МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ
Ректор _____ С.В. Іванов
(підпис)
«___» _____20___ р.

ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ

ЛАБОРАТОРНИЙ ПРАКТИКУМ

до розділу «Застосування інтелектуальних систем»

для студентів спеціальностей 7.05020201, 8.05020201 «Автоматизоване управління технологічними процесами», 7.05020202, 8.05020202 «Комп'ютерно-інтегровані технологічні процеси і виробництва» денної та заочної форм навчання

Всі цитати, цифровий та фактичний матеріал, бібліографічні відомості перевірені. Написання одиниць відповідає стандартам

Підписи авторів _____
_____
_____

«____»_____20___р.
Реєстраційний номер
електронного
лабораторного практикуму
у НМВ 37.48 – 02.07. 2014

СХВАЛЕНО
на засіданні кафедри
автоматизації процесів управління
Протокол № 11
від 14 травня 2014р

КИЇВ НУХТ 2014

Інтелектуальні системи [Електронний ресурс]: Лабораторний практикум до розділу «Застосування інтелектуальних систем»для студентів спеціальностей 7.05020201, 8.05020201 «Автоматизоване управління технологічними процесами», 7.05020202, 8.05020202 «Комп'ютерно-інтегровані технологічні процеси і виробництва»
денної та заочної форм навчання /уклад./ В.Д. Кишенько, Я.В. Смітюх, Ю.О. Чорна . – К: НУХТ, 2014. – 48 с.

Рецензент: **О.П. Лобок**, канд. фіз.-мат. наук

Укладачі: **В.Д. Кишенько**, канд. техн. наук
Я.В. Смітюх, канд. техн. наук
Ю.О. Чорна

Відповідальний за випуск: **А.П. Ладанюк**, д-р. техн. наук, професор

Подано в авторській редакції.

Лабораторна робота № 1
Модель нейрона. Побудова функцій активації в середовищі MATLAB.

Мета: Вивчити структурні схеми моделі нейрона. Ознайомитися із засобами середовища MATLAB для побудови графіків функцій активації.

Теоретичні відомості

Сьогодні є безперечним значний науковий та практичний інтерес до обчислювальних структур нового типу — штучних нейронних мереж. Він спричинений низкою успішних застосувань цієї нової технології, яка дозволила розробити ефективні підходи до вирішення проблем, що вважалися складними для реалізації на традиційних комп'ютерах. На назву “нейронні мережі” зараз претендують усі обчислювальні структури, які в тій чи іншій мірі моделюють роботу мозку. Але таке моделювання, здебільшого, є дуже фрагментарним, і говорити про створення у найближчому майбутньому штучного мозку або навіть деякої його моделі, яка дублювала б роботу мозку найпримітивніших живих створінь, ще зарано. Такий висновок випливає з надзвичайної складності цього загадкового витвору природи.

Успішний розвиток теорії нейронних мереж за останнє десятиліття дозволив реалізувати ряд таких глобальних властивостей. Найвідомішими з них є навчання, узагальнення та абстрагування.

Протягом останніх років значно активізувались дослідження, спрямовані на пошук більш ефективних моделей нейрона. В роботах [83,101] запропоновані моделі, принциповою відмінністю яких є залучення до процесу обробки інформації нових характеристик, що задані внутрішніми змінними. Наступним кроком у згаданому напрямку стала робота[119], в якій крім внутрішніх структур даних використовувались і внутрішні процедури, які реалізують характеристичні функції, призначені для модифікації спільних структур даних прихованого шару. Сучасний стан даної області свідчить про те, що вона вже цілком довела свою перспективність та інтенсивно розвивається.

За допомогою штучних нейронних мереж успішно реалізовано 18 ряд комерційних проектів, пов’язаних із розпізнаванням зображень та звукової інформації. Але для досягнення більш значних результатів потрібна розробка нових технологій та методів обробки інформації, які базувалися б на принципово нових теоретичних основах.

Нейрон, як і всі інші клітини, складається з ядра та цитоплазми (рис.1.1). Тіло клітини (сома) від навколишнього середовища відокремлене тонкою мембраною завтовшки 75 ангстрем, що складається з ліпідів (жироподібних речовин) та характеризується низькою провідністю.

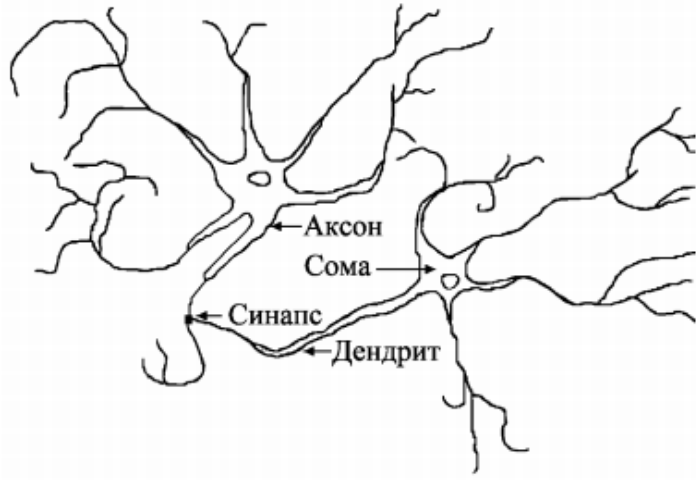


Рис. 1.1. Біологічний нейрон

Нейрони характеризуються також наявністю у них спеціальних відростків. По одних відростках нейрони одержують інформацію, а по других передають сигнали іншим нейронам. По деревовидних відростках (дендритах) нейрон отримує інформацію через спеціальні контакти (синапси). Передача інформації від одного нейрона до іншого відбувається шляхом розповсюдження нервового імпульсу вздовж нервового волокна– аксона. Кожен нейрон може мати велику кількість дендритів і тільки один аксон. Волокно аксона розгалужується і створює з'єднання з дендритами інших нейронів через відповідні синаптичні контакти.

Історично першою публікацією, що заклала підвалини для створення штучних нейронів та нейронних мереж, вважають роботу Уоррена С. Мак-Каллока та Вальтера Піттса. У цій роботі було започатковано теорію, в основі якої лежав той факт, що всі аспекти нервової діяльності можна моделювати за допомогою мережі елементів, які мають два стійкі стани. Модель нейрона представлена на рис. 1.2.

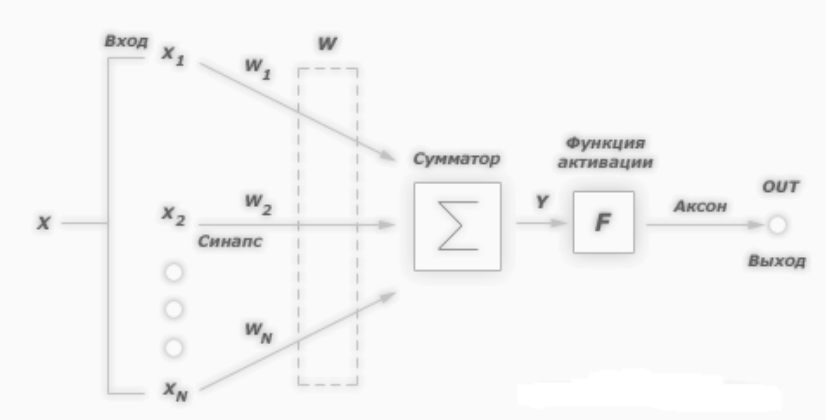


Рис. 1.2. Модель нейрона

У цій моделі нейрона можна виділити три основні елементи :

- синапси, кожен з яких характеризується своєю вагою або силою. Здійснюють зв'язок між нейронами, множать вхідний сигнал x_i на ваговий коефіцієнт w_i синапсу, що характеризує силу синаптичного зв'язку;
- суматор, аналог тіла клітини нейрона. Виконує додавання зовнішніх вхідних сигналів або сигналів, що надходять по синаптичних зв'язках від інших нейронів. Визначає рівень збудження нейрона;
- функція активації, визначає остаточний вихідний рівень нейрона, з яким сигнал збудження (гальмування) надходить на синапси наступних нейронів.

Модель нейрон імітує в першому наближенні властивості біологічного нейрона. На вхід штучного нейрона надходить деяка безліч сигналів, кожен з яких є виходом іншого нейрона. Кожен вхід множиться на відповідну вагу, пропорційний синаптичної силі, і все сумується, визначаючи рівень активації нейрона.

Хоча мережеві парадигми досить різноманітні, в основі майже всіх їх лежить ця модель нейрона. Тут безліч вхідних сигналів, позначених x_1, x_2, … x_N надходить на штучний нейрон. Ці вхідні сигнали, в сукупності позначаються вектором X, відповідають сигналам, що приходять в синапси біологічного нейрона. Кожен сигнал множиться на відповідну вагу w_1, w_2, … w_N і надходить на сумуючий блок, позначений Sigma. Кожна вага відповідає «силі» однієї біологічної синаптичного зв'язку. Безліч ваг в сукупності позначається вектором W. Сумуючий блок, що відповідає тілу біологічного елемента, складає зважені входи алгебраїчно, створюючи вихід Y. Далі Y надходить на вхід функції активації, визначаючи остаточний сигнал

збудження або гальмування нейрона на виході. Цей сигнал надходить на синапси наступних нейронів і т.д.

Математична модель нейрона:

$$S = \sum_{i=1}^{N} w_i \cdot x_i + b,$$
$$Y = f(S) \qquad (1.1)$$

де S - результат підсумовування (sum); w_i - вага (weight) синапсу, $i = 1...n$; x_i - компонент вхідного вектора (вхідний сигнал), $i = 1...n$; b - значення зсуву (bias); n - число входів нейрона; Y - вихідний сигнал нейрона; Y - нелінійне перетворення (функція активації)

Завдання на лабораторну роботу

1. В середовищі MATLAB побудувати одиничну функцію активації, задаючи довільні параметри.
2. В середовищі MATLAB побудувати лінійну функцію активації, задаючи довільні параметри.
3. В середовищі MATLAB побудувати логістичну функцію активації, задаючи довільні параметри.

Методика виконання

Функції активації (передавальні функції) нейрона можуть мати самий різний вигляд. Функція активації f, як правило, належить до класу сигмоїдальних функцій, які мають дві горизонтальні асимптоти і одну точку перегину, з аргументом функції n (входом) і значенням функції (виходом) a.

Розглянемо три найбільш поширені форми функції активації.

Одинична функція активації з жорстким обмеженням hardlim

Ця функція описується співвідношенням a = hardlim (n) = 1 (n) і показана на рис. 1.3

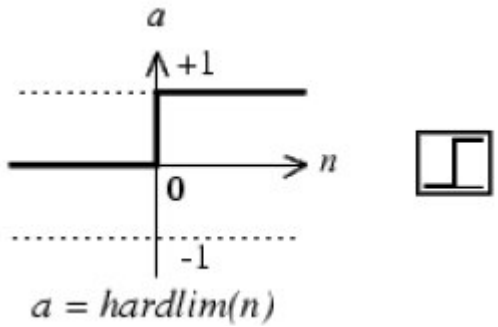


Рис. 1.3. Одинична функція активації

Вона дорівнює 0, якщо n <0, і дорівнює 1, якщо n ≥ 0.

Щоб побудувати графік цієї функції в діапазоні значень входу від -5 до +5, необхідно ввести наступні оператори мови MATLAB в командному вікні:

```
n = -5:0.1:5;
plot(n,hardlim(n),'b+:');
```
(1.2)

Лінійна функція активації purelin

Ця функція описується співвідношенням a = purelin (n) = n і показана на рис. 1.4.

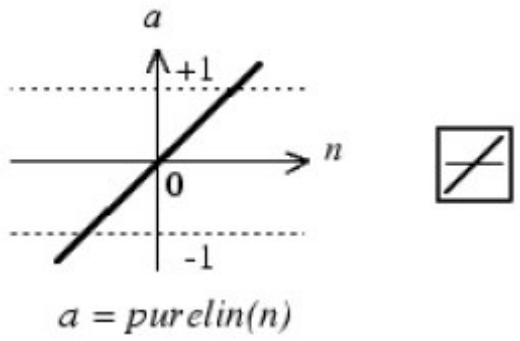


Рис. 1.4. Лінійна функція активації

Щоб побудувати графік цієї функції в діапазоні значень входу від -5 до +5, необхідно ввести наступні оператори мови MATLAB в командному вікні:

```
n=-5:0.1:5;
plot(n,purelin(n),'b+:');
```
(1.3)

Логістична функція активації logsig

Ця функція описується співвідношенням a = logsig (n) = 1 / (1 + exp (-n)) і показана на рис. 1.5.

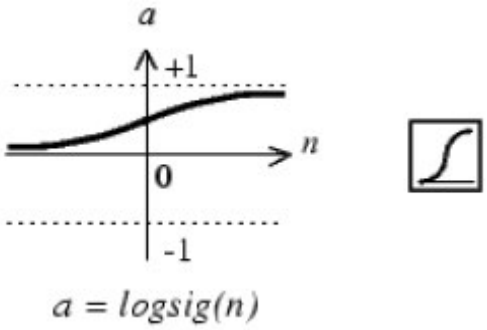


Рис. 1.5. Логістична функція активації

Дана функція належить до класу сигмоїдальних функцій, і її аргумент може приймати будь-яке значення в діапазоні від - ∞ до + ∞, а вихід змінюється в діапазоні від 0 до 1. Завдяки властивості диференціювання (немає точок розриву) ця функція часто використовується в мережах з навчанням на основі методу зворотного поширення помилки.

Щоб побудувати графік цієї функції в діапазоні значень входу від -5 до +5, необхідно ввести наступні оператори мови MATLAB в командному вікні:

```
n=-5:0.1:5;
plot(n,logsig(n),'b+:');
```
(1.4)

Аналіз одержаних результатів. Висновки і рекомендації.

В протоколі вказати тему та мету лабораторної роботи, навести коротко теоретичні відомості. Кожне завдання лабораторної роботи оформляєте заголовком та вказуються параметри функцій належності, які були обрані довільно.

У висновку вказати, як впливає зміна параметрів функції належності на її вигляд.

Контрольні питання

1. Що таке нейрон?
2. Охарактеризуйте математичну модель нейрона?
3. Що таке функція активації та назвіть її види?
4. Яка функція в середовищі MATLAB застосовуєть для побудови одиничної функцій активації?
5. Яка функція в середовищі MATLAB застосовуєть для побудови лінійної функцій активації?
6. Яка функція в середовищі MATLAB застосовуєть для побудови логістичної функцій активації?

Лабораторна робота № 2
Нейронна мережа Кохонена.

Мета: Дослідити архітектуру нейронної мережі Кохонена. Дослідити створення моделі нейронної мережі в середовищі MATLAB.

Теоретичні відомості

Мережі, звані картами Кохонена, - це один з різновидів нейронних мереж, однак вони принципово відрізняються від розглянутих вище, оскільки використовують неконтрольоване навчання. Нагадаємо, що при такому

навчанні навчальна множина складається лише із значень вхідних змінних, в процесі навчання немає порівнювання виходів нейронів з еталонними значеннями. Можна сказати, що така мережа вчиться розуміти структуру даних.

Ідея мережі Кохонена належить фінському вченому Тойво Кохоненом (1982 рік). Основний принцип роботи мереж - введення в правило навчання нейрона інформації щодо його розташування.

В основі ідеї мережі Кохонена лежить аналогія з властивостями людського мозку. Кора головного мозку людини являє собою плоский лист і згорнута складками. Таким чином, можна сказати, що вона володіє певними топологічними властивостями (ділянки, відповідальні за близькі частини тіла, примикають один до одного і все зображення людського тіла відображається на цю двовимірну поверхню).

Завдання, які вирішуються за допомогою карт Кохонена

Самоорганізуючі карти можуть використовуватися для вирішення таких завдань, як моделювання, прогнозування, пошук закономірностей у великих масивах даних, виявлення наборів незалежних ознак і стиснення інформації.

Найбільш поширене застосування мереж Кохонена – рішення задачі класифікації без вчителя, тобто кластеризації.

Нагадаємо, що при такій постановці завдання нам дано набір об'єктів, кожному з яких зіставлена рядок таблиці (вектор значень ознак). Потрібно розбити вихідну множину на класи, тобто для кожного об'єкта знайти клас, до якого він належить.

У результаті отримання нової інформації про класи можлива корекція існуючих правил класифікації об'єктів.

Ось два з поширених застосувань карт Кохонена: розвідувальний аналіз даних і виявлення нових явищ.

