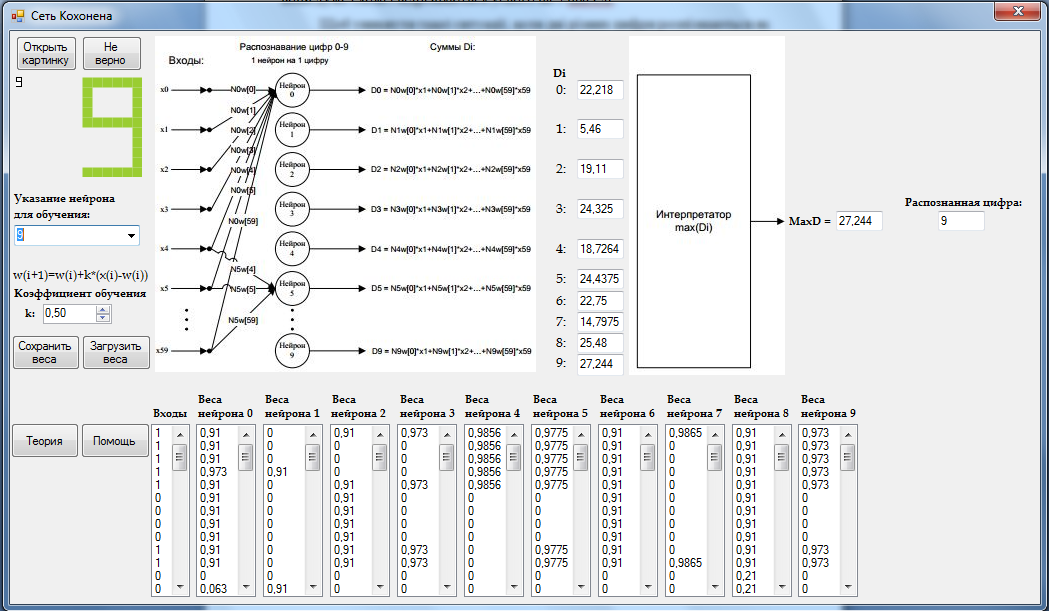


Рис. 8. Донавчання з k=0.5

Навчання мережі із коефіцієнтом навчання 0,5. Для «однорядних» цифр дорівнює 20. Мережа розпізнає усі цифри. Тобто швидше на 7 ітерацій. Ваги показані на рисунку 9.

