

МІНІСТЕРСТВО ОСВІТИ І НАУКИ,
МОЛОДІ ТА СПОРТУ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт з дисципліни
«Основи теорії інтелектуальних систем»
для студентів спеціальностей
8.05010201 «Комп'ютерні системи та мережі» та
8.05010203 «Спеціалізовані комп'ютерні системи»
усіх форм навчання

2012

Методичні вказівки до лабораторних робіт з дисципліни «Основи теорії інтелектуальних систем» для студентів спеціальностей 8.05010201 «Комп'ютерні системи та мережі» та 8.05010203 «Спеціалізовані комп'ютерні системи» усіх форм навчання. /Укл.: М.Ю. Тягунова, Д.В. Тягунов. – Запоріжжя: ЗНТУ, 2012. – 66 с.

Укладачі: М.Ю. Тягунова, к.т.н., доцент
Д.В. Тягунов, асистент

Рецензент: Р.К. Кудерметов, к.т.н., доцент

Відповідальний
за випуск М.Ю. Тягунова, к.т.н., доцент

Затверджено:
На засіданні кафедри
«Комп'ютерні системи та мережі»
Протокол № 4 від 28.12.2011 г.

ЗМІСТ

1 Лабораторная работа №1. Знайомство з інструментами моделювання нейронних мереж	4
2 Лабораторная работа №2. Застосування нейронних мереж Кохонена	41
3 Лабораторная работа №3. Застосування нейронних мереж Хопфілда	47
4 Лабораторная работа №4. Розпізнавання образів	52
5 Лабораторная работа №5. Застосування генетичних алгоритмів	55
6 Лабораторная работа №6. Експертні системи	62
Вимоги до оформлення письмових звітів	64
Література	65

1 ЛАБОРАТОРНА РОБОТА № 1

Знайомство з інструментами моделювання нейронних мереж

Мета роботи: вивчити основні інструменти моделювання нейронних мереж

1.1 Використання макета Matlab

Нейронні мережі (NN – Neural Networks) широко використовуються для вирішення різноманітних завдань. Серед областей, в яких розвивається застосування NN – обробка аналогових і цифрових сигналів, синтез та ідентифікація електронних ланцюгів і систем. Основи теорії та технології застосування NN широко представлені в пакеті MATLAB. У зв'язку з цим особливо слід відзначити версію пакету – MATLAB 6.0, де вперше представлений GUI (Graphical User Interface – графічний інтерфейс користувача) для NN – NNTool.

Прикладами застосування технології нейронних мереж для цифрової обробки сигналів є: фільтрація, оцінка параметрів, детектування, ідентифікація систем, розпізнавання образів, реконструкція сигналів, аналіз часових рядів і стиснення. Згадані види обробки застосовуються до різноманітних видів сигналів: звукових, відео, мовних, зображення, передачі повідомлень, геофізичних, локаційних, медичних вимірювань (кардіограми, енцефаллограми, пульс) та інших.

Обробка сигналів в технологіях NN виконується з допомогою NN без пам'яті, або NN с пам'яттю. І в тому і в іншому випадках ключовим елементом є NN без пам'яті. Подібна роль визначається тією обставиною, що при використанні нейронів з певними функціями активації (передавальними характеристиками) NN є універсальним апроксиматором. Останнє означає, що в заданому діапазоні зміни вхідних змінних NN може з заданою точністю відтворювати (моделювати) довільну безперервну функцію цих змінних.

Нижче обговорюються питання, пов'язані з так званими NN прямого поширення, тобто без зворотних зв'язків. Чудовою властивістю таких NN є їх стійкість.

Після того як структура NN обрана, повинні бути встановлені її параметри. Вибір структури NN і типів нейронів. Що ж стосується значень параметрів, то, як правило, вони визначаються в процесі розв'язання деякої оптимізаційної задачі. Ця процедура в теорії NN називається навчанням.

Графічний інтерфейс користувача NNTool дозволяє вибирати структури NN з великого переліку і надає безліч алгоритмів навчання для кожного типу мережі.

Далі розглянуті наступні питання, пов'язані з роботою з NNTool:

- призначення графічних керуючих елементів;
- підготовка даних;
- створення нейронної мережі;
- навчання мережі;
- прогін мережі.

1.1.1 Керуючі елементи NNTool

Щоб запустити NNTool, необхідно виконати однойменну команду в командному вікні MATLAB:

```
>>nntool
```

після цього з'явиться головне вікно NNTool, іменоване «вікном управління мережами і даними» (Network / Data Manager) (рис. 1.1).

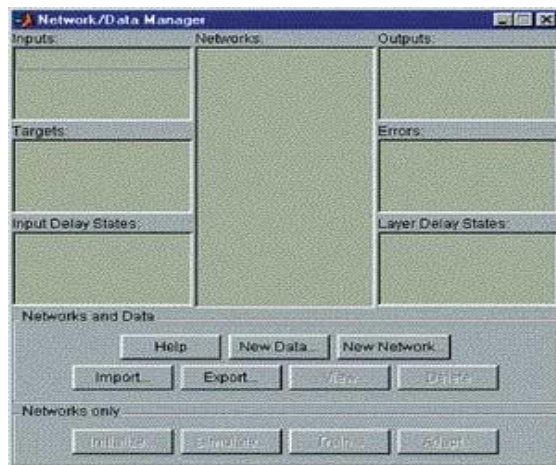


Рисунок 1.1 – Головне вікно NNTool

Панель «Мережі та дані» (Networks and Data) має функціональні клавіші з наступними призначеннями:

- Допомога (Help) – короткий опис керуючих елементів цього вікна;
- Нові дані (New Data ...) – виклик вікна, що дозволяє створювати нові набори даних;
- Нова мережа (New Network ...) – виклик вікна створення нової мережі;
- Імпорт (Import ...) – імпорт даних з робочого простору MATLAB в простір змінних NNTool;
- Експорт (Export ...) – експорт даних з простору змінних NNTool в робочий простір MATLAB;
- Вигляд (View) – графічне відображення архітектури вибраної мережі;
- Видалити (Delete) – видалення обраного об'єкта.

На панелі «Тільки мережі» (Networks only) розташовані клавіші для роботи виключно з мережами. При виборі покажчиком миші об'єкта будь-якого іншого типу, ці кнопки стають неактивними.

При роботі з NNTool важливо пам'ятати, що клавіші View, Delete, Initialize, Simulate, Train і Adapt (зображені на рис.1 як неактивні) діють стосовно того об'єкту, який відзначений у даний момент виділенням. Якщо такого об'єкта немає, або над виділеним об'єктом неможливо виробити вказане діяння, відповідна клавіша неактивна.

Розглянемо створення нейронної мережі за допомогою NNTool на прикладі.

Приклад 1. Хай потрібно створити нейронну мережу, що виконує логічну функцію «І».

Створення мережі. Виберемо мережу, що складається з одного персептрона з двома входами. У процесі навчання мережі на її входи подаються вхідні дані і проводиться співставлення значення, отриманого на виході, з цільовим (бажаним). На підставі результату порівняння (відхилення отриманого значення від бажаного) обчислюються величини зміни ваг і зсуву, що зменшують це відхилення.

Отже, перед створенням мережі необхідно заготовити набір навчальних і цільових даних. Складемо таблицю достовірності для

логічної функції «I», де P1 і P2 – входи, а Res – бажаний вихід (табл. 1.1).

Таблиця 1.1 – Таблиця достовірності логічної функції «I»

P1	P2	Res
0	0	0
0	1	0
1	0	0
1	1	1

Щоб задати матрицю, що складається з чотирьох векторів-рядків, як вхідні, скористаємося кнопкою New Data. У вікні слід провести зміни, показані на рис. 1.2, і натиснути клавішу «Створити» (Create).