Розвідувальний аналіз даних. Мережа Кохонена здатна розпізнавати кластери в даних, а також встановлювати близькість класів. Таким чином, користувач може поліпшити своє розуміння структури даних, щоб потім уточнити нейромережеву модель. Якщо в даних розпізнані класи, то їх можна позначити, після чого мережа зможе вирішувати завдання класифікації. Мережі Кохонена можна використовувати і в тих завданнях класифікації, де класи вже задані, - тоді перевага буде в тому, що мережа зможе виявити подібність між різними класами.

Виявлення нових явищ. Мережа Кохонена розпізнає кластери в навчальних даних і відносить всі дані до тих чи інших кластера. Якщо після цього мережа зустрінеться з набором даних, несхожим ні на один з відомих

зразків, то вона не зможе класифікувати такий набір і тим самим виявить його новизну.

Мережа Кохонена, на відміну від багатошарової нейронної мережі, дуже проста; вона являє собою два шари: вхідний і вихідний. Її також називають самоорганізуючою картою. Елементи карти розташовуються в деякому просторі, як правило, двовимірному. Мережа Кохонена зображена на рис. 2.1.

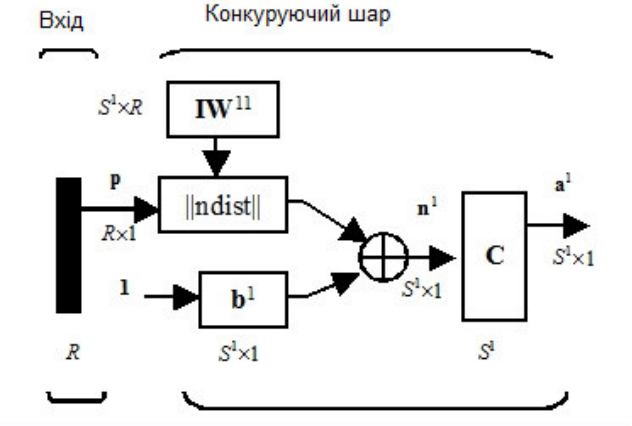


Рис. 2.1. Нейронна мережа Кохонена

Неважко переконатися, що це шар конкуруючого типу, оскільки в ньому застосована конкуруюча функція активації. Крім того, архітектура цього шару дуже нагадує архітектуру прихованого шару радіальної базисної мережі. Тут використано блок ndist для обчислення негативного евклідова відстані між вектором входу p і рядками матриці ваг IW^{11}. Вхід функції активації n^1- це результат підсумовування обчисленого відстані з вектором зміщення b. Якщо всі зсуви нульові, максимальне значення n^1 не може перевищувати 0. Нульове значення n^1 можливо тільки, коли вектор входу p виявляється рівним вектору ваги одного з нейронів. Якщо зміщення відмінні від 0, то можливі й позитивні значення для елементів вектора n^1.

Конкуруюча функція активації аналізує значення елементів вектора n^1 і формує виходи нейронів, рівні 0 для всіх нейронів, крім одного нейрона - переможця, що має на вході максимальне значення. Таким чином, вектор виходу шару a^1 має єдиний елемент, рівний 1, який відповідає нейрона-переможця, а інші рівні 0. Така активаційна характеристика може бути описана таким чином:

$$a_i^1 = \begin{cases} 1, & i = i^*, \quad i^* = \arg(\max \quad n_i^1) \\ 0, & i \neq i^* \end{cases} \qquad (2.1)$$

Мережа Кохонена навчається методом послідовних наближень. У процесі навчання таких мереж на входи подаються дані, але мережа при цьому підлаштовується не під еталонне значення виходу, а під закономірності у вхідних даних. Починається навчання з вибраного випадковим чином вихідного розташування центрів.

У процесі послідовної подачі на вхід мережі навчальних прикладів визначається найбільш схожий нейрон (той, у якого скалярна похідна вагів і поданого на вхід вектора мінімально). Цей нейрон оголошується переможцем і є центром при підстроюванні ваг у сусідніх нейронів. Таке правило навчання передбачає " змагальне " навчання з урахуванням відстані нейронів від "нейрона - переможця".

Навчання при цьому полягає не в мінімізації помилки, а в підстроюванні ваг (внутрішніх параметрів нейронної мережі) для найбільшого збігу з вхідними даними.

Основний ітераційний алгоритм Кохонена послідовно проходить ряд епох, на кожній з яких обробляється один приклад з навчальної вибірки. Вхідні сигнали послідовно пред'являються мережі, при цьому бажані вихідні сигнали не визначаються. Після пред'явлення достатнього числа вхідних векторів синаптичні ваги мережі стають здатні визначити кластери. Ваги організуються так, що топологічно близькі вузли чутливі до схожих вхідних сигналів.

В результаті роботи алгоритму центр кластера встановлюється в певній позиції, задовільним чином кластеризуються приклади, для яких даний нейрон є "переможцем ". В результаті навчання мережі необхідно визначити міру сусідства нейронів, тобто околиця нейрона - переможця.

Околиця являє собою кілька нейронів, які оточують нейрон – переможець.

Спочатку до околиці належить велике число нейронів, далі її розмір поступово зменшується. Мережа формує топологічну структуру, в якій схожі приклади утворюють групи прикладів, близько знаходяться на топологічної карті.

Отриману карту можна використовувати як засіб візуалізації при аналізі даних. В результаті навчання карта Кохонена класифікує вхідні приклади на кластери (групи схожих прикладів) і візуально відображає багатовимірні вхідні дані на площині нейронів.

Унікальність методу самоорганізованих карт полягає в перетворенні n - мірного простору в двомірне . Застосування двомірних сіток пов'язано з тим, що існує проблема відображення просторових структур більшої розмірності.

Маючи таке подання даних, можна візуально визначити наявність або відсутність взаємозв'язку у вхідних даних.

Нейрони карти Кохонена розташовують у вигляді двомірної матриці, розфарбовують цю матрицю залежно від аналізованих параметрів нейронів.

Таким чином, карти Кохонена (як і географічні карти) можна відображати:

- у двомірному вигляді, тоді карта розфарбовується відповідно до рівня виходу нейрона;
- в тривимірному вигляді.

В результаті роботи алгоритму отримуємо такі карти:

- карта входів нейронів;
- карта виходів нейронів;
- спеціальні карти.

Координати кожної карти визначають положення одного нейрона. Так, координати [15:30] визначають нейрон, який знаходиться на перетині 15-го шпальти з 30-м поруч в матриці нейронів. Розглянемо, що ж являють собою ці карти.

Карта входів нейронів.

Ваги нейронів підлаштовуються під значення вхідних змінних і відображають їх внутрішню структуру. Для кожного входу малюється своя карта, розфарбована у відповідності зі значенням конкретного ваги нейрона. При аналізі даних використовують кілька карт входів. На одній з карт виділяють область певного кольору - це означає, що відповідні вхідні приклади мають приблизно однакове значення відповідного входу. Кольорове розподіл нейронів з цієї області аналізується на інших картах для визначення схожих або відмінних характеристик. Приклад розглянутих карт входів буде наведено нижче.

Карта виходів нейронів.

На карту виходів нейронів проектується взаємне розташування досліджуваних вхідних даних. Нейрони з однаковими значеннями виходів утворюють кластери - замкнуті області на карті, які включають нейрони з однаковими значеннями виходів.

Завдання на лабораторну роботу

Завдання на лабораторну роботу полягає у побудові самоорганізованої мережі на основі кластерів, що задаються в середовищі MATLAB.

Методика виконання

Цей спосіб навчання мережі вам знадобитися, якщо ви не знаєте мети (значення T). Як бачите з цього графіка, система сама автоматично визначила і рознесла центри кластерів. Також є параметри для тонкої настройки цієї мережі, їх можна подивитися при навчанні мережі, такі як кількість епох, градієнт, похибка і т.д., до них потрібно звертатися як і'мя_мережі.команда, наприклад net.time. Мережа - самоорганізована мережа Кохонена

Кількість нейронів – 4

База знань – z

Розмірність бази знань - 2x200

Задамо одне з множин значень кластера № 1.

Центром кластера буде є точка (3, 0) з невеликим розкидом від центру rand (1,30) і кількістю відліків 50. Тобто x1 і y1 буде масивом розмірність 1x30

```
y1=0+rand(1,30)

x1=3+rand(1,30)
```

Подивимося отримані значення

```
plot(x1,y1,'ob')
```

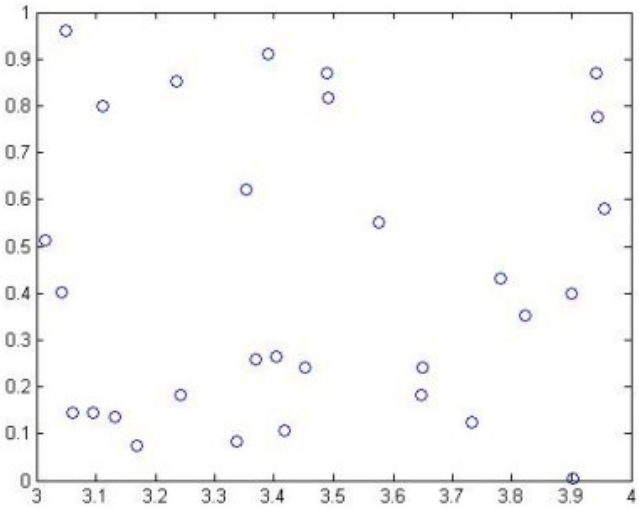


Рис. 2.2. Вікно сформованого першого кластеру

Також задамо безліч значень для другого кластера з центром (-3; 0)

```
y2=0+rand(1,30)

x2=-3+rand(1,30)
```

Також можна переглянути отримані значення

```
plot(x2,y2,'or')
```

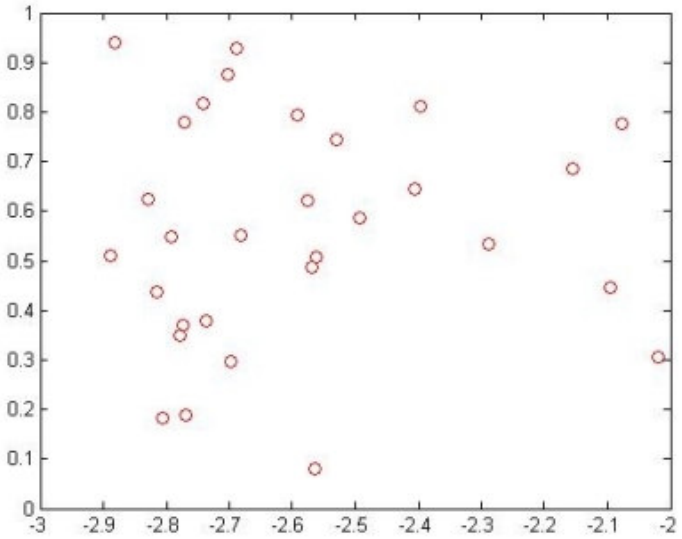


Рис.2.3 Вікно сформованого другого кластера

Далі задамо третій і четвертий кластери з центрами (0; 3) і (0; -3)

```
y3=3+rand(1,30)

x3=0+rand(1,30)

x4=0+rand(1,30)

y4=-2+rand(1,30)
```

Для повної наочності подивимося все кластери на одному графіку. Подивимося що у нас вийшло на figure(1)

```
figure (1)

hold on

plot(x3,y3,'og')

plot(x4,y4,'oy')

plot(x1,y1,'ob')

plot(x2,y2,'or')

grid on

hold off
```

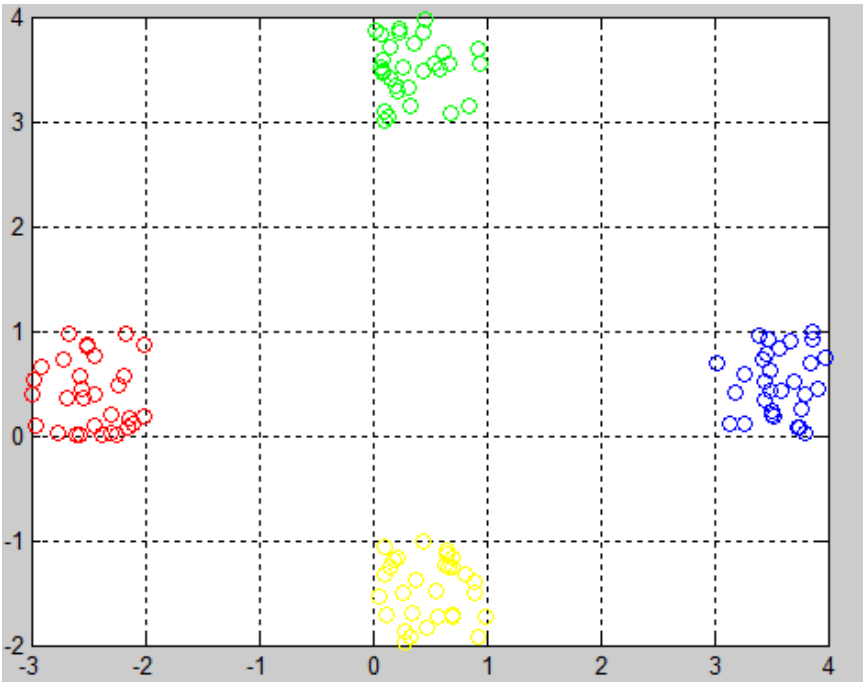


Рис. 2.4. Вікно чотирьох кластерів

Задамо мети наших чотирьох кластерів

```
T3(1:30)=10;

T4(1:30)=20;

T1(1:30)=30;

T2(1:30)=40;
```

Також як у прикладі про feed forward мережі, з'єднаємо всі цілі кластерів

```
T(1:30)=T1;

T(31:60)=T2;

T(61:90)=T3;

T(91:120)=T4;
```

Перед тек подавати це на вхід нейромережі,
Нам необхідно з'єднати всю базу знань для нейронної
Системи в одну матрицю, а саме сполучимо x і y і все
Це присвоїмо в змінну z

```
x(1:30)=x1;

x(31:60)=x2;

x(61:90)=x3;

x(91:120)=x4;

y(1:30)=y1;

y(31:60)=y2;

y(61:90)=y3;

y(91:120)=y4;

z(1,1:120)=x

z(2,1:120)=y
```

Задамо сітку карти Кохонена гексогональну 2х2,
Так як цього нам цілком вистачить

```
net = newsom(z,[2 2]);
```