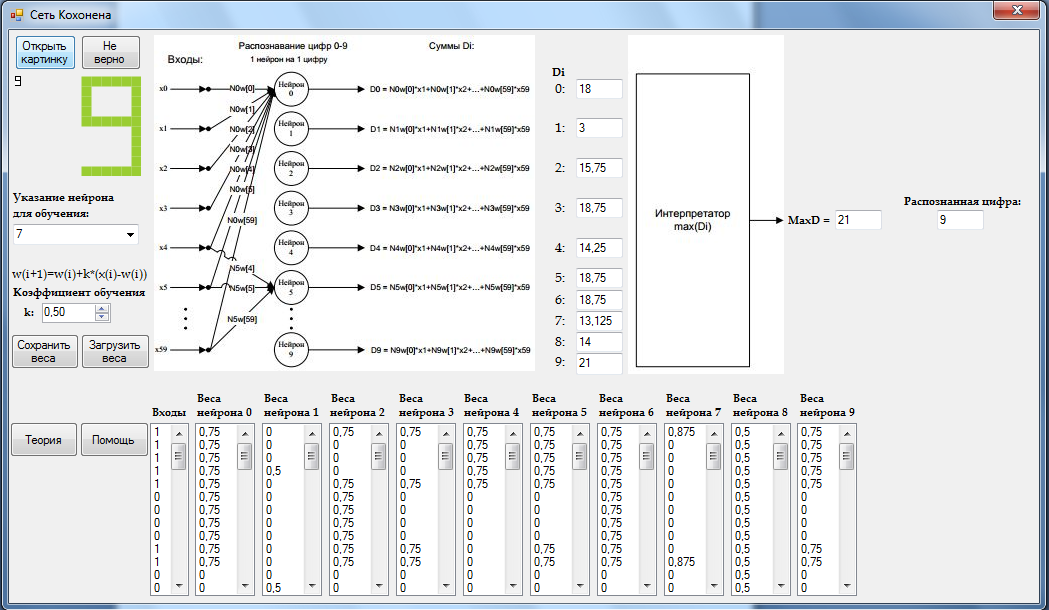


Рис. 9. Повне навчання при k=0.5

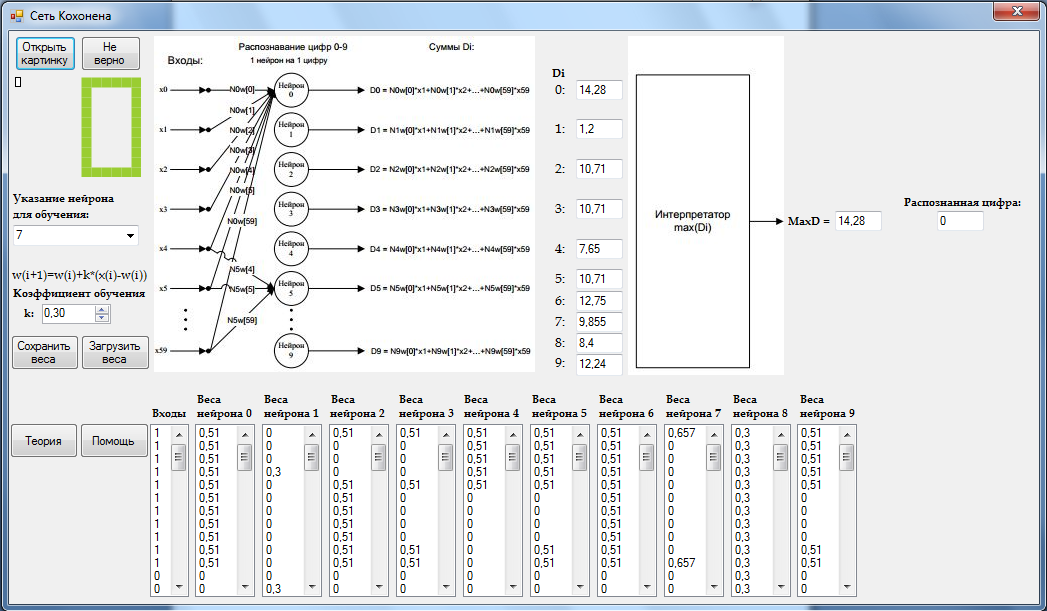
Навчання мережі із коефіцієнтом навчання 0,3. Для «однорядних» цифр дорівнює 19. Мережа розпізнає усі цифри. Тобто швидше на 8 ітерацій. Ваги показані на рисунку 10.

Рис. 10. Повне навчання при k=0.3

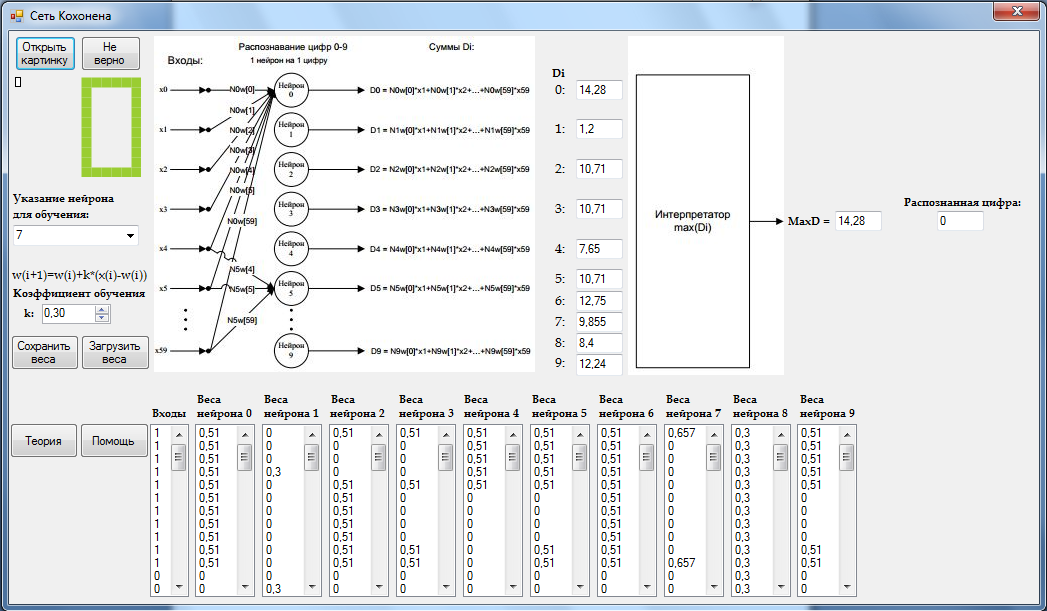
Навчання мережі із коефіцієнтом навчання 0,1. Для «однорядних» цифр дорівнює 27. Мережа розпізнає усі цифри. Ваги показані на рисунку 11.