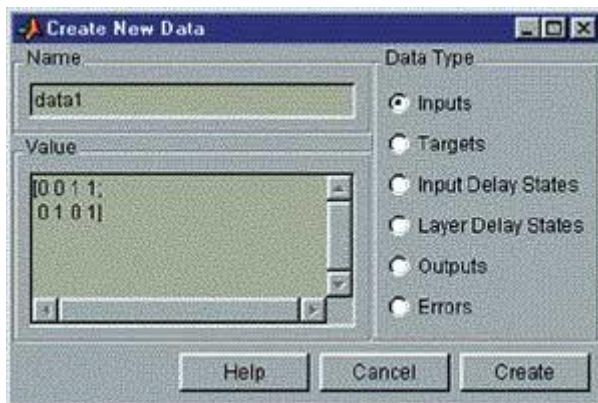


Рисунок 1.2 – Завдання вхідних векторів

Після цього у вікні управління з'явиться вектор data1 у розділі Inputs. Вектор цілей задається схожим чином (рис. 1.3).

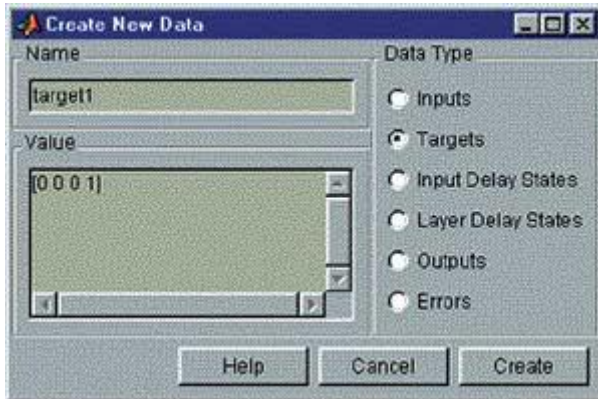


Рисунок 1.3 – Завдання цільового вектора

Після натискання на Create у розділі Targets з'явиться вектор target1.

Дані в полі «Значення» (Value) можуть бути представлені будь-яким зрозумілим MATLAB виразом. Приміром, попереднє визначення вектора цілей можна еквівалентно замінити рядком виду `bitand ([0 0 1 1], [0 1 0 1])`.

Тепер слід приступити до створення нейронної мережі. Вибираємо кнопку New Network і заповнюємо форму, як показано на рис. 1.4.

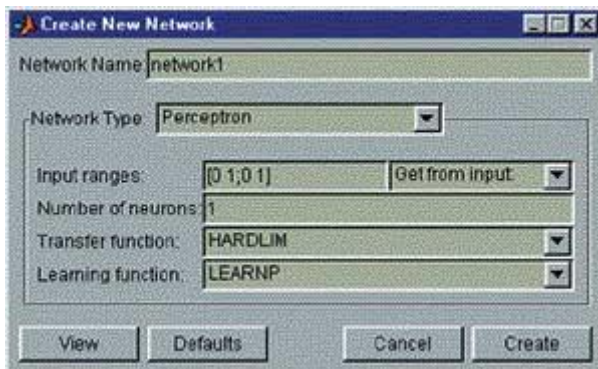


Рисунок 1.4 – Вікно «Створення мережі»

При цьому поля несуть наступні смислові навантаження:

- Ім'я мережі (Network Name) – це ім'я об'єкта створюваної мережі;
- Тип мережі (Network Type) – определяє тип мережі і в контексті обраного типу становить для введення різні параметри в частині вікна, розташованої нижче цього пункту. Таким чином, для різних типів мереж вікно змінює свій зміст;
- Вхідні діапазони (Input ranges) – матриця з числом рядків, що дорівнює числу входів мережі. Кожен рядок представляє собою вектор з двома елементами: перший – мінімальне значення сигналу, яке буде подано на відповідний вхід мережі при навчанні, другий – максимальне. Для спрощення введення цих значень передбачений список, що випадає «Отримати з входу» (Get from input), що дозволяє автоматично сформувати необхідні дані, вказавши ім'я вхідної змінної;
- Кількість нейронів (Number of neurons) – число нейронів у шарі;
- Передавальна функція (Transfer function) – в цьому пункті вибирається передаточна функція (функція активації) нейронів;
- Функція навчання (Learning function) – функція, що відповідає за оновлення ваг і зміщень мережі в процесі навчання.

За допомогою клавіші «Вид» (View) можна подивитися архітектуру створюваної мережі (рис. 1.5). Так, ми маємо можливість пересвідчитися, чи всі дії були проведені вірно. На рис. 5 зображена персептронна мережа з вихідним блоком, який реалізує передатну функцію з жорстким обмеженням. Кількість нейронів у шарі дорівнює одному, що символічно відображається розмірністю вектора-стовпця на виході шару і вказується числом безпосередньо під блоком передавальної функції. Розглянута мережа має два входи, бо розмірність вхідного вектора-стовпця дорівнює двом.

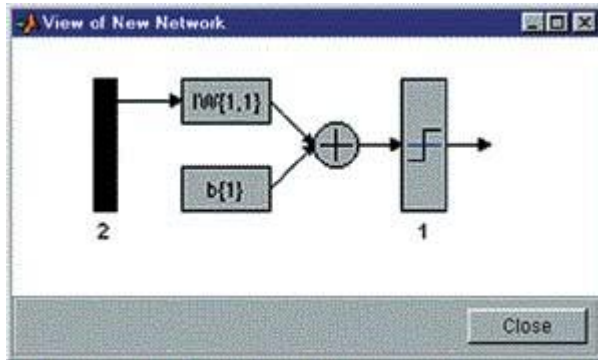


Рисунок 1.5 – Попередній перегляд створюваної мережі

Отже, структура мережі відповідає нашому завданню. Тепер можна закрити вікно попереднього перегляду, натиснувши клавішу «Закрити» (Close), і підтвердити намір створити мережу, натиснувши «Створити» (Create) у вікні створення мережі.

В результаті виконаних операцій в розділі «Мережі» (Networks) головного вікна NNTool з'явиться об'єкт з ім'ям network1.

Навчання. Наша мета – побудувати нейронну мережу, яка виконує функцію логічного «І». Очевидно, не можна розраховувати на те, що відразу після етапу створення мережі остання буде забезпечувати правильний результат (правильне співвідношення «вхід / вихід»). Для досягнення мети мережу необхідно належним чином навчити, тобто підібрати значення параметрів, які підходять. У MATLAB реалізовано більшість відомих алгоритмів навчання нейронних мереж, серед яких представлено два для персептронних мереж розглянутого виду. Створюючи мережу, ми вказали LEARNP як функції, що реалізує алгоритм навчання (рис. 1.4).

Повернімося до головного вікна NNTool. На даному етапі інтерес представляє нижня панель «Тільки мережі» (Networks only). Натиснення будь-який з клавіш на цій панелі викличе вікно, на безлічі вкладок якого представлені параметри мережі, необхідні для її навчання і прогону, а також ті, які відображають поточний стан мережі.

Відзначивши покажчиком миші об'єкт мережі network1, відкриємо вікно управління мережею натисканням кнопки Train. Перед нами постане вкладка «Train» вікна властивостей мережі, що

містить, у свою чергу, ще одну панель вкладок (рис. 1.6). Їх головне призначення – управління процесом навчання. На вкладці «Інформація навчання» (Training info) потрібно вказати набір навчальних даних у полі «Входи» (Inputs) і набір цільових даних у полі «Цілі» (Targets). Поля «Виходи» (Outputs) і «Помилки» (Errors) NNTool заповнює автоматично. При цьому результати навчання, до яких відносяться виходи і помилки, будуть зберігатися в змінних із зазначеними іменами.

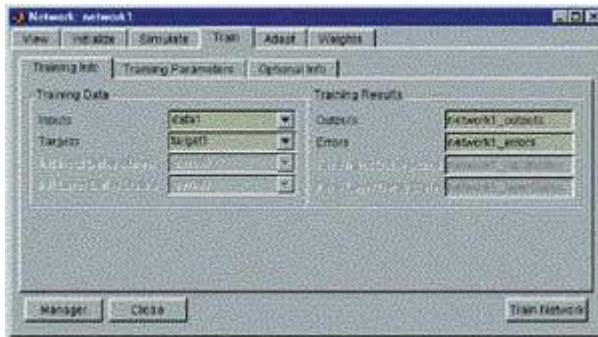


Рисунок 1.6 – Вікно параметрів мережі, відкрите на вкладці «навчання» (Train)

Завершити процес навчання можна, керуючись різними критеріями. Можливі ситуації, коли переважно зупинити навчання, вважаючи достатнім деякий інтервал часу. З іншого боку, об'єктивним критерієм є рівень помилки.