Проведемо цикл навчання

```
net = train(net, z)
```

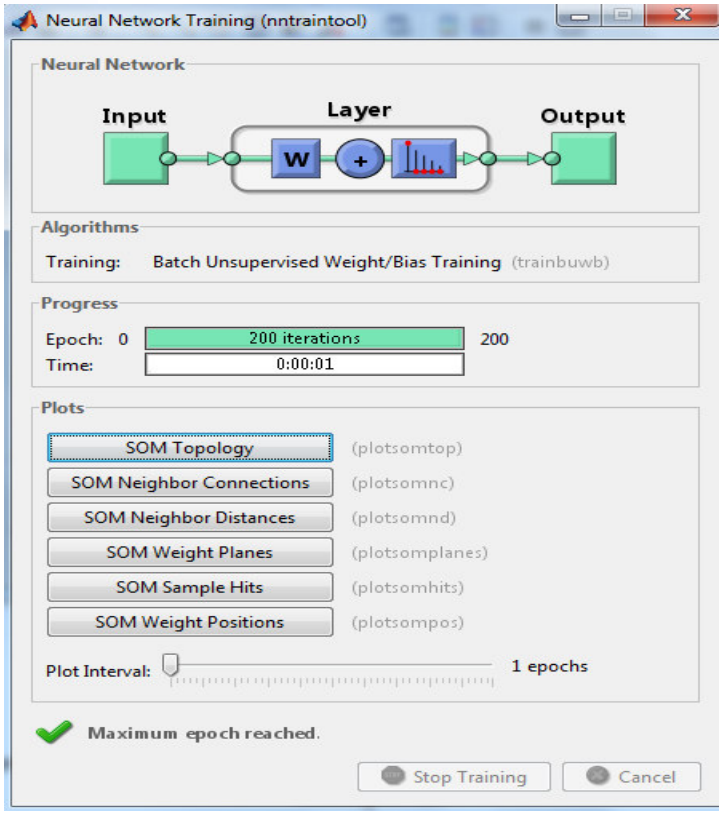


Рис. 2.5. Вікно навчання мережі

Ось так виглядає карта Кохонена, яку ми задали

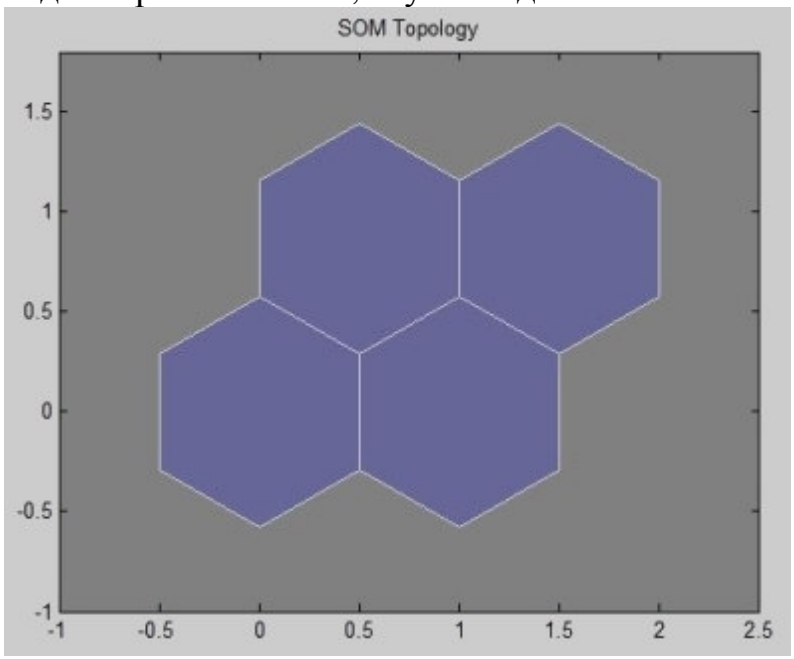


Рис. 2.6. Вікно карти Кохонена

Ось так розподілилися по кластерах усі спектри, які ми використовували при навчанні

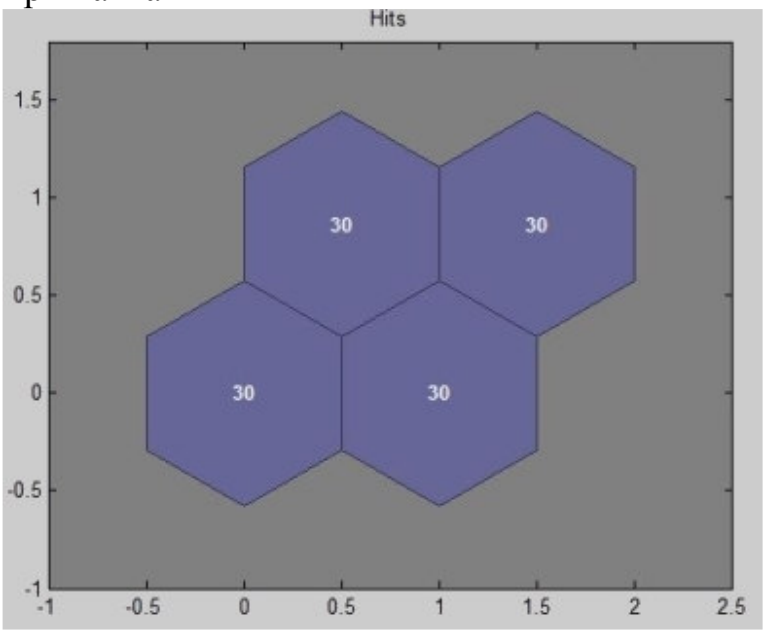


Рис. 2.7. Вікно розподілу кластерів по спектрам

Ось так розподілилися центри кластерів нейронної мережі Кохонена

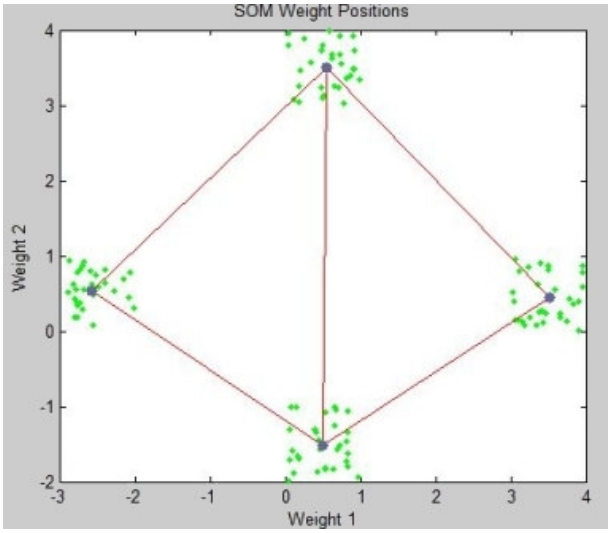


Рис. 2.8. Вікно розподілу центрів кластерів мережі Кохонена

Тепер перевіримо вийшла мережу на тих

Спектрах за допомогою, яких ми навчали мережу

```
a = sim(net,z)
```

Аналогічно можете перевірити на інших спектрах

Аналіз одержаних результатів. Висновки і рекомендації.

В протоколі вказати тему та мету лабораторної роботи, навести коротко теоретичні відомості. Записати програму побудови самоорганізованої мережі на основі кластерів, що задаються в середовищі MATLAB. Зробити скріншоти відповідних вікон.

У висновку вказати з яких елементів складається мережа Кохонена та для чого створюються кластери.

Контрольні питання

1. Що таке карти Кохонена?
2. З яких елементів складається нейронна мережа Кохонена?
3. Як відбувається навчання мережі Кохонена?
4. Як будується карта входів нейронів?
5. Як будується карта виходів нейронів?
6. Як задаються кластери для мережі Кохонена в середовищі Matlab?
7. Як задається сітка для мережі Кохонена в середовищі Matlab

Лабораторна робота № 3
Лінійна мережа. Навчання лінійної мережі з використанням алгоритму Хоффа.

Мета: Ознайомитись з будовою лінійної мережі. Вивчити алгоритми налаштування параметрів лінійної нейронної мережі з використанням процедури **train** в середовищі MATLAB.

Теоретичні відомості

Лінійні нейронні мережі за своєю структурою аналогічні персептрону і відрізняються лише функцією активації, яка є лінійною. Вихід лінійної мережі може приймати будь-яке значення, в той час як вихід персептрона обмежений значеннями 0 або 1. Лінійні мережі, як і персептрони, здатні вирішувати тільки лінійно віддільні задачі класифікації, проте в них використовується інше правило навчання, засноване на методі найменших квадратів, яке є більш потужним, ніж правило навчання персептрона. Налаштування параметрів виконується таким чином, щоб забезпечити мінімум помилки. Поверхня помилки як функція входів має єдиний мінімум, і визначення цього мінімуму не викликає труднощів. На відміну від персептрона настройка лінійної мережі може бути виконана за допомогою як процедури адаптації, так і процедури навчання; в останньому випадку використовується правило навчання WH (Widrow - Hoff). Крім того, в розділі розглядаються адаптуються лінійні нейронні мережі ADALINE (ADAptive Linear Neuron networks), які дозволяють коригувати ваги і зміщення при надходженні на вхід кожного нового елемента навчальної множини. Такі мережі широко застосовуються при вирішенні задач обробки сигналів і в системах управління. Основоположною роботою в цій області є книга Уїдроу і Хоффа, в якій вони і ввели скорочення ADALINE для адаптуються лінійних нейронів.

Архітектура лінійної мережі

Модель нейрона.

На рис. 3.1 показаний лінійний нейрон з двома входами. Він має структуру, подібну зі структурою персептрона; єдина відмінність полягає в тому, що використовується лінійна функція активації purelin.

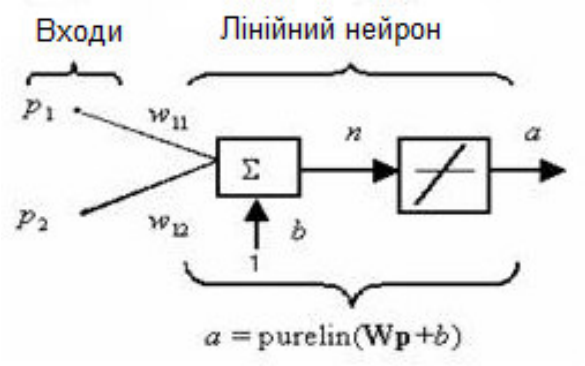


Рис. 3.1. Модель лінійного нейрона

Вагова матриця W в цьому випадку має тільки один рядок і вихід мережі визначається виразом:

$$a = purelin(n) = purelin(W_p + b) = W_p + b = w_{11}p_1 + w_{12}p_2 + b \quad (3.1)$$

Подібно персептрону, лінійна мережа задає в просторі входів розділяючу лінію, на якій функція активації n дорівнює 0 (рис. 3.2).

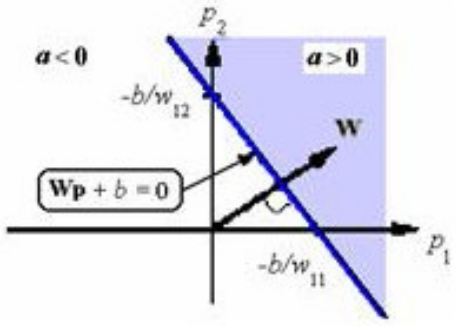
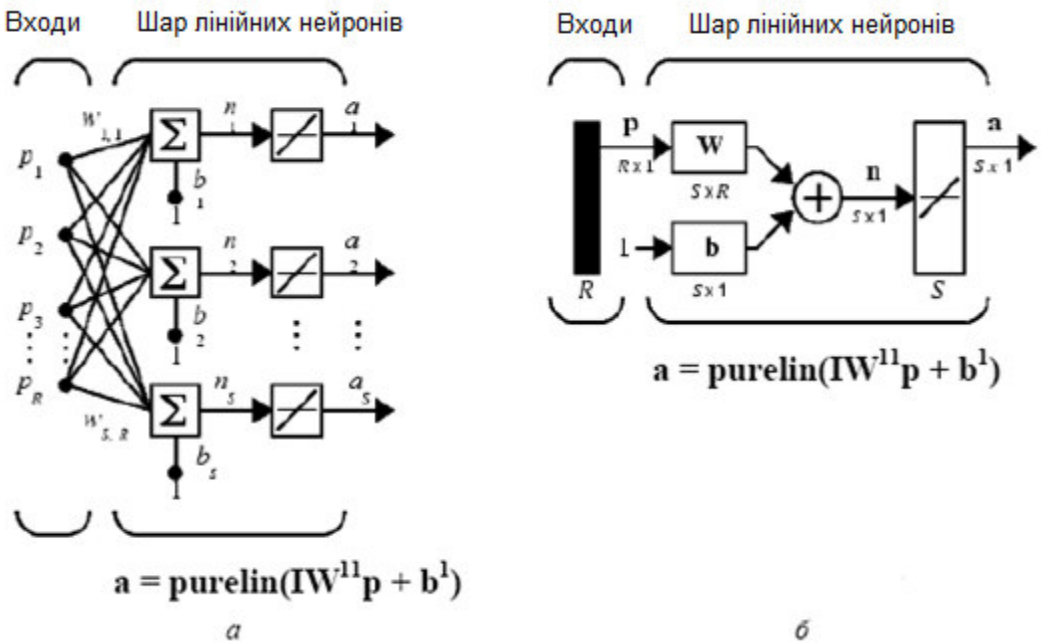


Рис.3.2. Лінійна мережа в просторі входів

Вектори входи, розташовані вище цієї лінії, відповідають позитивним значенням виходу, а розташовані нижче - негативним. Це означає, що лінійна мережа може бути застосована для вирішення задач класифікації. Однак така класифікація може бути виконана тільки для класу лінійно відокремлюваних об'єктів. Таким чином, лінійні мережі мають те ж саме обмеження, що і персептрон.

Архітектура мережі.

Лінійна мережа, показана на рис. 3.3, а, включає S нейронів, розміщених в одному шарі і пов'язаних з R входами через матрицю ваг W.



а *б*

Рис. 3.3. Архітектура лінійної мережі

На рис. 3.3, б показана укрупнена структурна схема цієї мережі, вектор виходу а якої має розмір $S \times 1$.

Навчання лінійної мережі.

Для заданої лінійної мережі та відповідного безлічі векторів входу і цілей можна обчислити вектор виходу мережі і сформувати різницю між вектором виходу і цільовим вектором, яка визначить деяку похибку. У процесі навчання мережі потрібно знайти такі значення ваг і зміщень, щоб сума квадратів відповідних похибок була мінімальною. Це завдання можна вирішити, тому що для лінійних систем функція квадратичної помилки є унімодальною. Як і для персептрона, застосовується процедура навчання з учителем, яка використовує навчальну множину виду:

$$\{p_1 t_1\}, \{p_2 t_2\}, ..., \{p_Q t_Q\} \tag{3.2}$$

де $p_1, p_2, ..., p_Q$ - входи мережі; $t_1, t_2, ..., t_Q$ - відповідні цільові виходи. Потрібно мінімізувати наступну функцію середньої квадратичної помилки:

$$mse = \frac{1}{Q}\sum_{k=1}^{Q} e(k)^2 = \frac{1}{Q}\sum_{k=1}^{Q} (t(k) - a(k))^2 \tag{3.3}$$

Правило найменших квадратів.

Для лінійної нейронної мережі використовується рекурентне навчальне правило найменших квадратів, яке є найбільш потужним, ніж навчальне правило персептрона. Правило найменших квадратів, або правило навчання WH (Уїдроу-Хоффа), мінімізує середнє значення суми квадратів помилок навчання. Процес навчання нейронної мережі полягає в наступному. Автори

алгоритму припустили, що можна оцінювати повну середню квадратичну похибку, використовуючи середню квадратичну похибку на кожній ітерації. Сформуємо приватну похідну по вагах і зсуву від квадрата похибки на k-й ітерації:

$$\begin{cases} \dfrac{\partial e^2(k)}{\partial w_{1,j}} = 2e(k)\dfrac{\partial e(k)}{\partial w_{1,j}}, \quad j = 1,\ldots,R; \\ \dfrac{\partial e^2(k)}{\partial b} = 2e(k)\dfrac{\partial e(k)}{\partial b}. \end{cases} \tag{3.4}$$

Підставляючи вираз для помилки у формі

$$e(k) = t(k) - \left(\sum_{j=1}^{R} w_{1,j} p_i(k) + b \right), \tag{3.5}$$

і отримаємо

$$\begin{cases} \dfrac{\partial e(k)}{\partial w_{1,j}} = -p_j(k); \\ \dfrac{\partial e(k)}{\partial b} = -1. \end{cases} \tag{3.6}$$

Тут pj (k) - j-й елемент вектора входу на k-й ітерації. Ці співвідношення лежать в основі навчального алгоритму WH

$$\begin{cases} \mathbf{w}(k+1) = \mathbf{w}(k) + e(k)\mathbf{p}^{\mathrm{T}}(k); \\ b(k+1) = b(k) + 2\alpha e(k). \end{cases} \tag{3.7}$$

Результат може бути узагальнений на випадок багатьох нейронів і представлений у наступній матричної формі:

$$\begin{cases} \mathbf{W}(k+1) = \mathbf{W}(k) + \mathbf{e}(k)\mathbf{p}^{\mathrm{T}}(k); \\ \mathbf{b}(k+1) = \mathbf{b}(k) + 2\alpha\, \mathbf{e}(k). \end{cases} \tag{3.8}$$

Тут помилка e і зсув b - вектори; α - параметр швидкості навчання. При великих значеннях α навчання відбувається швидко, однак при дуже великих значеннях може призводити до нестійкості.

Щоб гарантувати стійкість процесу навчання, параметр швидкості навчання не повинен перевищувати величини 1/max (| λ |), де λ - власне значення матриці кореляцій p * pт векторів входу. Використовуючи правило навчання WH і метод найшвидшого спуску, завжди можна навчити мережу так, щоб похибка навчання була мінімальною.

Функція learnwh призначена для налаштування параметрів лінійної мережі і реалізує наступне навчальне правило:

$$\begin{cases} \mathbf{dw} = lr * \mathbf{e} * \mathbf{p}^{\mathrm{T}}; \\ \mathbf{db} = lr * \sum_{i=1}^{Q} e_i, \end{cases} \quad (3.9)$$

де lr - параметр швидкості навчання. Максимальне значення параметра швидкості навчання, яке гарантує стійкість процедури налаштування, обчислюється за допомогою функції maxlinlr.