Рис. 11. Повне навчання при k=0.1

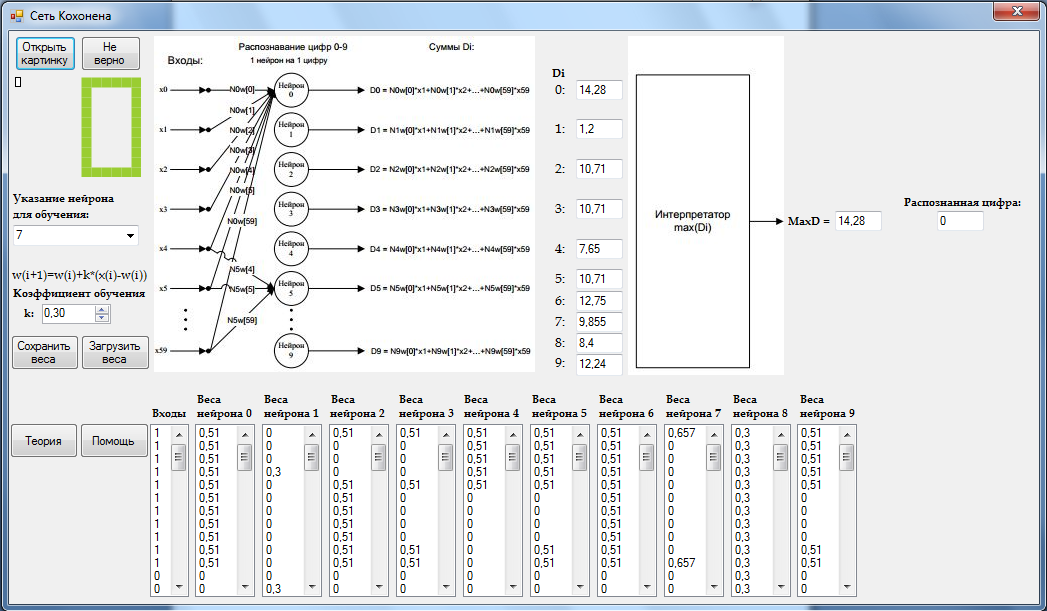
Навчання мережі із коефіцієнтом навчання 0,9. Для «однорядних» цифр дорівнює 16. Мережа розпізнає усі цифри. Ваги показані на рисунку 12.

Рис. 12. Повне навчання при k=0.9

# Висновок

Мережа розпізнає цифри, але можливе однакове розпізнання подібних цифр, таких як 3 та 9. Щоб мінімізувати такі випадки, необхідно правильно підібрати коефіцієнт навчання, а в кінці навчання поступово змінювати його. Суттєвою перепоною стає число 8, до якого подібно багато інших чисел, у цьому простому прикладі мережі.

# Список використаної літератури

1. Конспект лекцій
2. Мережа Кохонена

<https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C_%D0%9A%D0%BE%D1%85%D0%BE%D0%BD%D0%B5%D0%BD%D0%B0>

1. Штучні нейронні мережі <http://uatur.com/html/neural_nets/>