На вкладці «Параметри навчання» (Training parameters) для нашої мережі (рис.1.7). можна встановити наступні поля:

- Кількість епох (epochs) – визначає число епох (інтервал часу), після яких навчання буде припинено;
- Епохою називають одноразове подання всіх навчальних вхідних даних на входи мережі;
- Досягнення мети або попадання (goal) – тут задається абсолютна величина функції помилки, при якій мета буде вважатися досягнутою;
- Період оновлення (show) – період оновлення графіка кривої навчання, виражений числом епох;

- Час навчання (time) – після закінчення зазначеного тут тимчасового інтервалу, вираженого в секундах, навчання припиняється.

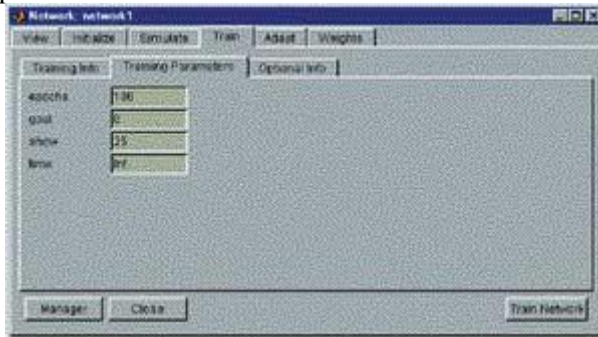


Рисунок 1.7 – Вкладка параметрів навчання

Беручи до уваги той факт, що для задач з лінійно віддільні множинами (а наша задача відноситься до цього класу) завжди існує точне рішення, встановимо поріг досягнення мети, рівний нулю. Значення інших параметрів залишимо за умовчанням. Зауважимо тільки, що поле часу навчання містить запис Inf, який визначає нескінченний інтервал часу (від англійського Infinite – нескінченний).

Наступна вкладка «Необов'язкова інформація» (Optional Info) показана на рис. 1.8.



Рисунок 1.8 – Вкладка необов'язкової інформації

Розглянемо вкладку навчання (Train). Щоб почати навчання, потрібно натиснути кнопку «Навчити мережу» (Train Network). Після

цього, якщо в поточний момент мережа не задовольняє ні одній з умов, зазначених у розділі параметрів навчання (Training Parameters), з'явиться вікно, що ілюструє динаміку цільової функції – криву навчання. У нашому випадку графік може виглядати так, як показано на рис. 1.9. Кнопкою «Зупинити навчання» (Stop Training) можна припинити цей процес. З рисунку видно, що навчання було зупинено, коли цільова функція досягла встановленої величини ($\text{goal} = 0$).

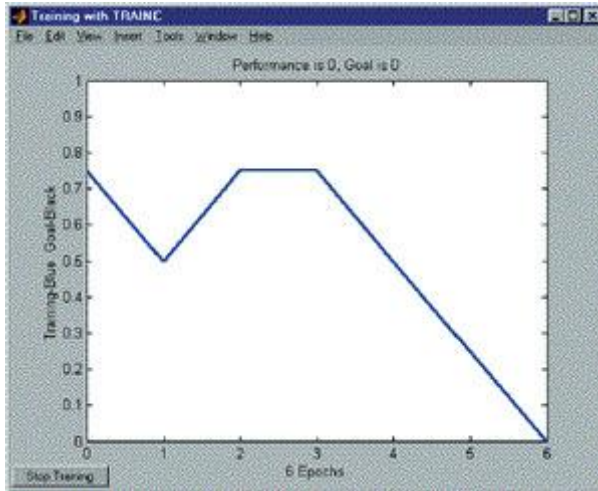


Рисунок 1.9 – Крива навчання

Слід зазначити, що для персептронів, що мають функцію активації з жорстким обмеженням, помилка розраховується як різниця між ціллю і одержаним виходом.

Отже, алгоритм навчання знайшов точне рішення задачі. У методичних цілях переконаємося в правильності рішення задачі шляхом прогону навченої мережі. Для цього необхідно відкрити вкладку «Прогін» (Simulate) і вибрати у випадяючому списку «Входи» (Inputs) заготовлені дані. У даній задачі природно використовувати той самий набір даних, що й при навчанні data1. При бажанні можна встановити прапорець «Задати ціль» (Supply Targets). Тоді в результаті прогону додатково будуть розраховані значення помилки. Натискання кнопки «Прогін мережі» (Simulate Network) запише результати прогону в змінну, ім'я якої вказано у полі «Виходи» (Outputs). Тепер

можна повернутися в основне вікно NNTool і, виділивши мишею вихідну змінну network1, натиснути кнопку «Перегляд» (View). Вміст вікна перегляду співпадає із значенням вектора цілей – мережа працює правильно.

Слід зауважити, що мережа створюється ініціалізованою, тобто значення ваг і зміщень задаються певним чином. Перед кожним наступним досвідом навчання звичайно початкові умови оновлюються, для чого на вкладці «Ініціалізація» (Initialize) передбачена функція ініціалізації. Так, якщо потрібно провести кілька незалежних дослідів навчання, ініціалізація ваг і зміщень перед кожним з них здійснюється натисканням кнопки «Ініціалізувати ваги» (Initialize Weights).

Повернемося до вкладки «Необов'язкова інформація» (Optional Info) (рис. 1.8). Щоб зрозуміти, якій цілі служать представлені тут параметри, необхідно обговорити два поняття: перенавчання та узагальнення.

При виборі нейронної мережі для вирішення конкретного завдання важко передбачити її порядок. Якщо обрати невиправдано великий порядок, мережа може виявитися занадто гнучкою і може представити просту залежність складним чином. Це явище називається перенавчанням. У разі мережі з недостатньою кількістю нейронів, необхідний рівень помилки ніколи не буде досягнутий. Тут у наявності надмірне узагальнення.

Для попередження перенавчання застосовується така техніка. Дані діляться на дві множини: навчальні (Training Data) та контрольні (Validation Data). Контрольна множина в навчанні не використовується. На початку роботи помилки мережі на навчальному і контрольному множинах будуть однаковими. У міру того, як мережа навчається, помилка навчання убиває, і, поки навчання зменшує дійсну функцію помилки, помилка на контрольній множині також буде спадати. Якщо ж контрольна помилка перестала зменшуватися або навіть стала зростати, це вказує на те, що навчання слід закінчити. Зупинка на цьому етапі називається ранньої зупинкою (Early stopping).

Таким чином, необхідно провести серію експериментів з різними мережами, перш ніж буде отримана та, яка підходить. При цьому щоб не бути введеним в оману локальними мінімумами функції помилки, слід кілька разів навчати кожну мережу.

Якщо в результаті послідовних кроків навчання і контролю помилка залишається неприпустимо великою, доцільно змінити модель нейронної мережі (наприклад, ускладнити мережу, збільшивши число нейронів, або використовувати мережу іншого виду). У такій ситуації рекомендується застосовувати ще одну множину – тестову множину спостережень (Test Data), яка представляє собою незалежну вибірку з вхідних даних. Підсумкова модель тестується на цій множині, що дає додаткову можливість переконатися в достовірності отриманих результатів. Очевидно, щоб зіграти свою роль, тестова множина повинна бути використана тільки один раз. Якщо її використати для коригування мережі, вона фактично перетвориться на контрольну множину.

Установка верхнього прапорця (рис.1.8) дозволить задати контрольну множину і відповідний вектор цілей (можливо, той же, що при навчанні). Встановлення нижнього дозволяє задати тестову множину і вектор цілей для неї.