За допомогою демонстраційної програми demolin7 можна досліджувати стійкість процедури налаштування залежно від параметра швидкості навчання.

Процедура навчання лінійної мережі.

Для навчання лінійної нейронної мережі може бути застосована типова процедура навчання за допомогою функції train. Ця функція для кожного вектора входу виконує настройку ваг і зміщень, використовуючи функцію learnp. У результаті мережа буде налаштовуватися по сумі всіх корекцій. Кожен перерахунок для набору вхідних векторів називається епохою. Це й відрізняє процедуру навчання від процедури адаптації adapt, коли настройка параметрів реалізується при представленні кожного окремого вектора входу. Потім процедура train моделює налаштовану мережу для наявного набору векторів, порівнює результати з набором цільових векторів і обчислює середньоквадратичну помилку. Як тільки значення помилки стає менше заданого або вичерпано граничне число епох, навчання припиняється.

Передбачення сигналу.

Спробуємо застосувати мережу ADALINE для передбачення значень детермінованого процесу p (t). Звернемося до рис. 3.4.

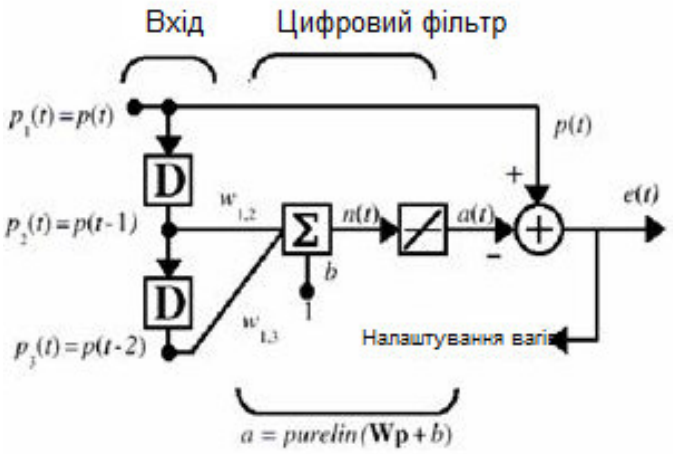


Рис. 3.4. Структура цифрового фільтра

Деякий сигнал надходить на лінію затримки так, що на її виході формуються 2 сигналу: p (t-1), p (t-2). Налаштування мережі реалізується за

допомогою М-функції adapt, яка змінює параметри мережі на кожному кроці з метою мінімізувати похибку e (t) = a (t) - p (t). Якщо ця погрішність нульова, то вихід мережі a (t) точно дорівнює p (t) і мережа виконує пророкування належним чином.

Багатовимірні цифрові фільтри.

Для проектування багатовимірних фільтрів слід застосовувати мережі ADALINE, в яких використовується більше одного нейрона. Це призведе до того, що замість вектора ваг входу буде використовуватися матриця ваг W, а замість єдиного зміщення - вектор зміщень b. Структурна схема такого багатовимірного фільтра показана на рис. 3.5.

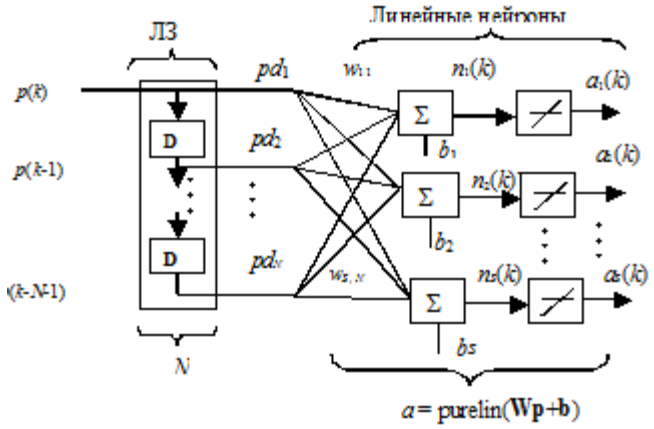


Рис. 3.5. Структура багатовимірного фільтра

Ця схема досить складна для сприйняття, і її можна представити в укрупненому вигляді (рис. 3.6, а). Якщо в лінії затримки (ЛЗ) потрібна показати більше деталей, то можна використовувати наступний варіант структурної схеми (рис. 3.6, б). а б

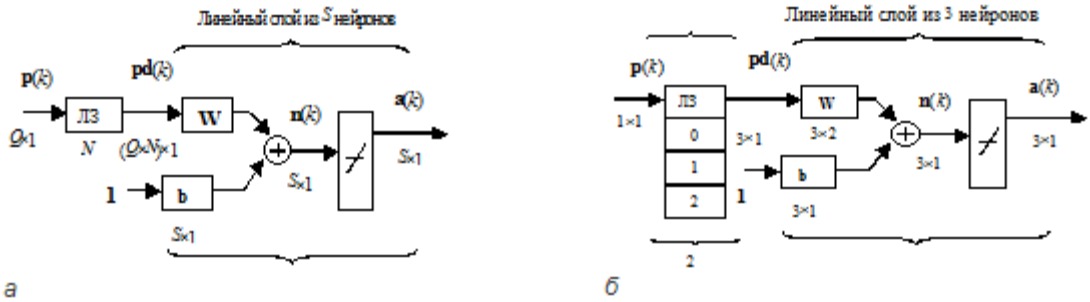


Рис. 3.6. Структура лінійних мереж

Тут ЛЗ представлена в розгорнутому вигляді, вказуючи, що на вхід вагової матриці подається поточне значення і значення з запізненням на 1 і 2 періоду дискретності. Можна використовувати і більше число блоків затримки, але вказувати їх треба в порядку зростання зверху вниз. На закінчення відзначимо основні переваги та обмеження лінійних мереж:

Одношарові лінійні мережі можуть вирішувати задачі лінійної апроксимації функцій і розпізнавання образів.

Одношарові лінійні мережі можуть бути або розраховані безпосередньо, або навчені з використанням правила навчання WH. Крім того, для їх налаштування можуть застосовуватися процедури адаптації.

Архітектура одношарової лінійної мережі повністю визначається завданням, яка повинна буде вирішена, причому число входів мережі і число нейронів у шарі визначається числом входів і виходів завдання.

Адаптуються лінійні мережі ADALINE знаходять велике практичне застосування при побудові цифрових фільтрів для обробки сигналів.

Лінійні нейронні мережі можуть бути успішно навчені тільки в тому випадку, коли входи і виходи пов'язані лінійно. Проте навіть у тому випадку, коли лінійна мережа не може знайти точного рішення, вона в змозі побудувати найбільш близьке рішення в сенсі мінімуму середньоквадратичної помилки за умови, що параметр навчання досить малий. Така мережа знайде найбільш точне рішення в рамках лінійної структури мережі. Це обумовлено тим, що поверхня помилки навчання є багатовимірним параболоїдом, мають єдиний мінімум, і алгоритм градієнтного спуску повинен привести рішення до цього мінімуму.

При роботі з моделями лінійних мереж можуть виникати ситуації, коли число параметрів, що настроюються недостатньо, щоб виконати всі умови; в цьому випадку говорять, що мережа перевизначена. Однак може мати місце і зворотна ситуація, коли число параметрів, що настроюються занадто велике, і в цьому випадку говорять, що мережа недовизначена. Проте в обох випадках метод найменших квадратів здійснює настройку, прагнучи мінімізувати помилку мережі. Ці ситуації пояснюються демонстраційними прикладами demolin4 і demolin.

Розв'язання лінійної задачі за допомогою лінійної нейронної мережі може бути встановлена наступним чином. Якщо сумарна кількість ваг і зміщень лінійної мережі S * R + S, де R - кількість входів, S - кількість шарів, дорівнює кількості пар векторів входу і цілі Q, то таке завдання можна вирішити за допомогою лінійної нейронної мережі. Це справедливо за винятком того випадку, коли вектори входу є лінійно залежними і використовується мережу без зсувів. Демонстраційний приклад demolin6 пояснює цю ситуацію.

Завдання на лабораторну роботу

Завдання на лабораторну роботу полягає у створенні моделі лінійної мережі, налаштувати та навчити лінійну мережу з використанням правила найменших квадратів та використати мережу ADALINE для прогнозування значень детермінованого процесу в середовищі MATLAB.

Методика виконання

1. Лінійну мережу з одним нейроном, показану на рис. 3.1, можна створити таким чином:

```
net = newlin ([-1 1; -1 1], 1);
```

Перший вхідний аргумент задає діапазон зміни елементів вектора входу; другий аргумент вказує, що мережа має єдиний вихід. Початкові ваги і зміщення за замовчуванням рівні 0.
Привласнимо ваг і зсуву такі значення:

```
W = net.IW{1,1}
b = net.b{1}
net.IW{1,1} = [2 3];
net.b{1} = -4;
```

Тепер можна промоделювати лінійну мережу для наступного пред'явленого вектора входу:

```
p = [5; 6];
a = sim (net, p)
```

Відповідь:
a = 24

2. На відміну від багатьох інших мереж настройка лінійної мережі для заданої навчальної множини може бути виконана за допомогою прямого розрахунку з використанням M-функції newlind. Припустимо, що задані наступні вектори, що належать навчаючій множині:

```
P = [1 -1.2];
T = [0.5 1];
```

Побудуємо лінійну мережу і промоделюємо її:

```
net = newlind(P,T);
Y = sim(net,P)
net.IW
net.b
```

Вихід мережі відповідає цільовому вектору.

```
Y =
   0.5000    1.0000
ans =
   [-0.2273]
ans =
   [0.7273]
```

Задамо наступний діапазон ваг і зміщень, розрахуємо критерій якості навчання і побудуємо його лінії рівня:

```
w_range=-1:0.1: 0; b_range=0.5:0.1:1;
ES = errsurf(P,T, w_range, b_range, 'purelin');
contour(w_range, b_range,ES,20)
hold on
plot(-2.2727e-001,7.2727e-001, 'x')
hold off
```

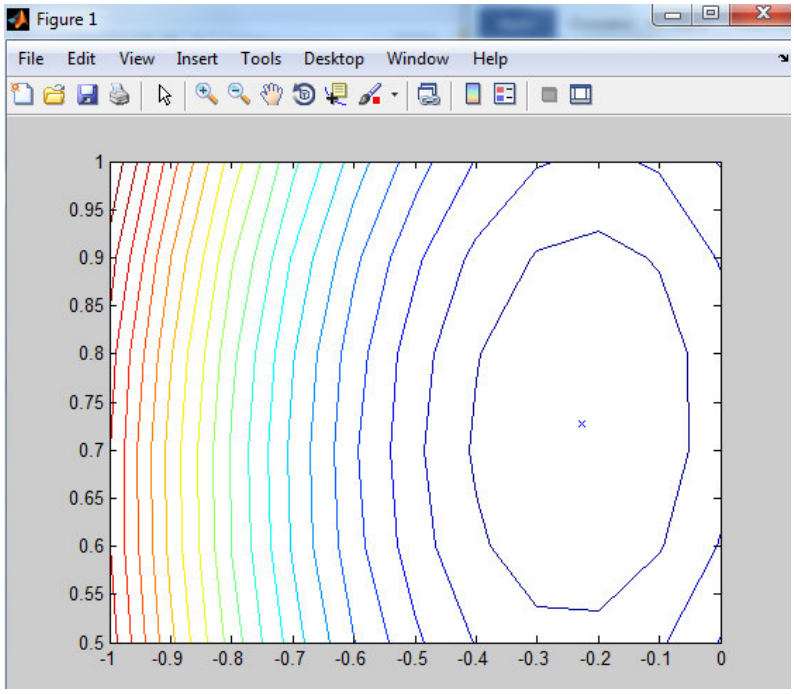


Рис. 3.7. Вікно відображення ліній рівнів навчання

На графіку знаком x відзначені оптимальні значення ваги і зміщення для даної мережі.

Демонстраційний приклад demolin1 пояснює структуру лінійної мережі, побудова поверхні помилок і вибір оптимальних налаштувань.

3. Лінійний нейрон навчається щоб знайти мінімальне рішення помилки для простого завдання. P визначає дві вхідні образи 1-елемент (вектор-стовпців). T визначає пов'язані цілі 1-елемент (вектор-стовпців).

ERRSURF обчислює помилки для нейрона з діапазоном можливих значення ваг і зсувів. PLOTES відображає ділянки цієї поверхні помилку з контуру ділянки під ним. Кращі значення ваг і зміщень є ті, які призводять до низької точці на поверхні помилок.

MAXLINLR знаходить швидкий стабільний курс навчання для підготовки лінійну мережу. NEWLIN створює лінійний нейрон. Щоб зрозуміти, що відбувається, коли швидкість навчання є занадто великою, потрібно збільшити швидкість навчання до 225% від рекомендованої вартості. NEWLIN приймає такі аргументи: 1) Rx2 матриця мінімального і максимального значень для вхідних елементів R, 2) Кількість елементів у вихідний вектор, 3) вектор Затримка на вході, і 4) Вартість навчання.

Для зупинки навчання встановлюється максимальну кількість епох.

Щоб показати шлях навчання ми будемо навчати тільки одину епоху в той час, і використовуємо PLOTEP. Сюжет показує історію навчання. Кожна точка представляє епоху і сині лінії показують кожну зміну відповідно до правила навчання (Уїдроу-Гоффа за замовчуванням).

```
P = [+1.0 -1.2];
T = [+0.5 +1.0];
w_range = -2:0.4:2;
b_range = -2:0.4:2;
maxlr = maxlinlr(P,'bias');
net = newlin([-2 2],1,[0],maxlr*2.25);
ES = errsurf(P,T,w_range,b_range,'purelin');
plotes(w_range,b_range,ES)
net.trainParam.epochs = 20;
%[net,tr] = train(net,P,T);
net.trainParam.epochs = 1;
net.trainParam.show = NaN;
h=plotep(net.IW{1},net.b{1},mse(T-sim(net,P)));
[net,tr] = train(net,P,T);
r = tr;
epoch = 1;
```

```
while epoch < 20
    epoch = epoch+1;
    [net,tr] = train(net,P,T);
    if length(tr.epoch) > 1
       h = plotep(net.IW{1,1},net.b{1},tr.perf(2),h);
       r.epoch=[r.epoch epoch];
       r.perf=[r.perf tr.perf(2)];
       r.vperf=[r.vperf NaN];
       r.tperf=[r.tperf NaN];
    else
       break
    end
end
tr=r;
plotperf(tr,net.trainParam.goal);
```

Функція train виводить навчену мережу і історію виконання підготовки (TR). Тут помилки наведені стосовно тренувальних епох.