Навчання мережі можна проводити в різних режимах. У зв'язку з цим, в NNTool передбачено дві вкладки, що представляють навчальні функції: розглянута раніше вкладка Train і «Адаптація» (Adapt). Adapt вміщує вкладку інформація адаптації (Adaption Info), на якій містяться поля, схожі за своїм призначенням з полями вкладки Training Info і виконують ті ж функції і вкладку параметри адаптації (Adaption Parameters). Остання містить єдине поле «Проходи» (passes). Значення, зазначена в цьому полі, визначає, скільки разів всі вхідні вектори будуть представлені мережі в процесі навчання.

Параметри вкладок «Train» і «Adapt» в MATLAB використовуються функціями train і adapt, відповідно. Детальна довідкова інформація з цих функцій наведена в [1].

1.1.2 Поділ лінійно-невіддільних множин

Розглянута задача синтезу логічного елемента «І» може трактуватися як завдання розпізнавання лінійно віддільних множин. На практиці ж частіше зустрічаються завдання поділу лінійно невіддільних множин, коли застосування персептрона з функцією активації з жорстким обмеженням не дасть рішення. У таких випадках слід використовувати інші функції активації.

Показовим прикладом лінійно невіддільною завдання є створення нейронної мережі, що виконує функцію «виключаюче або».

Приклад 2.

Хай потрібно створити нейронну мережу, що виконує логічну функцію «виключаюче або».

Розглянемо таблицю істинності цієї функції (табл. 1.2).

Таблиця 1.2 –Таблиця істинності логічної функції «виключаюче або»

P1	P2	Res
0	0	0
0	1	1
1	0	1
1	1	0

Що ж мається на увазі під «лінійною невіддільністю» множин? Щоб відповісти на це питання, надамо множини вихідних значень у просторі входів (рис. 1.10), за наступним правилом: поєднання входів P1 і P2, при яких вихід Res звертається в нуль, позначаються кружечком, а ті, при яких Res звертається в одиницю – хрестиком.

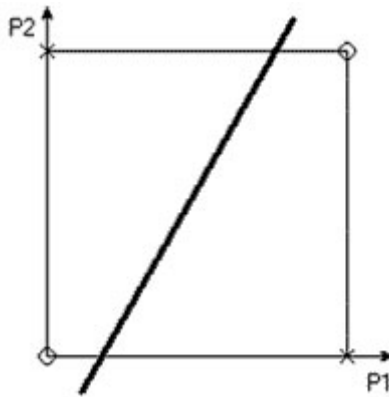


Рисунок 1.10 – Стан логічного елемента «виключаюче або»

Наша мета – провести межу, яка відокремлює множину нулів від множини хрестиків. З побудованої картини на рис. 10 видно, що неможливо провести пряму лінію, яка б відокремила нулі від одиниць. Саме в цьому сенсі множина нулів лінійно невіддільна від множини

одиниць, і персептрони, розглянуті раніше, в принципі, не можуть вирішити розглянуту задачу.

Якщо ж використовувати персептрони зі спеціальними нелінійними функціями активації, наприклад, сігмоїдними, то рішення задачі можливе.

Виберемо персептрон з двома нейронами прихованого шару, у яких функції активації сігмоїдні, і одним вихідним нейроном з лінійною функцією активації (рис. 1.11). В якості функції помилки вкажемо MSE (Mean Square Error – середній квадрат помилки). Нагадаємо, що функція помилки встановлюється у вікні «Створення мережі» після вибору типу мережі.

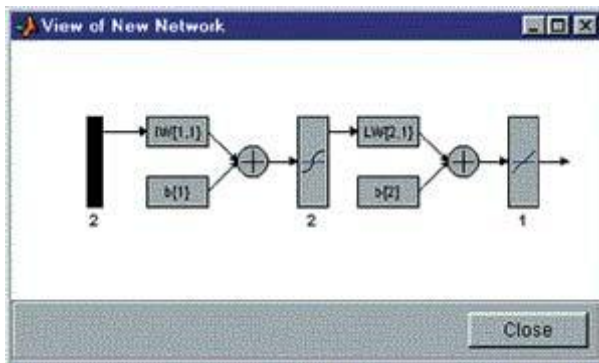


Рисунок 1.11 – Мережа для розв'язання задачі «виключаюче або»

Ініціалізуємо мережу, натиснувши кнопку Initialize Weights на вкладці Initialize, після чого навчимо, вказавши в якості входних значень сформовану раніше змінну data1, як цілі – новий вектор, відповідний бажаним виходам. У процесі навчання мережа не може забезпечити точного рішення, тобто звести помилку до нуля. Проте виходить наближення, яке можна спостерігати по кривій навчання на рис. 1.12.

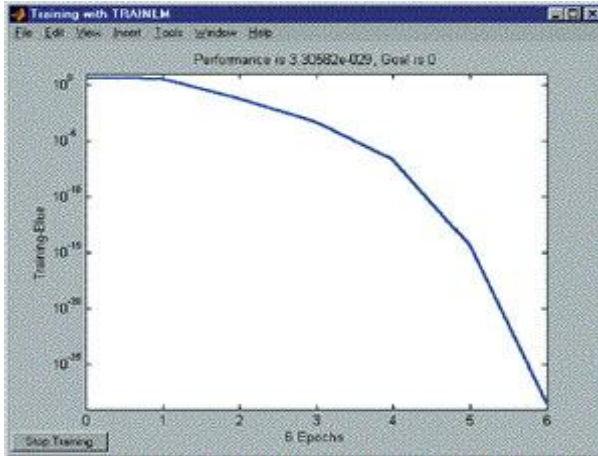


Рисунок 1.12 – Крива навчання в задачі «виключаюче або»

Слід зазначити, що дана крива може змінювати свою форму від експерименту до експерименту, але, у разі успішного навчання, характер функції буде монотонно убувати.

В результаті навчання, помилка була мінімізована до вельми малого значення, яке практично можна вважати рівним нулю.

Задача синтезу елемента «виключаюче або» є також прикладом задачі класифікації. Вона відображає спільний підхід до вирішення подібного роду задач.

1.1.3 Завдання апроксимації

Одним з найбільш чудових властивостей нейронних мереж є здатність апроксимувати і, більше того, бути універсальними апроксиматорами. Сказане означає, що за допомогою нейронних ланцюгів можна апроксимувати як завгодно точно безперервні функції багатьох змінних. Розглянемо приклад.

Приклад 3.

Необхідно виконати апроксимацію функції наступного вигляду

$$\sin\left(\frac{5px}{N} + \sin\left(\frac{7px}{N}\right)\right),$$

де $x \in \overline{1, N}$, а N – кількість точок функції.

Заготовимо цільові дані, ввівши в поле «Значення» (Value) вікна створення нових даних вираз:

$$\sin(5 * \pi * [1:100] / 100 + \sin(7 * \pi * [1:100] / 100)).$$

Ця крива є відрізком періодичного коливання з частотою $\frac{5p}{N}$, модульованого по фазі гармонійним коливанням з частотою $7N$ (рис. 1.15).

Тепер заготовимо набір навчальних даних (1, 2, 3, ..., 100), задавши їх наступним виразом: 1:100.

Виберемо перцептрон (Feed-Forward Back Propagation) з тринадцятьма сігмоїдними (TANSIG) нейронами прихованого шару і одним лінійним (PURELIN) нейроном вихідного шару. Навчання будемо виробляти, використовуючи алгоритм Левенберга-Маркардта (Levenberg-Marquardt), який реалізує функція TRAINLM. Функція помилки – MSE. Отримана мережа має вигляд, зображений на рис. 1.13.

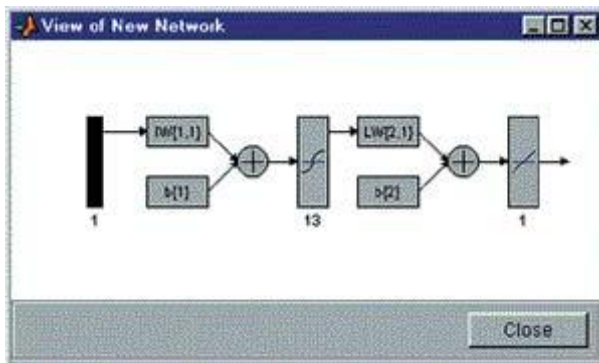


Рисунок 1.13 – Архітектура мережі для вирішення задачі апроксимації

Тепер можна приступити до навчання. Для цього необхідно вказати, які набори даних повинні бути використані в якості навчальних і цільових, а потім провести навчання (рис. 1.14).

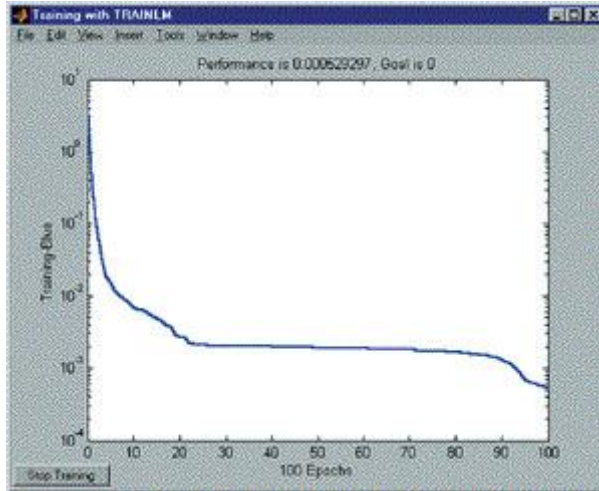


Рисунок 1.14 – Крива навчання в задачі апроксимації

Рис. 1.15 ілюструє різницю між цільовими даними та отриманої апроксимуючої кривої.

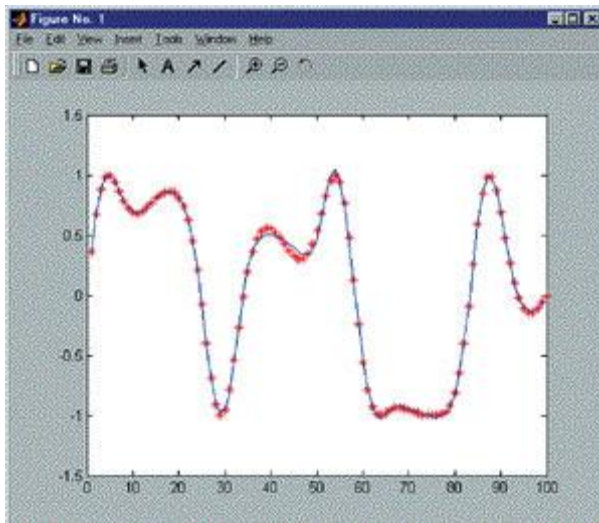


Рисунок 1.15 – Червона крива – цільові дані, синя крива – апроксимуюча функція

З рис. 1.14 і 1.15 видно, наскільки зменшилася помилка апроксимації за 100 епох навчання. Форма кривий навчання на останніх епохах свідчить також про те, що точність наближення може бути підвищена.

1.1.4 Імпорт-експорт даних

На практиці часто доводиться переносити дані з одного комп'ютера на інший або користуватися зовнішніми, по відношенню до NNTool, засобами обробки даних. У зв'язку з цим виникає необхідність збереження результатів роботи та завантаження даних. Не менш важливим є обмін даними між NNTool і MATLAB, оскільки простори їх змінних не перетинаються.

Ці завдання вирішують засоби імпорту-експорту та завантаження-збереження даних. Доступ до них здійснюється через головне вікно NNTool (рис. 1.1) за допомогою кнопок Import і Export.

Імпорт. Джерелом служить змінна в робочому просторі MATLAB, а пунктом призначення – змінна в робочому просторі NNTool. Натиснувши кнопку Import, потрапляємо у вікно «Імпорт – Завантаження даних» (Import or Load to Network / Data Manager). За замовчуванням тут встановлена опція завантаження з робочого простору MATLAB, тому в центрі вікна з'являється список, що належать йому змінних. Вибравши мишею потрібну змінну в полі «Вибір змінної» (Select a Variable), – їй можна задати довільне ім'я в полі «Ім'я» (Name), під яким вона буде скопійована в NNTool. Коли змінна виділена, NNTool аналізує її тип і робить доступними для вибору тих «Категорії даних» (Import As), які підтримуються. Вказавши одну з них, слід натиснути кнопку «Імпортувати» (Import), щоб завершити копіювання. Після того, як всі дії успішно проведені, ім'я імпортованої змінної з'явиться в одному зі списків головного вікна NNTool.

Завантаження з файлу. Тут джерело – файл. При цьому важливо, щоб його формат підтримувався NNTool. У відповідному форматі зберігаються так звані MAT-файли. Вони містять бінарні дані і дозволяють MATLAB зберігати змінні будь-яких підтримуваних типів і розмірностей. Такі файли можуть створюватися, наприклад, в процесі роботи з NNTool. Щоб завантажити змінні з MAT-файлу в NNTool, необхідно відкрити вікно імпорту, натиснувши кнопку «Import» головного вікна NNTool. Потім слід відзначити опцію

«Завантажити з файлу» (Load from disk file) і, натиснувши «Огляд» (Browse), відкрити файл з даними, що зберігаються у форматі MAT-файлів. Найчастіше, це файли з розширенням MAT. У результаті, список у вікні імпорту заповниться іменами змінних, збережених у вказаному MAT-файлі. Подальша послідовність дій повністю співпадає з описаною в пункті Імпорт (рис. 1.16).

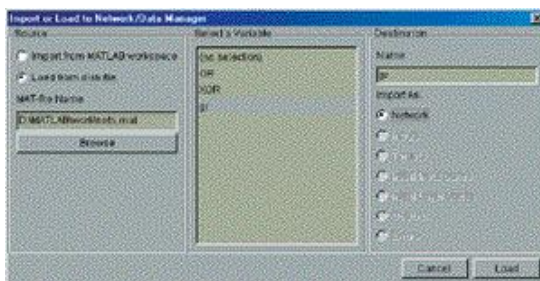


Рисунок 1.16 – Вікно імпорту та завантаження даних з MAT-файлу

Експорт. Ця функція копіює задані змінні з NNTool в робочий простір MATLAB. Вона доступна по натисненню кнопки Export головного вікна NNTool. У вікні експорту єдиним списком перераховані змінні всіх категорій, представлених у NNTool. Тут необхідно виділити ті з них, які підлягають експорту, і натиснути Export. Тепер виділені змінні скопійовані в робочий простір MATLAB.

Збереження у файлі. У цілому, процедура схожа з експортом, за одним винятком: відзначивши змінні, слід натиснути кнопку «Зберегти» (Save). Тоді з'явиться вікно, в якому можна задати ім'я файлу. Його можна вказати без розширення – за умовчанням NNTool прикріпить розширення MAT. Це пов'язано з тим, що NNTool зберігає дані тільки у форматі MAT-файлів (рис. 1.17).

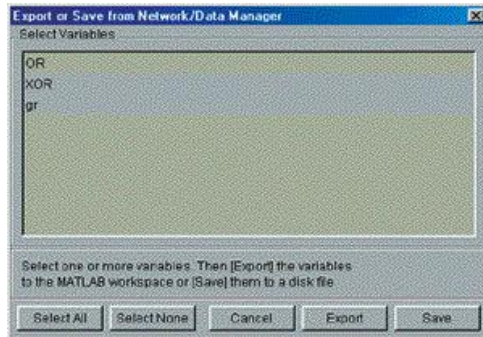


Рисунок 1.17 – Вікно експорту та збереження даних у MAT-файлі

Ми розглянули найпростіші задачі синтезу ланцюгів і обробки сигналів, для вирішення яких застосовувалися найбільш популярні нейронні мережі прямого поширення. Насправді, NNTool дозволяє вирішувати значно ширше коло завдань, надаючи можливість використовувати мережі різноманітних архітектур, з пам'яттю і без, із зворотними зв'язками і без таких. При цьому слід мати на увазі, що успіх багато в чому залежить від розуміння поведінки мереж, які конструюються, та їх апроксимаційних можливостей.

Завдання 1.

Виконати приклади 1-3 і показати їх викладачеві.