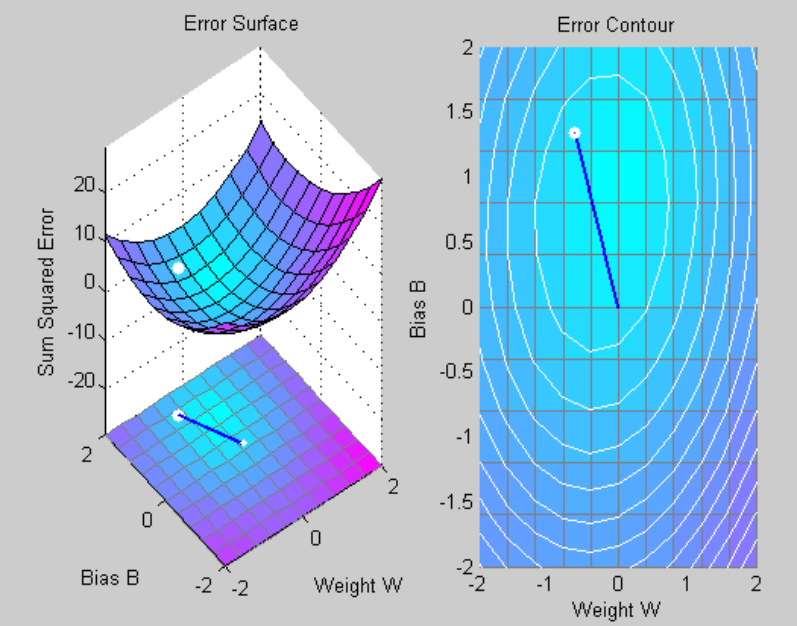


Рис. 3.8. Вікно шляху навчання для однієї епохи

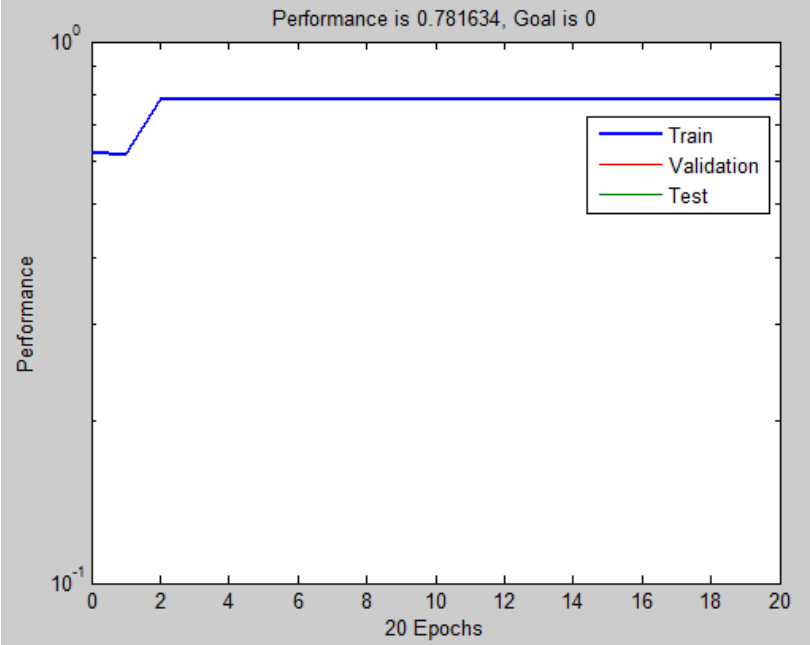


Рис. 3.9. Вікно помилки навчання відповідно до епох

Аналіз одержаних результатів. Висновки і рекомендації.

В протоколі вказати тему та мету лабораторної роботи, навести коротко теоретичні відомості. Протокол оформити відповідно до методики виконання та зробити скріншоти відповідно до завдання.

У висновку вказати як створюються моделі лінійної мережі, як відбувається їх налаштування та навчання в середовищі MATLAB.

Контрольні питання

1. Яка мережа називається лінійною?
2. Опишіть структуру лінійного нейрону?
3. Опишіть структуру цифрового фільтра?
4. Опишіть структуру багатовимірного фільтра?
5. Як відбувається створення лінійної мережі в середовищі Matlab?
6. Як відбувається навчання лінійної мережі?

Лабораторна робота № 4
Побудова нейронної мережі засобами Deductor Studio

Мета: Засвоїти основні механізми створення інтелектуальних нейронних мереж в середовищі Deducor Studio.

Загальні відомості

Deductor є аналітичною платформою, тобто основою для створення закінчених прикладних рішень. Технології, що реалізовані в Deductor дозволяють на базі єдиної архітектури пройти всі етапи побудови аналітичної системи: від створення сховища даних до автоматичного підбору моделей і візуалізації отриманих результатів.

Deductor надає аналітикам інструментальні засоби, які є необхідними для вирішення найрізноманітніших аналітичних завдань: корпоративна звітність, прогнозування, сегментація, пошук закономірностей – ці та інші завдання, де застосовуються такі методики аналізу, як OLAP, Knowledge Discovery in Databases і Data Mining. Deductor є ідеальною платформою для створення систем підтримки приняття рішень.

Deductor є повнофункціональною платформою для вирішення завдань Knowledge Discovery in Databases, що дозволяє провести всі вищеописані кроки.

1. Підготовка початкового набору даних. До складу системи входить Deductor Warehouse – багатовимірне сховище даних, що орієнтоване на рішення задач консолідації інформації з різнорідних джерел і швидкого видобутку потрібного набору даних. Deductor Warehouse підтримує потужний семантичний прошарок, що дозволяє кінцевому користувачеві оперувати бізнес термінами для отримання потрібних даних. Окрім власного сховища Deductor підтримує роботу і з іншими джерелами: Oracle, DB2, MS SQL, Informix, Sybase, Interbase, DBase, FoxPro, Paradox, MS Access, CSV (текстові файли з роздільниками), ODBC, ADO. Для забезпечення максимальної швидкодії Deductor підтримує прямий (direct) доступ до більшості найпопулярніших баз даних.
2. Передобробка. Deductor містить великий набір механізмів передобробки і очищення даних: заповнення пропусків, редагування аномалій, очищення від шумів, згладжування, фільтрація і багато чого іншого з можливістю комбінування методів передобробки.
3. Трансформація, нормалізація даних. Deductor містить великий набір механізмів трансформації даних, що дозволяють провести всю

підготовчу роботу для подальшого аналізу. Окрім цього, система містить широкий спектр механізмів нормалізації для всіх типів даних: числових, рядкових, дата/час і логічних.

4. Data Mining. У складі пакету містяться алгоритми, що реалізують популярні та ефективні методи Data Mining: нейронні мережі, дерева рішень, самоорганізовані карти Кохонена, асоціативні правила тощо.

5. Постобробка даних. Результати будь-якої обробки можуть бути відображені за допомогою великого набору механізмів візуалізації: OLAP, таблиці, діаграми, дерева тощо. Для деяких механізмів передбачені спеціалізовані візуалізатори, які забезпечують легкість інтерпретації результатів. Результати можна експортувати для подальшої обробки за допомогою інших додатків. Це дає можливість ефективно використовувати отримані знання або моделі на інших даних.

Deductor задовольняє всім вимогам для успішної взаємодії з експертом (аналітиком):

- Єдина платформа, в якій можна пройти всі етапи Knowledge Discovery in Databases;
- Всі операції проводяться за допомогою майстрів, завдяки яким знижуються вимоги до знання експертом математичного апарату;
- Можливість довільного комбінування будь-яких методів обробки;
- Великий набір методів візуалізації отриманих результатів;
- Пакетне виконання всій дій по обробці даних.

Крім того, Deductor містить спеціальний додаток – Deductor Viewer для кінцевих користувачів, що дозволяє їм отримати кінцеві результати, без необхідності розбиратися в методах аналізу даних. Для отримання результату користувачеві достатньо просто вибрати потрібний звіт, все інше буде виконано автоматично.

Вирішувані задачі

Технології, що реалізовані в Deductor можуть використовуватися як в комплексі, так і окремо для вирішення широкого спектру бізнес-проблем:

- Системи корпоративної звітності. Готове сховище даних і гнучкі механізми передобробки, очищення, завантаження, візуалізація дозволяє швидко створювати закінчені системи звітності у стислі терміни.
- Обробка нерегламентованих запитів.
- Аналіз тенденцій і закономірностей, планування, ранжування. Простота використання та інтуїтивно зрозуміла модель даних дозволяє провести аналіз за принципом "якщо, тоді", співвіднести гіпотези з відомостями, що зберігаються в базі даних, знайти аномальні значення, оцінити наслідки ухвалення бізнес рішень.

• Прогнозування. Якщо побудувати модель на історичних даних, можна використати її для прогнозування ситуації в майбутньому. В міру зміни ситуації, немає необхідності перебудовувати все, потрібно лише довчити модель.

• Управління ризиками. Алгоритми, що реалізовані в системі дозволять достатньо точно визначитися з тим, які характеристики об'єктів і як впливають на ризики, завдяки чому можна прогнозувати виникнення ризикової події і завчасно прийняти необхідні заходи до зниження розміру можливих несприятливих наслідків.

• Аналіз даних маркетингових та соціологічних досліджень. Аналізуючи відомості про споживачів, можна визначити, хто є клієнтом і чому. Як змінюються їх уподобання в залежності від віку, освіти, соціального та матеріального стану і множини інших показників. Розуміння цього сприятиме правильному позиціонуванню продуктів та стимулюванню продажів.

• Діагностика. Механізми аналізу, що закладені в системі Deductor, з успіхом можна застосувати в медичній діагностиці та діагностиці складного обладнання. Наприклад, можна побудувати модель на основі відомостей про відмови. При її допомозі швидко локалізувати проблеми і знайти причини збоїв.

• Виявлення об'єктів на основі нечітких критеріїв. Часто зустрічається ситуація, коли необхідно виявити об'єкт, базуючись не на чітких критеріях, таких, як вартість, технічні характеристики продукту, а на розмитих формулюваннях, наприклад, знайти продукти, які є подібними до зазначених, з погляду споживача.

Це є лише невеликий список вирішуваних задач. Фактично йдеться про будь-які завдання, де потрібно консолідувати дані, відобразити їх різними способами, побудувати моделі і застосувати отримані моделі до нових даних.

Склад системи

Deductor складається з п'яти частин:

• **Studio** – програма, що реалізує функції імпорту, обробки, візуалізації та експорту даних. Deductor Studio може функціонувати і без сховища даних, отримуючи інформацію з інших джерел, але оптимальним буде їх сумісне використання. В Deductor Studio міститься повний набір механізмів, що дозволяє отримати інформацію з певного джерела даних, провести весь цикл обробки (очищення, трансформацію даних, побудову моделей), відобразити отримані результати в зручний спосіб (OLAP, діаграми, дерева.) та експортувати результати на зовні.

• **Viewer** – робоче місце кінцевого користувача. Дозволяє відокремити процес побудови сценаріїв від використання вже готових моделей. Всі складні операції по підготовці сценаріїв обробки виконуються аналітиками-експертами за допомогою Deductor Studio, а Deductor Viewer забезпечує користувачам простій спосіб роботи з готовими результатами, приховує від них всі складнощі побудови моделей і не висуває високих вимог до кваліфікації співробітників.

• **Warehouse** – багатовимірне сховище даних, що акумулює всю необхідну інформацію для аналізу обраної області. Використання єдиного сховища дозволяє забезпечити несуперечність даних, їх централізоване зберігання і автоматично забезпечує підтримку процесу аналізу даних.

• **Server** – служба, що забезпечує віддалену аналітичну обробку даних. Дозволяє автоматично обробляти дані і перенавчати моделі на сервері, оптимізує виконання сценаріїв за рахунок кешування проектів і використання багатопотокової обробки.

• **Client** – клієнт доступу до Deductor Server. Забезпечує доступ до сервера із зовнішніх застосувань та управління його роботою.

Архітектура, що реалізована в Deductor дозволяє досягти максимальної гнучкості при створенні закінченого рішення. Завдяки даній архітектурі можна зібрати в одному аналітичному застосуванні всі необхідні інструменти аналізу і реалізувати автоматичне виконання підготованого сценарію.

Технологічна платформа містить засоби, що дозволяють максимально скоротити терміни розробки, швидко створювати і виводити на ринок нові прикладні рішення і швидко адаптувати їх відповідно до змінних вимог підприємств. Можливості платформи забезпечують не лише високу швидкість первинної розробки продукту, але і його подальшу швидку адаптацію.

Створення закінченого рішення займає мало часу. Досить отримати дані, визначити сценарій обробки і задати місце для експорту отриманих результатів. Наявність могутнього набору механізмів обробки і візуалізації дозволяє просуватися по кроках, від найпростіших способів аналізу до складніших, таким чином, перші результати користувач отримує практично відразу, але при цьому можна легко нарощувати потужність рішення.

Новий погляд на дані

Deductor дозволяє абсолютно по новому розглядати дані, витягуючи з них максимум цінної інформації. Він об'єднує всі необхідні для аналізу інструменти, надає користувачам величезні можливості:

• Могутня аналітична платформа;

• Сучасні самонавчальні механізми аналізу;

- Єдине сховище даних;
- Єдиний користувацький інтерфейс для будь-яких механізмів аналізу;
- Пакетне виконання сценаріїв обробки;
- Віддалена аналітична обробка;
- Розмежування роботи аналітика від роботи кінцевого користувача.

Завдання на лабораторну роботу

Завдання на лабораторну роботу полягає у створенні багатошарової нейронної мережі в середовищі Deducor Studio та навчанні мережі з використанням відомих алгоритмів навчання.

Методика виконання

На першому етапі виконання роботи необхідно здійснити імпорт даних з інших підсистем введення. Враховуючи, що в академічній версії найбільш простим введення даних є формування *.txt файлу в якому є можливість задати статистичні дані. Як приклад це показано на рис. 4.1.

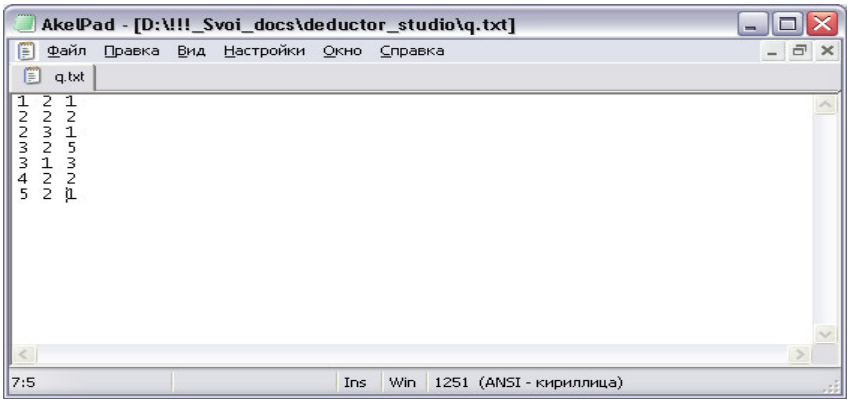


Рис. 4.1. Введення даних в файл

На наступному етапі дані завантажуються в середовище за допомогою майстра імпорту рис. 4.2, 4.3.

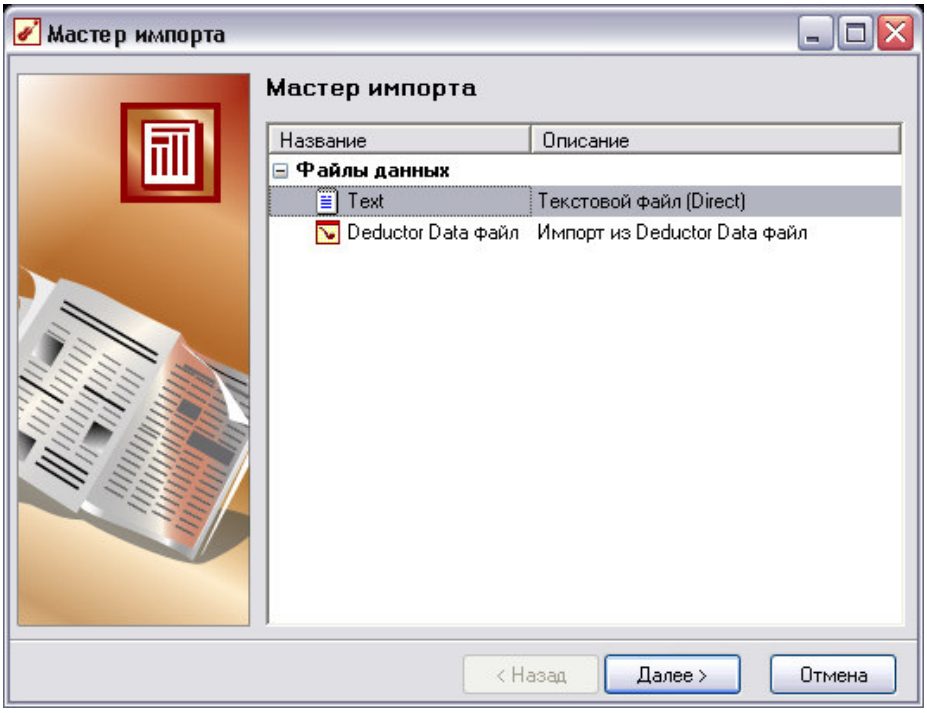


Рис. 4.2. Майстер завантаження даних

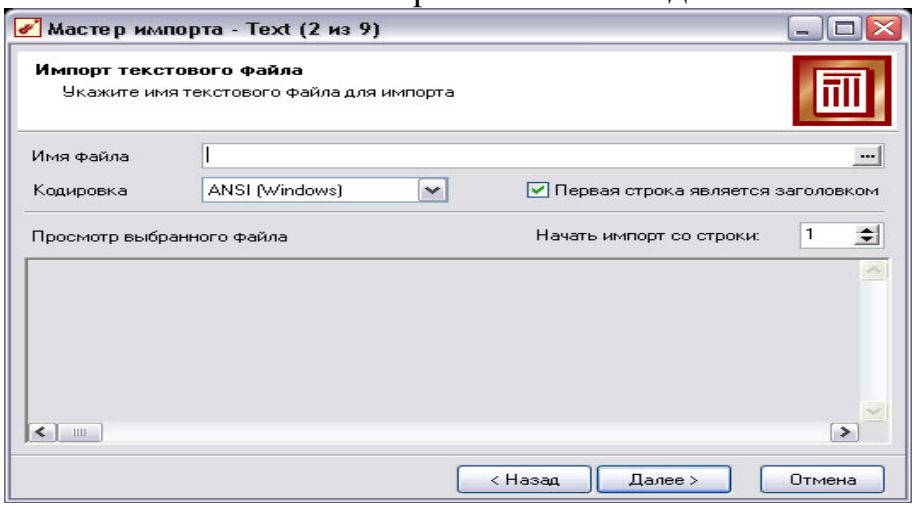


Рис. 4.3. Робота майстра завантаження даних

Допускається вручну ввести шлях до файлу в рядку поля Ім'я файлу. Є можливість використовувати як абсолютні, так і відносні шляхи для файлів.