1.2 Використання пакету *STATISTICA Neural Networks*

Методи нейронних мереж отримують все більше поширення у дуже різних областях, від фундаментальних досліджень до "видобутку даних", прогнозування в бізнесі, управління ризиками, додатків в техніці і багатьох інших. Система *STATISTICA Neural Networks* – найбільш сучасна за технологією і краща за робочими характеристиками серед нейроно-мережових пакетів, представлених зараз на ринку, програмного забезпечення. Серед її переваг – такі що не мають аналогів в інших системах можливості, як *Автоматичний конструктор мережі* (заснована на методах штучного інтелекту система побудови архітектури мережі) і *Нейро-генетичний відбір даних* (засіб виявлення найбільш значущих змінних).

STATISTICA Neural Networks є єдиним в світі програмним продуктом для проведення нейромережових досліджень, повністю

перекладеним на російську мову. Це означає, що весь інтерфейс (десятки діалогових вікон і сценаріїв дослідження) та довідкова система STATISTICA Neural Networks переведені на російську мову і доступні користувачеві в єдиному середовищі.

Розглянемо основні можливості пакету STATISTICA Neural Networks 6.1.

1.2.1 Проведення класифікації

Вирішення задач класифікації - одна з найбільш важливих областей застосування нейронних мереж. У таких завданнях вхідні дані являють собою результати вимірювань деяких характеристик об'єкта. Мета полягає в тому, щоб визначити, до якого з декількох заданих класів належить цей об'єкт. Зазвичай класів буває рівно два (або один, і спостереження може або належати, або не належати йому). Задача «виключаюче або» - приклад завдання класифікації з двома класами. Спостереження може належати або не належати класу Хог.

Розглянемо роботу з пакетом STATISTICA Neural Networks на прикладі задачі «виключаюче або».

У пакеті ST Neural Networks навчальні дані зберігаються у вигляді набору даних, що містить деяку кількість спостережень, для кожного з яких задані значення декількох вхідних та вихідних змінних. Всі дані знаходяться у форматі системи STATISTICA, тому їх легко можна імпортувати із STATISTICA або, навпаки, перетворити на формат даних STATISTICA, користуючись різноманітними можливостями системи введення / виводу. Проте новий набір даних можна створити прямо в пакеті STATISTICA Neural Networks. Для цього потрібно виконати наступні дії.

1. Створити нову таблицю даних за допомогою меню *Файл - Создать - Таблица*.

2. У діалоговому вікні створення нового документа (рис. 1.18) ввести значення змінних майбутнього набору даних і спостережень.

3. Натиснути кнопку *OK*.

У цьому ж вікні можна задати тип даних, формат їх відображення та спосіб створення таблиці даних.

Кожному спостереженню відповідає один рядок таблиці. У початковий момент всі елементи таблиці будуть порожні.

Данные: Таблица (3v * 4с)

	1 First	2 Second	3 Xor	
1	1	0	1	
2	0	0	0	
3	1	1	0	
4	0	1	1	

Рисунок 1.18 – Вікно створення нового документа

1.2.1.1 Створення нової мережі

Нова мережа створюється засобами стартової панелі ST Neural Networks *Нейронные сети*. Щоб відкрити її, потрібно зайти в меню *Анализ* стандартної панелі інструментів і обрати *Нейронные сети*.

Стартова панель, представлена на рис. 1.19, містить три вкладки (*Быстрый*, *Дополнительно* та *Сети/Ансамбли*).

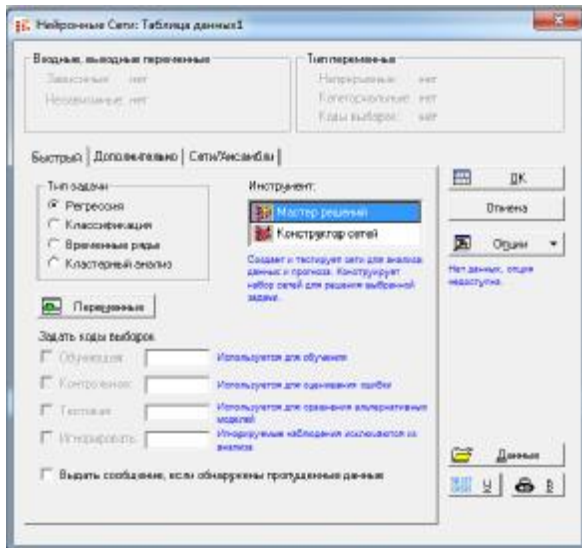


Рисунок 1.19 – Стартова панель

Щоб створити мережу для задачі XOR, необхідно:

1. На вкладці *Быстрый* встановити *Тип задачи* як *Классификация*.

2. Натиснути кнопку *Переменные*, щоб вивести стандартний діалог вибору змінних. Це вікно автоматично пропонує вибір змінних, відповідних вказаному типу задачі.

3. Вибрати змінну *Xor* як *Категориальную выходную переменную*, а змінні *First* і *Second* як *Непрерывные входные переменные*. Натиснути *ОК* у вікні вибору змінних, щоб повернутися в стартову панель, на вкладку *Быстрый*.

4. Як інструмент обрати *Конструктор сетей*.

5. Натиснути *ОК* на стартовій панелі, щоб вивести діалогове вікно *Конструктора сетей*.

6. Обрати тип мережі зі списку на вкладці *Быстрый*. Для задачі «виключаюче або» потрібен тип *Многослойный персептрон* (МП), який завжди пропонується за умовчанням.

7. На вкладці *Элементы* задати число прихованих шарів і кількість елементів в кожному шарі (в даному випадку потрібен один шар, що складається з двох елементів).

8. Натиснути кнопку *Правка*, щоб вивести вікно *Редактор сетей*, в якому можна переглянути усі параметри мережевої архітектури.

9. Ввести необхідні виправлення згідно п. 1.2.1.2 Задання режиму пре-і постпроцесування параметрів мережі.

10. Натиснути кнопку *ОК*, для виведення *Результаты* і потім знову натисніть *ОК*, щоб повернутися на стартову панель, в якій можна зберегти побудовану нову мережу.

1.2.1.2 Задання режиму пре - і постпроцесування параметрів мережі

Внести необхідні виправлення параметрів мережі можна в *Редакторе сетей*, у вікні якого міститься вісім вкладок, що дозволяють переглядати та/або редагувати по суті всі параметри створення нейронної мережі.

1. Для редагування вхідних і вихідних змінних виберіть вкладку *Переменные*. На ній змінні виводяться в окремих списках, які показують параметри кожної змінної. У даному випадку має бути дві вхідних і одна вихідна змінна.

2. На цій же вкладці можна змінити, якщо буде потрібно, перетворюючу функцію для пре- і постпроцесування. В даному випадку цілком підходить функція *Минимакс*.

3. На вкладці *Дополнительно* можна задати функції нормалізації нейрона, які забезпечують деякі подальші можливості. Вони дозволяють STATISTICA Neural Networks моделювати рідкісні типи мереж, які можуть бути цікаві теоретикам нейронних мереж. Однак, як правило, вони не потрібні, і за замовчуванням обидві функції встановлені на значення *Нет*. Ці функції нормалізації нейрона діють на повний вектор входів або виходів, а не на вхідну або вихідну змінну.

4. Для завдання «виключаюче або» потрібна мережа з трьома шарами: вхідним шаром з двох елементів, проміжним шаром з двох елементів і вихідним шаром з одного елемента. Це можна переглянути на вкладці *Слои*.

Переглянути архітектуру побудованої мережі можна, натиснувши на кнопку *Архитектура сети* в діалоговому вікні *Результаты* на вкладці *Дополнительно* (рис.1.20).

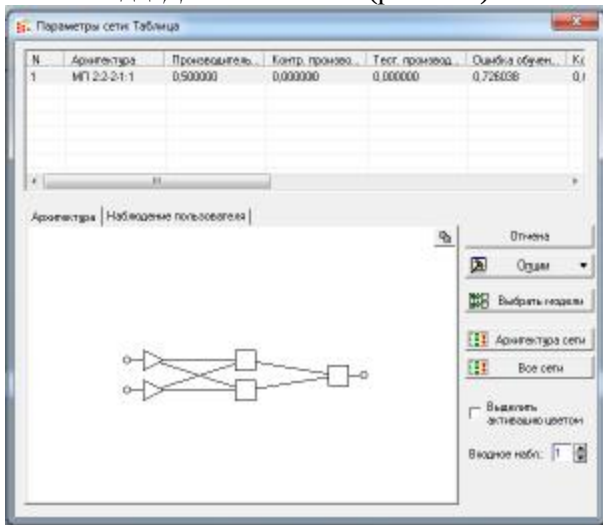


Рисунок 1.20 – Архітектура мережі

1.2.1.3 Збереження даних

Для збереження мереж використовується вкладка *Сети/Ансамбли* стартової панелі *Нейронные сети*. Дані зберігаються засобами стандартної панелі інструментів *Файл-Сохранить как ...*.

В якості стандартного розширення імені файлу мережі використовуються *.snp*.

Під час сеансу роботи має сенс періодично зберігати набір даних; функція *Сохранить* меню *Файл* зберігає поточний стан файлу даних, не вимагаючи повторного введення імені файлу. Те ж саме можна зробити за допомогою кнопки *Сохранить* на стандартній панелі інструментів.

Збережіть створену мережу з ім'ям *hog.snp*.

1.2.1.4 Навчання

Наступний крок після завдання набору даних та побудови відповідної мережі - це навчання.

У пакеті *ST Neural Networks* реалізовані основні алгоритми навчання багатoshарових персептронів: методи зворотного поширення, пов'язаних градієнтів і Левенберга-Маркара.

Щоб навчити мережу в задачі «виключаюче або», стежачи при цьому за роботою алгоритму, необхідно діяти так, як описано далі.

Навчання методом зворотного поширення:

1. Відкрийте файл збереженої мережі за допомогою вкладки *Сети/Ансамбли* стартової панелі інструментів *Нейронные сети*.
2. Виконайте дії, описані в пункті «Створення мережі», починаючи з 1-го по 7-й пункт.
3. Встановивши необхідні параметри в діалоговому вікні *Конструктор сетей*, натисніть *ОК*.

Натискання кнопки *ОК* у вікні *Конструктор сетей* виводить діалоги для навчання відповідно до типу мережі. Для багатoshарового персептрону в задачі класифікації буде показаний наступний діалог (рис. 1.21)

Для застосування на двох послідовних етапах навчання багатoshарових персептронів доступні шість видів алгоритмів. Виберіть для першого етапу *Обратное распространение*. У нашому прикладі нам не знадобиться другий етап навчання, тому слід прибрати прапорець навпроти.

Опція *Интерактивное обучение (отображать график)* включається на вкладці *Интерактивный*. Після встановлення всіх необхідних параметрів і натискання кнопки *ОК* з'явиться діалог, що відображає графік навчання та/або значення помилки для заданої вибірки.

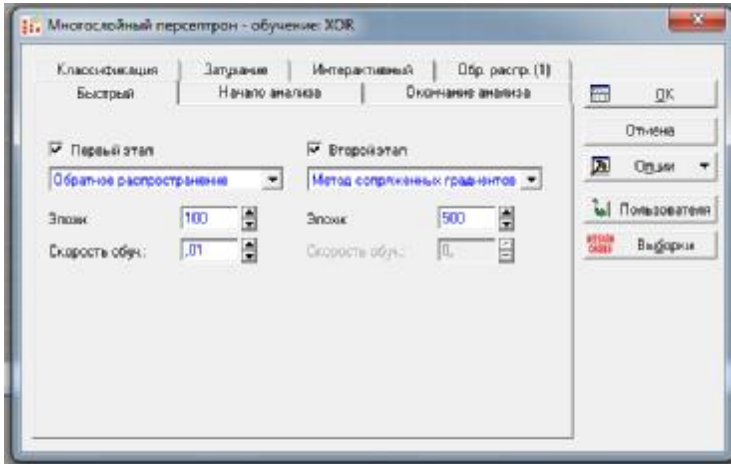


Рисунок 1.21 – Діалог для навчання багатослового персептрону

Цей графік оновлюється в режимі реального часу. Ви можете перервати навчання в будь-який момент (закінчивши навчання і зберігши нейронну мережу або скасувавши навчання), а також збільшити період навчання на кілька епох. Крім цього, ви можете визначити заголовок графіка, вставити легенду і мітки ліній на графік результатів.

У нижньому правому куті діалогу відображається рядок станів і час роботи. Також відображається відсоток зайнятості центрального процесора.

Не дивуйтеся, якщо спочатку помилка не буде істотно зменшуватися. Хоча задача «виключаюче або» виглядає зовсім простою, багатословому персептрону вирішити її набагато складніше, ніж багато інших реальних задач, які здаються дуже складними. При тих параметрах за замовчуванням, які запропонувала нам програма ST Neural Networks, може знадобитися близько тисячі ітерацій, перш ніж помилка стане близька до нуля.

1.2.1.5 Оптимізація навчання

Режим роботи алгоритму зворотного поширення залежить від ряду параметрів. Більшість з них зібрано в діалоговому вікні *Многослойный персептрон — обучение* на вкладці *Обратное распространение/Обр.распр.(1)*. Для багатьох практичних задач хорошим початковим наближенням будуть значення, що приймаються

за умовчанням, але при необхідності їх можна змінити. У випадку ж з завданням «виключаюче або» ці налаштування надаються дуже невдалими.

Опишемо коротко найбільш важливі параметри і виберемо їх значення для завдання «виключаюче або»:

1. *Епохи*. Визначає число епох навчання. Значення за замовчуванням 100 цілком прийнятно;

2. *Скорість обучения*. При збільшенні швидкості навчання алгоритм працює швидше, але в деяких завданнях це може привести до нестійкості (особливо якщо дані зашумлені). Для завдання «виключаюче або» підходить відносно висока швидкість навчання, наприклад 0,9;

3. *Момент*. Цей параметр покращує (прискорює) навчання в ситуаціях, коли помилка мало змінюється, а також надає алгоритму додаткову стійкість. Значення цього параметра завжди має лежати в інтервалі $[0; 1)$ (тобто > 0 і < 1). Часто рекомендується використовувати високу швидкість навчання у поєднанні з невеликим коефіцієнтом інерції і навпаки. Однак, в задачі «виключаюче або» обидва параметра потрібно взяти великими, наприклад 0,9.

4. *Перемешивать наблюдения*. При використанні цієї функції порядок, в якому спостереження подаються на вхід мережі, змінюється в кожній новій епосі. Це додає до навчання деякий шум, в наслідок якого помилка може відчувати невеликі коливання. Однак при цьому менше ймовірність того, що алгоритм «застрягне», і загальні показники його роботи зазвичай поліпшуються.

При таких налаштуваннях параметрів пакет ST Neural Networks вирішує завдання «виключаюче або» приблизно за 200 ітерацій.

У пакеті ST Neural Networks є можливість змінювати швидкість навчання та/або коефіцієнт інерції від епохи до епохи, поступово зрушуючи їх від початкових значень, заданих в полях у лівій частині діалогового вікна *Обратное распространение/Обр.распр.(1)*, до їх кінцевих значень, заданим в правій частині вікна. Наприклад, можна зменшувати швидкість по ходу навчання. При заданні початкових значень кінцеві значення за умовчанням встановлюються такими ж.

1.2.1.6 Виконання повторних прогонів

Якщо ви хочете порівняти результати роботи алгоритму в різних варіантах, скористайтесь кнопкою *Дополнительно* вікна *Обучение*. У результаті ваги мережі знову будуть встановлені випадковим чином

для початку наступного сеансу навчання. Якщо тепер після кнопки *Дополнительно* натиснути кнопку *Обучить*, на графіку почне малюватися нова лінія.

1.2.1.7 Помилки для окремих спостережень

У вікні *График обучения* виводиться сумарна помилка мережі. Але іноді буває корисно простежити за тим, як алгоритм навчання сприймає окремі спостереження.

У пакеті ST Neural Networks це робиться у вікні *Предсказанные*. У наведеній таблиці результатів можна порівнювати *Предсказанные* і *Наблюдаемые* значення залежної змінної.