Вони вказуються щодо поточної директорії Deductor. При відкритті Deductor поточної директорією є директорія файлу проекту. Тому, якщо файл проекту і текстові файли розташовуються в одній папці, то використання відносних шляхів в Майстрі імпорту дозволить НЕ перенастроювати вузли імпорту при зміні розташування папки на жорсткому диску.

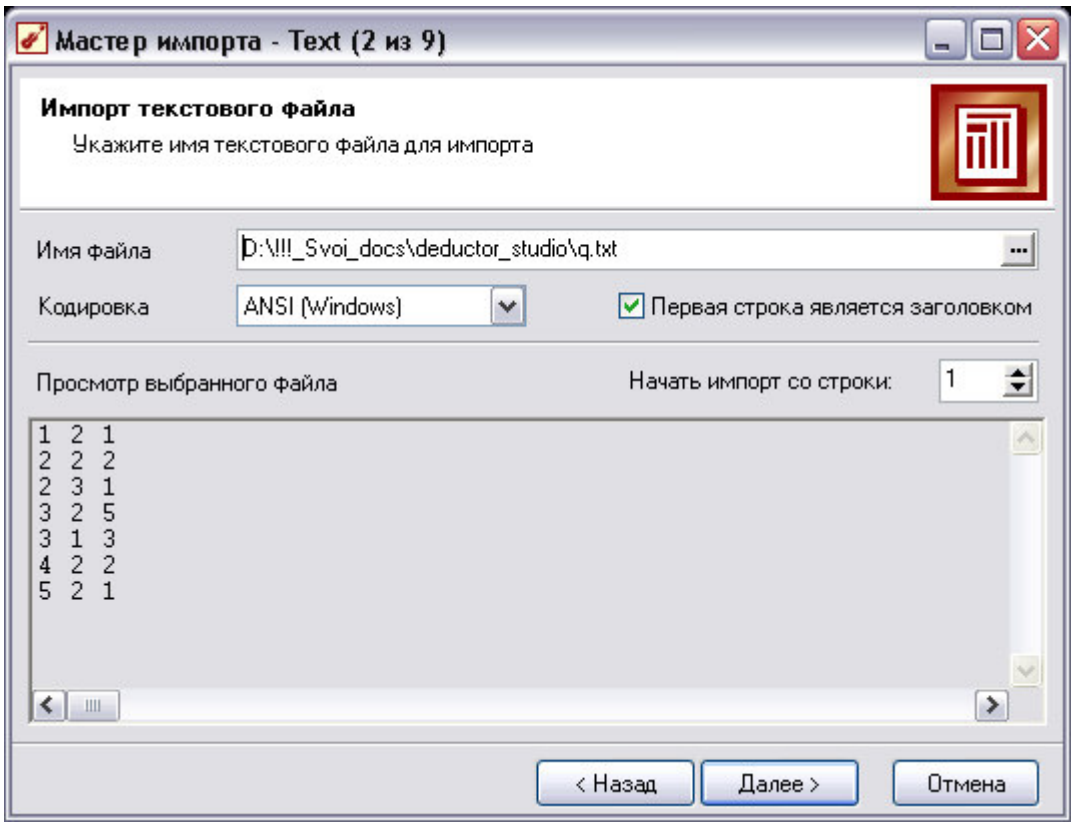


Рис. 4.4. Вікно імпорту даних та їх перегляду

Тут також доступні налаштування:

- **Почати імпорт з рядка** - номер рядка, починаючи з якої буде робитися імпорт даних з файлу.
- прапор **Перший рядок є заголовком** - установка прапорця означає, що вузол буде імпортувати дані з урахуванням того, що всі записи першого рядка є заголовками стовпців.
- **Кодування** - ANSI (Windows) або ANCII (MS DOS).

На кроці **Налаштування параметрів імпорту** потрібно налаштувати параметри імпорту даних з текстового файлу, так як існує кілька форматів структурованих текстових файлів.

Доступні опції:

• перемикач **Формат вихідних даних**, який визначає символ-роздільник у файлі (наприклад: символ табуляції, пробіл, кома). Роздільник найчастіше присутня. Якщо ж ні, то потрібно вибрати перемикач **Фіксованої ширини** (поля мають задану ширину), а пізніше встановити ширину кожного поля.

• **Обмежувач строк** - при завданні даного параметра необхідно вказати, який саме обмежувач строкового значення потрібно використовувати при імпорті даних з текстового файлу. Зазвичай таким обмежувачем є символ подвійної лапки ".

• **Роздільник дробової і цілої частини числа** - при завданні даного параметра необхідно вказати символ, що розділяє дробову і цілу частини в числових значеннях, що містяться у файлі.

• **Роздільник компонентів дати** - вказується символ, що розділяє компоненти дати у відповідних значеннях, що містяться в

файлі.

• **Роздільник компонентів часу** - вказується символ, розділяє компоненти часу у відповідних значеннях, містяться у файлі.

• **Формати Дати/Часу** - вказуються формати дати / часу, використовувані в імпортованому файлі.

• **Представлення значень** - опція для полів логічного типу, яке може приймати одне з трьох значень - істина (true), брехня (False) і пусте значення (null). Визначає регламент записи в ці значення. Так, при налаштуваннях за замовчуванням для будь-якого логічного поля значення Так буде сприйматися як істина, Ні - як брехня.

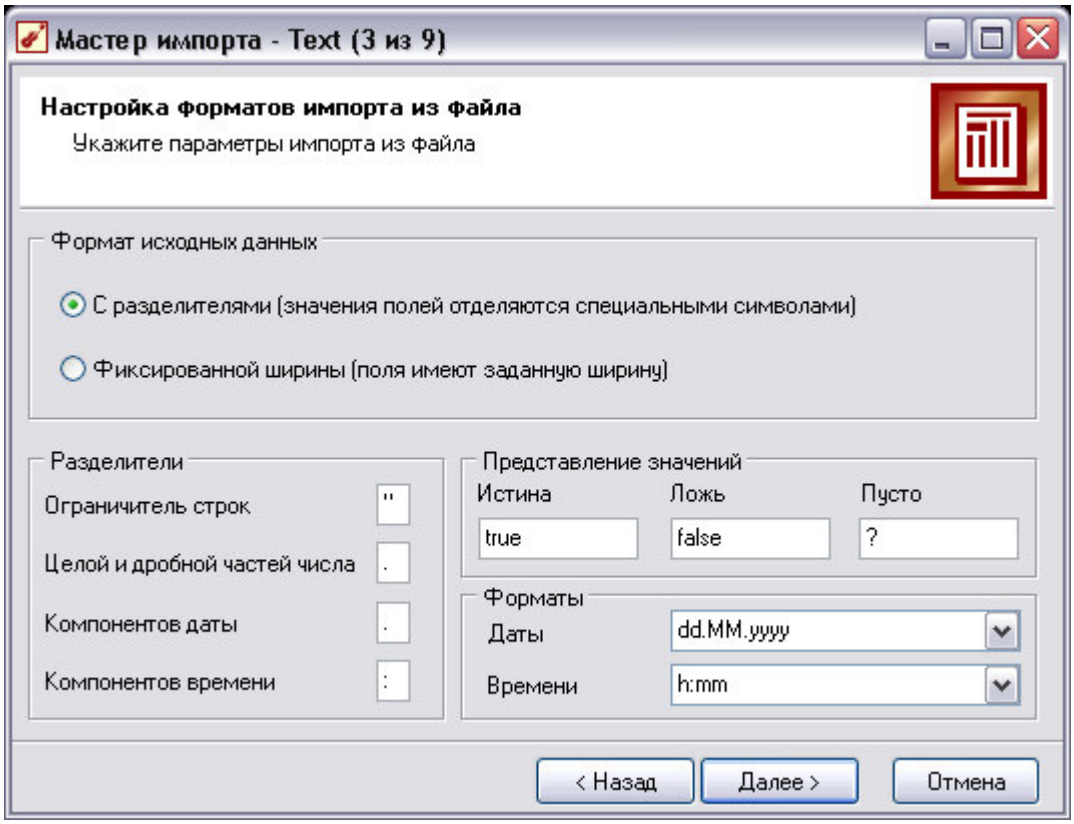


Рис. 4.5. Вікно налаштування формату імпорту із файлу

В якості роздільників, уявлень значень і форматів по замовчуванням завжди пропонуються системні налаштування операційної системи. Тому при імпорті необхідно звертати увагу на їх відповідність формату в

імпортованому текстовому файлі. Наступне вікно майстра залежить від встановленого перемикача в прапорці **Формат вихідних даних**. Якщо був обраний формат **З роздільниками**, то з'явиться вкладка, на якій потрібно явно вказати символ-роздільник (за замовчуванням - табуляція). Тут же знаходиться прапор **Вважати послідовні роздільники одним** - у разі послідовно йдуть символів-роздільників вони сприйматимуться за один. Таке буває, наприклад, коли символом-роздільником виступають кілька пробілів. Попередній перегляд текстового файлу у вигляді таблиці внизу дозволяє переконатися в коректності вибору налаштувань імпорту навіть не запускаючи його.

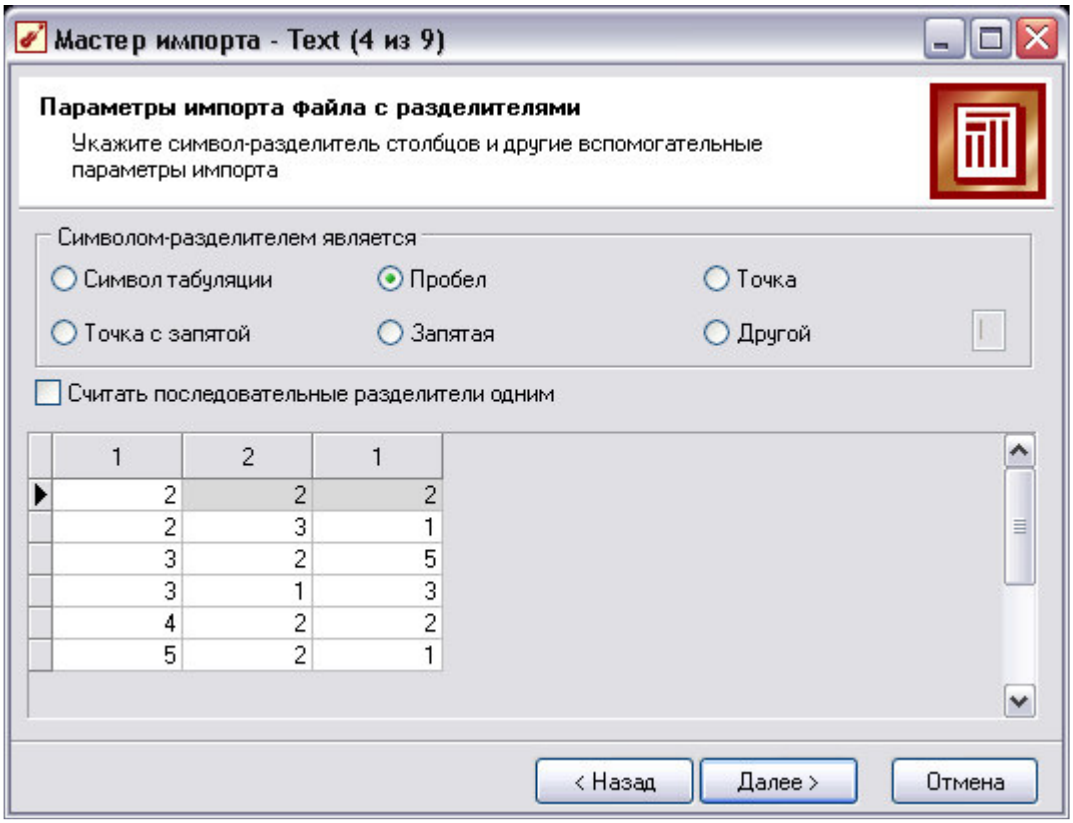


Рис. 4.6. Вікно встановлення формату символів-роздільників імпортованих даних

На кроці **Налаштування параметрів стовпців** потрібно налаштувати наступні параметри стовпців імпортованих даних, вказавши відповідні значення в полях.

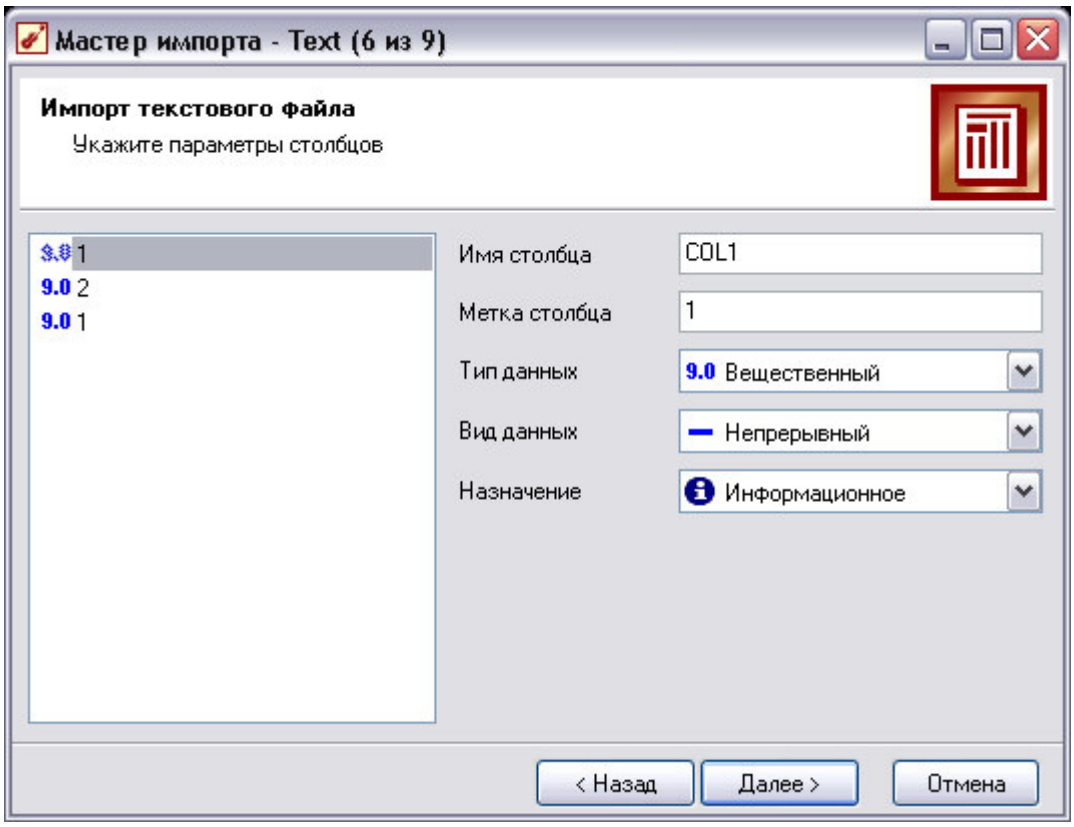


Рис. 4.7. Вікно налаштувань параметрів стовпців

Ім'я стовпця - вказується ім'я, яке буде служити ідентифікатором стовпця в наступних вузлах. За замовчуванням пропонується заголовок стовпця з текстового файлу, якщо на попередньому кроці був встановлений прапорець Перший рядок є заголовком.

Інакше будуть запропоновані імена типу COL1, COL2 і т.д. Можна ввести будь-які імена, які семантично відображають вміст стовпця, однак допускаються тільки латинські символи, і ім'я шпальти має бути унікальним в межах всіх стовпців імпортованого файлу.

Мітка шпальти - назва, під яким даний стовпець буде видно в візуалізаторах. Допускаються будь-які символи, унікальність імен НЕ обов'язкове.

Тип даних - вказується тип даних, що містяться в стовпці. Тип вибирається зі списку, що відкривається клацанням по кнопці в правій частині поля:

Тип Опис

0/1 логічний - дані в полі можуть приймати тільки два значення - 0 або 1

дата / час - поле містить дані типу дата / час

9.0 речовий - числа з плаваючою точкою

12 цілий - цілі числа

ab строковий - рядки символів

Безперервними можуть бути тільки числові дані. Дискретний характер носять, як правило, строкові дані, але не завжди. Дискретними можуть бути призначені залежно від контексту розв'язуваної задачі дані цілого типу, рідше - речового. Вид даних стовпця впливає на:

- алгоритм розрахунку статистики по стовпці;
- роботу аналітичних алгоритмів.

Призначення - визначає порядок використання поля набору даних, отриманого в результаті імпорту стовпця (поля), при подальшій обробці імпортованих даних:

- первинний ключ – поле буде використовуватися в якості первинного ключа;
- вхідне – поле набору даних, побудованих на основі стовпця, буде являтися вхідним полем обробника (нейронної мережі, дерева рішень і т.д.);
- вихідне – поле набору даних, побудоване на основі стовпця, буде бути вихідним полем обробника (наприклад, цільових перевірок ним полем для навчання нейронної мережі);
- інформаційне - поле містить допоміжну інформацію, яку часто корисно відображати, але не слід використовувати при обробці;
- вимірювання - поле буде використовуватися в якості вимірювання в багатовимірній візуалізації;
- атрибут - поле містить опис властивостей або параметрів деякого об'єкта;
- факт - значення поля будуть використані в якості фактів в багатовимірної візуалізації;
- ID транзакція - поле, що містить ідентифікатор подій, відбуваються спільно (одночасно);
- елемент поле, що містить елемент транзакції (подія).

На кроці **Запуск процесу імпорту** стартує сам процес імпорту даних з раніше налаштованими параметрами. Хід процесу імпорту відображається за допомогою індикатора. Якщо процес імпорту зупинився, це сигналізує про можливих помилок при читанні даних. У цьому випадку з'являється вікно з повідомленням про помилку.

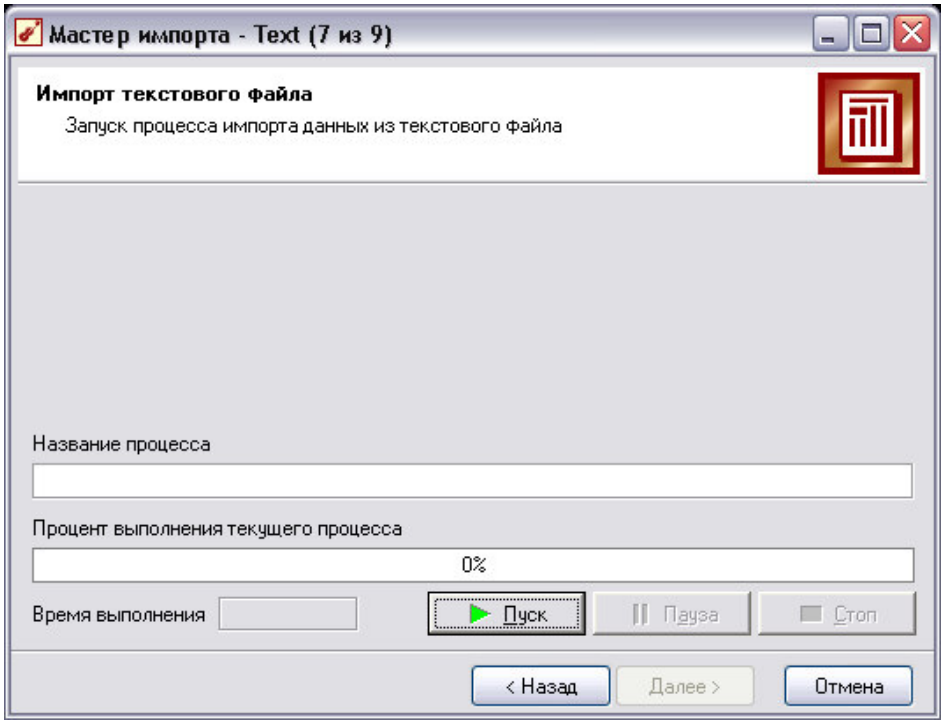


Рис. 4.8. Вікно запуску імпорту даних

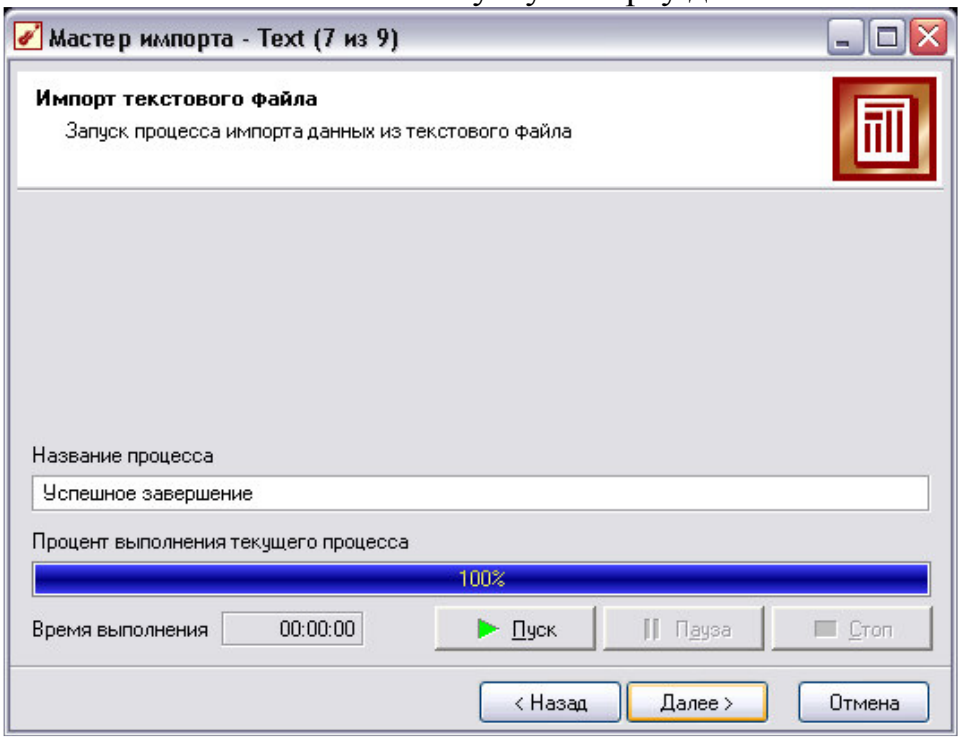


Рис. 4.9. Вікно стану процесу запуску даних

Під обробкою в Deductor мається на увазі будь-яка дія, що пов'язана з якимось перетворенням даних, наприклад, фільтрація, побудова моделі, очищення та інше. Власне в блоці "Обробка даних" і виробляються

найважливіші з точки зору аналізу дії. Найбільш істотною особливістю механізмів обробки, реалізованих в Deductor, є те, що отримані в результаті їх застосування дані можна знову обробляти будь-яким методом з доступних. Таким чином, є можливість будувати скільки завгодно складні сценарії.

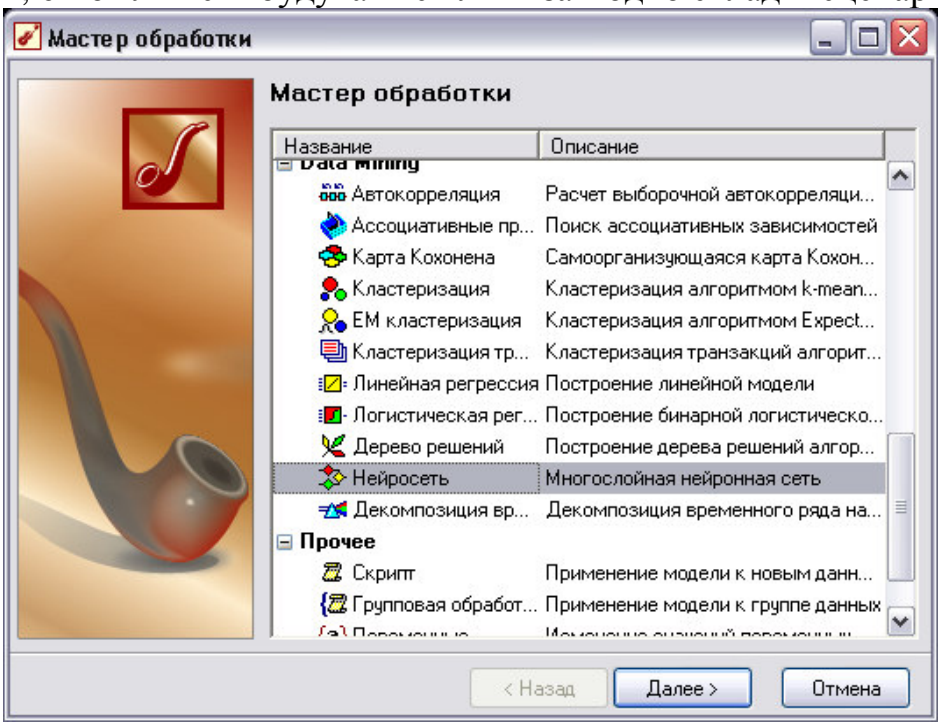


Рис. 4.10. Вікно вибору створення нейронної мережі для обробки даних

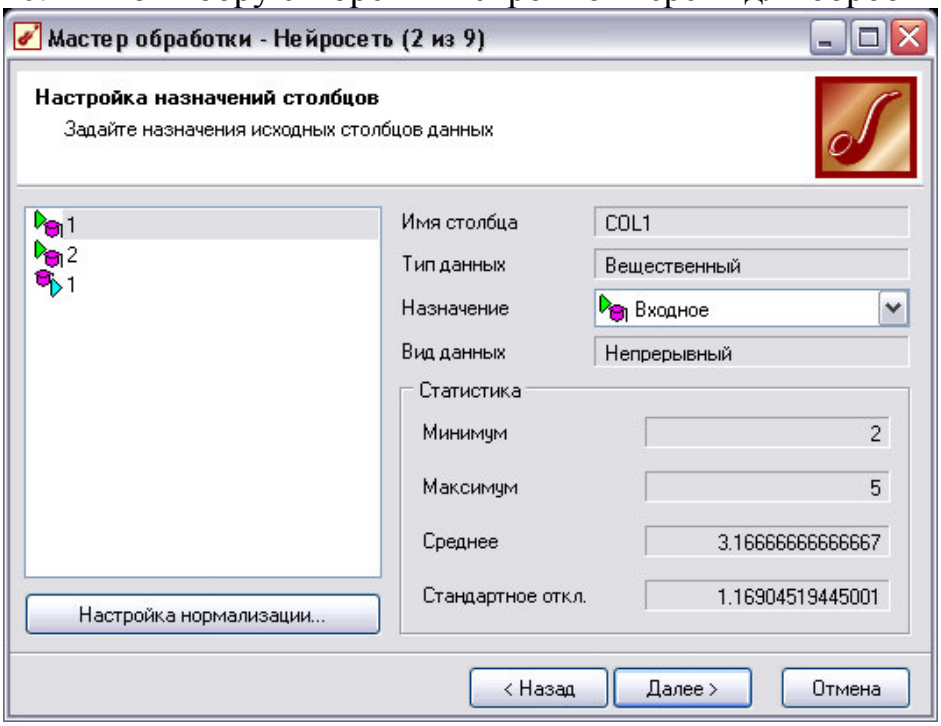


Рис. 4.11. Вікно налаштувань призначень стовпців

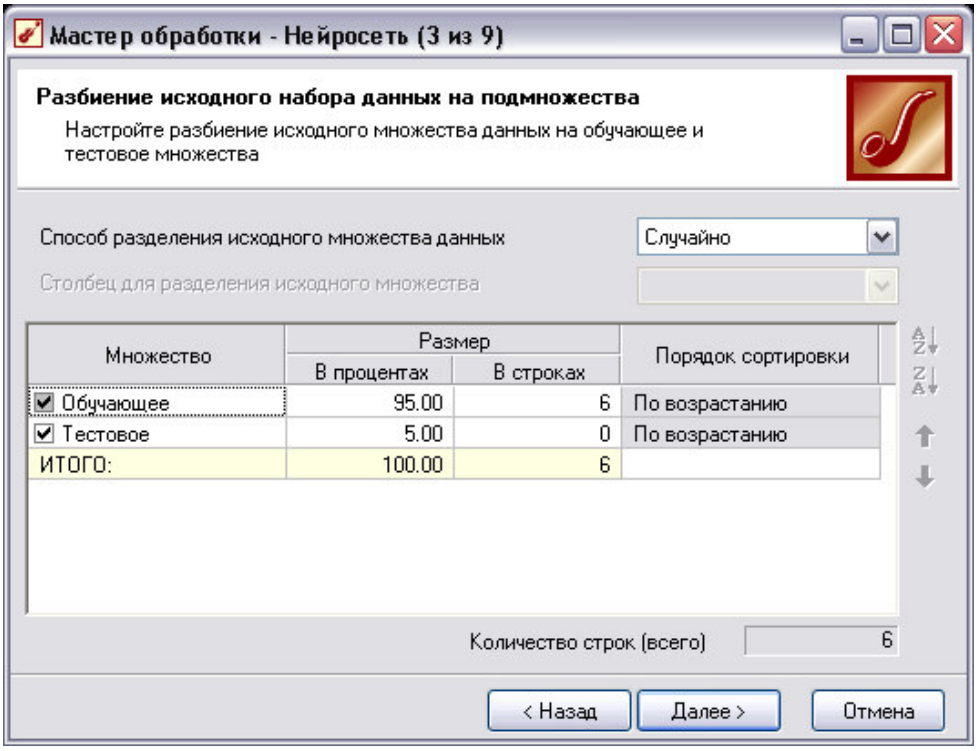


Рис. 4.12. Вікно розбиття вихідного набору даних на підмножини

Перейдемо до наступного кроку, на якому відзначимо необхідну кількість слоїв і нейронів в нейромережі.

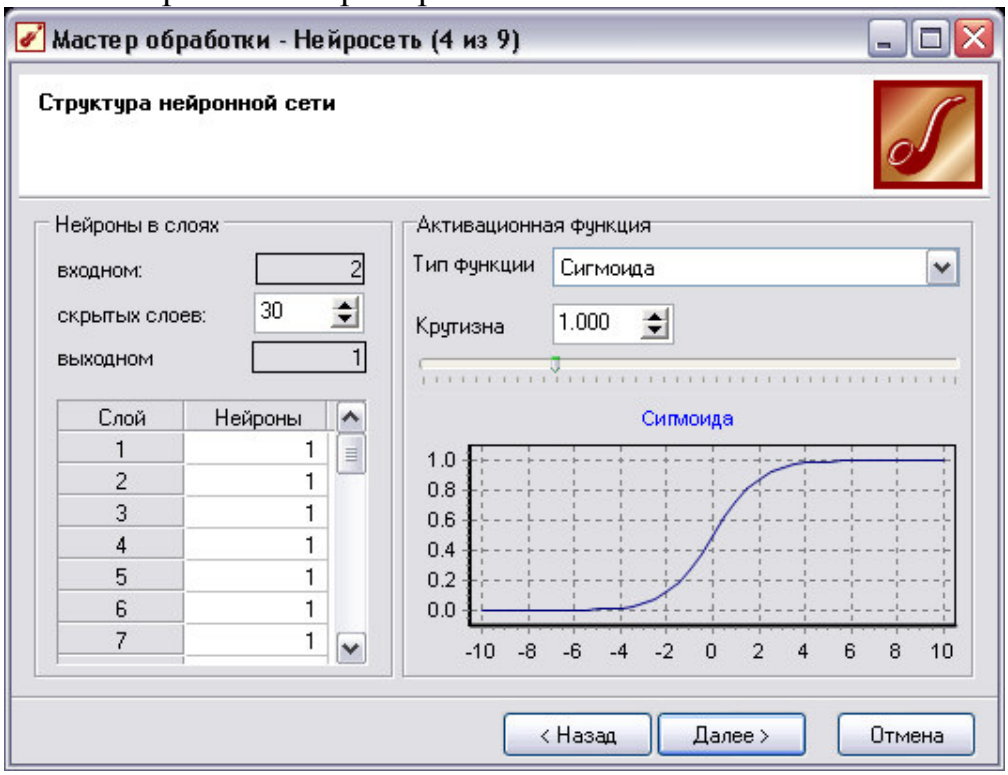


Рис. 4.13. Структура нейронної мережі

Одним із серйозних недоліків алгоритму з зворотним поширенням помилки, використовуваного для навчання багатошарових нейронних мереж, є занадто довгий процес навчання, що робить непридатним використання даного алгоритму для широкого кола завдань, які вимагають швидкого вирішення. В даний час відомо достатню кількість алгоритмів прискорюють процес навчання, таких як: QuickProp, метод сполучених градієнтів, метод Левенберга-Маркара. Розглядається один з таких алгоритмів названий Resilient Propagation (Rprop), який був запропонований М. Рідміллером (M. Riedmiller) і Г. Брауном (H.Braun).

На відміну від стандартного алгоритму Backprop, RProp використовує тільки знаки часткових похідних для підстроювання вагових коефіцієнтів. Алгоритм використовує так зване «навчання за епохами», коли корекція ваг відбувається після пред'явлення мережі всіх прикладів з навчальної вибірки.

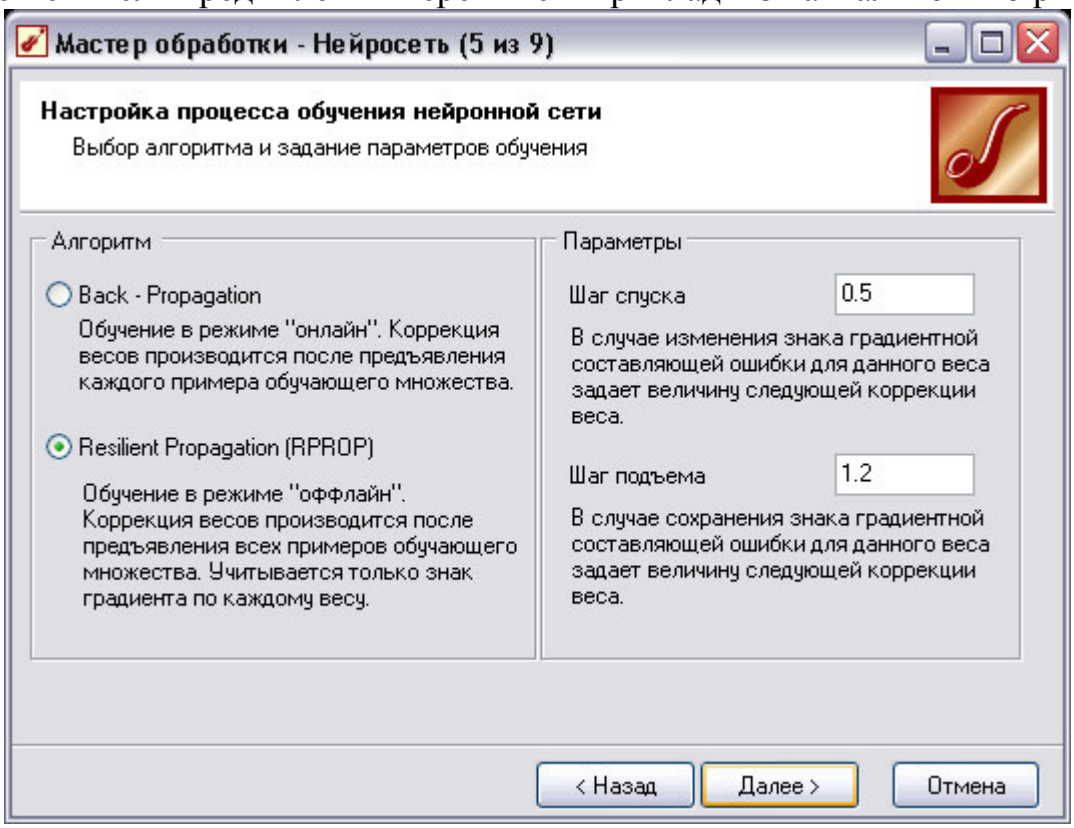


Рис. 4.14. Вікно налаштування процесу навчання

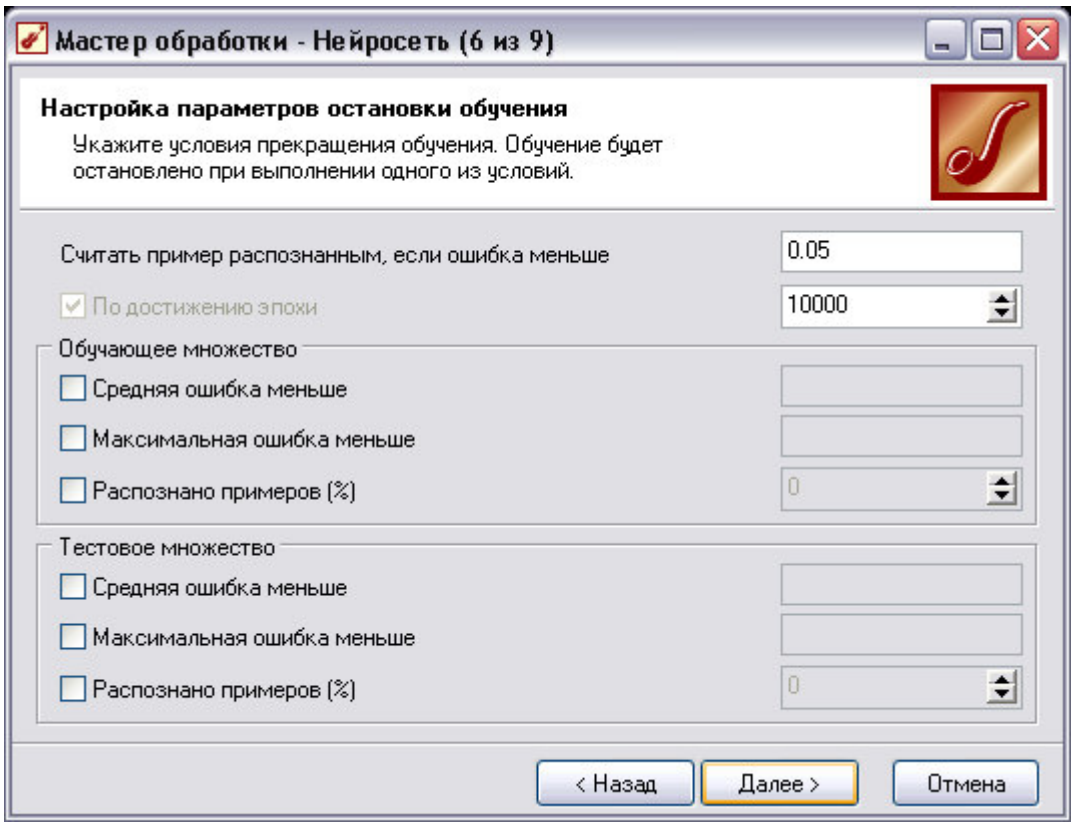


Рис. 4.15. Вікно налаштування параметрів зупинки навчання

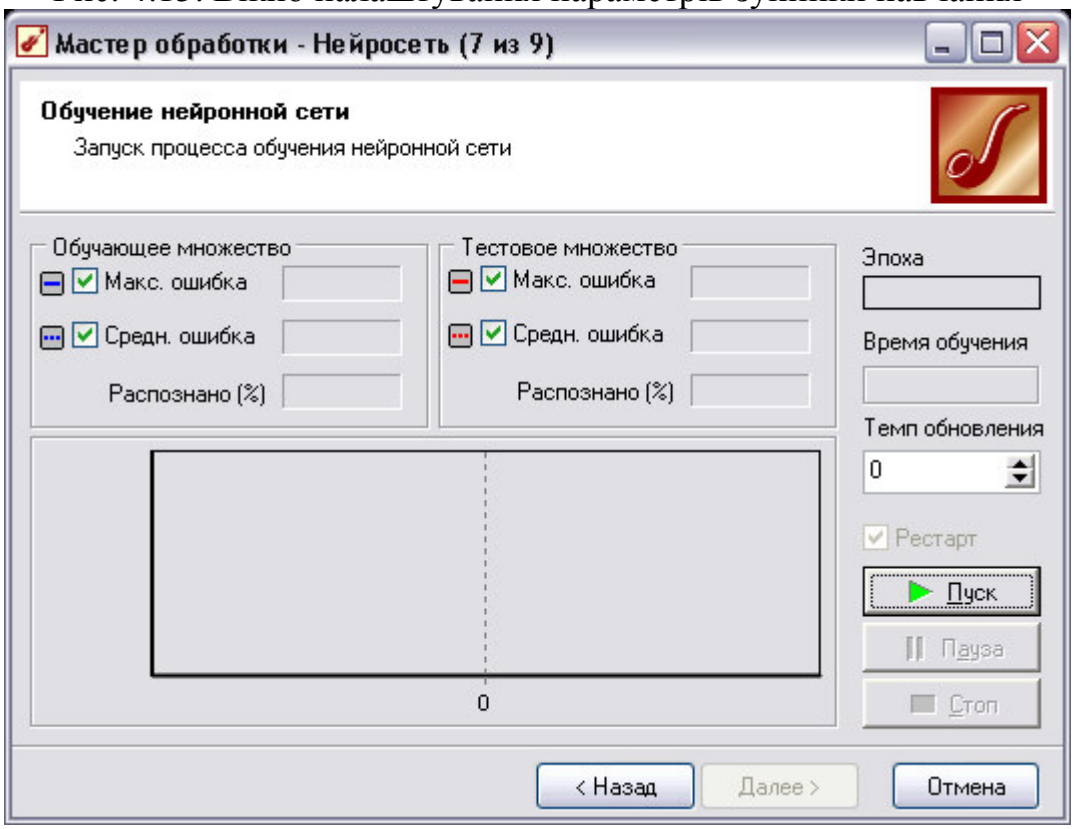


Рис. 4.16. Вікно навчання нейронної мережі

Спосіб початкової ініціалізації карти визначає, як будуть встановлені початкові ваги нейронів карти. Вдало вибраний спосіб ініціалізації може істотно прискорити навчання і привести до отримання більш якісних результатів. Доступні три варіанти:

- Випадковими значеннями - початкові ваги нейронів будуть ініційовані випадковими значеннями.
- З навчальної множини - в якості початкових терезів будуть використовуватися випадкові приклади з навчальної множини.
- З власних векторів - початкові ваги нейронів карти будуть проініціалізовані значеннями підмножини гіперплощини, через яку проходять два головних власних вектора матриці коваріації вхідних значень навчальної вибірки.

При виборі способу початкової ініціалізації слід керуватися наступною інформацією:

- об'ємом навчальної вибірки;
- кількістю епох, відведених для навчання;
- розміром карти.

Між зазначеними параметрами і способом початкової ініціалізації існує багато залежностей. Виділимо кілька головних.

1. Якщо обсяг навчальної вибірки значно (в 100 і більше) перевищує число осередків карти і час навчання не грає першочерговим ролі, то краще вибрати ініціалізацію випадковими значеннями, так як це дасть меншу ймовірність попадання в локальний мінімум помилки кластеризації.

2. Якщо обсяг навчальної вибірки не дуже великий, час навчання обмежено або необхідно зменшити ймовірність появи після навчання порожніх клітинок, в які не потрапило жодного примірника навчальної вибірки, то слід використовувати ініціалізацію прикладами з навчальної множини.

3. Ініціалізацію з власних векторів можна використовувати при будь-якому збігу обставин. Єдине зауваження: ймовірність появи порожніх клітинок після навчання вище, ніж при ініціалізації прикладами з навчальної множини. Саме цей спосіб краще вибирати при першому ознайомленні з даними.

Швидкість навчання - задається швидкість навчання на початку та в кінці навчання мережі Кохонена. Рекомендовані значення: 0,1-0,3 на початку і 0,05-0,005 наприкінці навчання.

Радіус навчання - задається радіус навчання на початку та в кінці навчання мережі Кохонена. Радіус на початку повинен бути досить великий - приблизно половина або менше розміру карти (максимальне лінійне відстань від будь-якого нейрона до іншого будь-якого нейрона). а в кінці - досить малим, приблизно 1 або менше. Початковий радіус в Deductor підбирається автоматично залежно від розміру картки.

У цьому ж блоці задається Функція сусідства: Гауссова або Ступенева. Якщо функція сусідства Ступенева, то «сусідами» для нейрона-переможця будуть рахуватися всі нейрони, лінійне відстань до яких не більше поточного радіуса навчання. Якщо використовується Гауссова функція сусідства, то «сусідами» для нейрона-переможця будуть рахуватися всі нейрони карти, але різною мірою повноти. При використанні Гауссової функції сусідства навчання проходить більш плавно і рівномірно, так як одночасно змінюються ваги всіх нейронів, що може дати трохи кращий результат, ніж якби використовувалася ступінчаста функція. Однак час, необхідний на навчання, потрібно небагато більшу, з причини того, що на кожній епосі коригуються всі нейрони.

Кластеризація - у цій області вказуються параметри алгоритму k-means (G-means), який запускається після алгоритму Кохонена для угруповання осередків карти. Тут потрібно тільки визначити, дозволити алгоритму автоматично визначити число кластерів (G-means), або відразу зафіксувати його (k-means). Слід знати, що автоматично підбирається число кластерів не завжди приводить до бажаного результату - число кластерів може пропонуватися занадто великим, тому розраховувати на цю опцію можна тільки на етапі дослідження.

У наступному вікні, натиснувши кнопку **Пуск**, можна буде побачити динаміку процесу навчання мережі Кохонена. За замовчуванням алгоритм робить 500 ітерацій (епох). Якщо попередньо встановити прапор Рестарт, то ваги нейронів будуть проініціалізовані відповідно до обраного на попередньому кроці способу ініціалізації, інакше навчання розпочнеться з поточних вагових коефіцієнтів (це справедливо тільки при повторній налаштуванні вузла).

Аналіз одержаних результатів. Висновки і рекомендації.

В протоколі вказати тему та мету лабораторної роботи, навести коротко теоретичні відомості. Протокол оформити відповідно до методики виконання та зробити скріншоти відповідно до завдання.

У висновку вказати переваги та недоліки побудови нейронної мережі в середовищі Deductor.

Контрольні питання

1. Для чого застосовується Deductor?
2. Які задачі можливо вирішувати в середовищі Deductor?
3. З яких частин складається Deductor?
4. Як створити новий проект?
5. Як відбувається побудова нейронної мережі в середовищі Deductor?
6. Охарактеризуйте алгоритм навчання RProp?

Рекомендована література

1. Акіменко В.В. Проектування СППР на основі нечіткої логіки. Навчально-методичний посібник / В.В. Акіменко, Ю.В. Загородній. – К.: Вид-во КНУ, 2007. – 94с.
2. Гаврилова Т.А., Базы знаний интеллектуальных систем / Т.А. Гаврилова, В.Ф. Хорошевский. – СПб.: Питер, 2001. - 384 с.
3. Кишенько В.Д. Інтелектуальні системи [Текст]: конспект лекцій для студ. напряму 0925 "Автоматизація та комп'ютерно-інтегровані технології" ден. та заоч. форм навч./ В.Д. Кишенько – К.:НУХТ, 2008.- 133 с.
4. Куссуль Н.М. Інтелектуальні обчислення [Текст]: навч. посібник/ Н.М. Куссуль., А.Ю. Шелестов., А.М. Лавренюк. –К.: "Наукова думка", 2006.- 186 с.
5. Литвин В. В. Бази знань інтелектуальних систем підтримки прийняття рішень [Текст]/ В. В. Литвин. — Львів: Видавництво Львівської політехніки, 2011.- 240 с.
6. Леоненков А.В. Нечёткое моделирование в среде MATLAB. - СПб : БХВ-Петербург, 2003. - 736 с.
7. Хайкин С. Нейронные сети: полный курс [Текст]:/С. Хайкин. - 2-е издание. : Пер. с англ. – М. : Издательский дом «Вильямс», 2008.- 1104 с.