Опція *Интерактивное обучение* дозволяє стежити за тим, як помилки змінюються в процесі навчання. Дану опцію необхідно активізувати перед запуском алгоритму навчання. Зробивши це, ви зможете спостерігати, як алгоритм намагається шукати компроміс між навчальними і заважають спостереженнями.

1.2.1.8 Запуск нейронної мережі

Після того, як мережа навчена, її можна запустити на виконання. У пакеті ST Neural Networks це можна зробити в декількох варіантах:

- на поточному наборі даних - в цілому чи на окремих спостереженнях;
- на іншому наборі даних - в цілому чи на окремих спостереженнях (такий набір даних уже може не містити вихідних значень і призначатися виключно для тестування);
- на одному конкретному спостереженні, для якого значення змінних введені користувачем, а не взяті з якогось файлу даних;
- з іншої програми за допомогою інтерфейсу програмування SNN API.
- запускаючи попередньо згенерований код на якій-небудь мові програмування (в генераторі коду підтримуються наступні мови програмування: C/C++, STATISTICA Visual Basic (SVB), Predictive Model Markup (PMML)).

Розглянемо деякі з варіантів.

Обробка спостережень по одному. Для обробки окремих спостережень з набору даних необхідно у вікні *Результаты* перейти на вкладку *Дополнительно*. Потім обираємо *Наблюдения пользователя*. Отже, ми потрапляємо у вікно *Прогноз для наблюдения пользователя* (рис. 1.22).

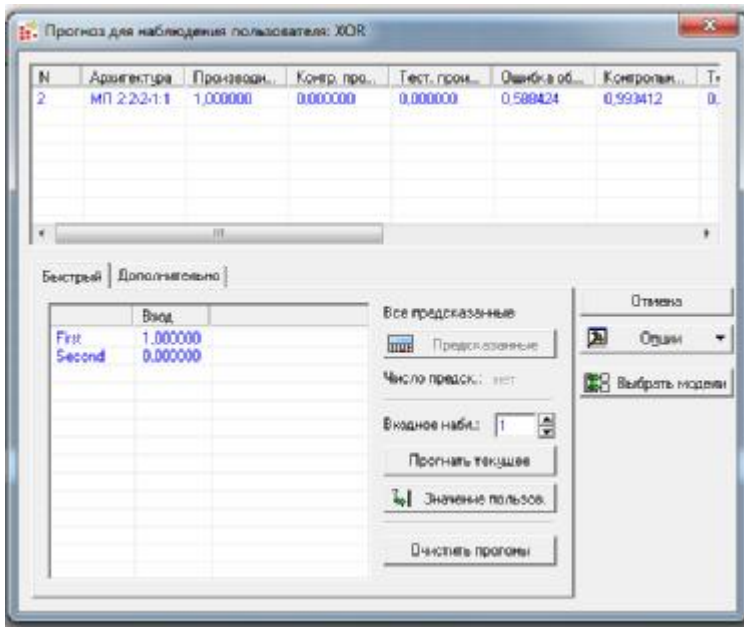


Рисунок 1.22 – Прогноз для спостереження користувача

У полі *Входное наблюдение* задається номер спостереження, що підлягає обробці. Щоб обробити поточне спостереження, натисніть кнопку *Прогнать текущее*, а для обробки будь-якого іншого спостереження введіть відповідний номер в поле *Входное наблюдение* і натисніть клавішу *Enter*. Щоб переглянути результати прогонів натисніть кнопку *Предсказанные*.

Найпростіший спосіб роботи з цим вікном такий: прорахувати перший тестовий приклад, натиснувши кнопку *Прогнать текущее*, а потім, натискаючи верхню стрілку на кнопці мікропрокрутки, яка розташована праворуч від поля *Входное наблюдение*, послідовно обробляти інші спостереження.

Значення вхідних змінних для поточного спостереження відображаються в таблиці, розташованій у вікні.

Крім фактичного вихідного значення, яке видає мережу, можна вивести також цільове значення. Якщо для навчання мережі

використовувався цей же набір даних, то фактичне значення має бути близьке до цільового.

У пакеті ST Neural Networks передбачені різні формати виводу. Різні опції знаходяться на вкладці *Дополнительно*.

Прогін всього набору даних. Для тестування мережі на всьому наборі даних служить інструмент *Запустить модель*, доступ до якого здійснюється через стартове вікно *Нейронных сетей*. Якщо потрібно прогнати спостереження з активної таблиці, то можна безпосередньо запускати вказаний інструмент. Якщо потрібно запустити мережу на нових даних, то спочатку необхідно вказати нове джерело даних та вхідні змінні.

В результаті роботи інструменту ми переходимо до стандартного вікна *Результаты*. При цьому статистики для мереж мають значення, які вони приймали для навчального набору даних. Всі інші результати відносяться вже до нових даних.

Тестування на окремому спостереженні. Іноді необхідно протестувати мережу на окремому спостереженні, яке не належить ні до якого набору даних. Причини для цього можуть бути такі:

- навчена мережа використовується для побудови прогнозів на нових даних з невідомими вихідними значеннями;
- ви хочете поекспериментувати з мережею, наприклад, перевірити чутливість результату до малих змін у даних.

Тестування заданих користувачем спостережень проводиться з вікна *Прогноз* для спостереження користувача, доступ до якого здійснюється через вкладку *Дополнительно* - *Наблюдение пользователя* меню *Результаты*.

Для цього потрібно натиснути кнопку *Значение пользователя* н ввести в з'явилася таблицю необхідні значення вхідних змінних.

Натискаємо *ОК*, а у вікні *Прогноз для наблюдения пользователя* - *Прогнать текущее*.

1.2.2 Робота з номінальними змінними

У пакеті ST Neural Networks можна працювати з так званими номінальними змінними (або атрибутами), тобто зі змінними, які можуть приймати кінцеве число значень, представлених у вигляді рядків тексту. Найпростіший приклад - змінна *Стать* = {Чол, Жен}, це номінальна змінна з двома можливими значеннями (станами). У задачі

«виключаюче або» вихідна змінна саме повинна бути номінальною змінною з двома станами: $Xor = \{False, True\}$.

В описуваній програмі номінальними можуть бути як вхідні, так і вихідні змінні, і є багато способів перетворення нечислової інформації, що міститься в них, до вигляду, зрозумілому нейронній мережі, і навпаки, способів інтерпретувати числовий вихід мережі як номінальну змінну. Підтримка номінальних змінних - органічна частина системи пре-і постпроцесування пакета ST Neural Networks.

Проілюструємо сказане, змінивши приклад *Xor* таким чином, щоб виходом була номінальна змінна.

Завдання *Xor* з номінальною вихідною змінною. Почнемо з того, що заново визначимо набір даних. Використаємо команду *Создать - Таблица*. Вказуємо число змінних рівним 3, число спостережень - 4.

У таблиці, задамо імена змінних *First*, *Second* і *Xor* (двічі клацаючи на заголовках стовпців).

Змінна *Xor* - номінальна, вона приймає два значення: *False* і *True*.

Введіть відповідні дані спостережень. Вводити значення номінальної змінної можна різними способами: безпосередньо надрукувати слова (*True* або *False*), ввести деякі кодові значення, а потім в редакторі текстових міток задати відповідні текстові значення.

Після того, як таблиця даних буде створена, створіть мережу за допомогою модуля *Нейронные сети*. Оберемо тип завдання - *Классификация*, задамо змінні та оберемо інструмент *Конструктор сетей*. У наступному вікні виберемо тип мережі - *Многослойный перцептрон*. Задайте число прихованих елементів рівним двом. Поповніть мережу за допомогою алгоритму зворотного поширення (при цьому всі наявні спостереження будемо використовувати як навчальну вибірку).

Зауваження. Функція перетворення для вихідної змінної буде змінена з *Минимум* на *Биполярная*: програма ST Neural Networks автоматично визначила, що це двозначна номінальна вихідна змінна, і відповідним чином змінила режим препроцесування. Такий спосіб - звичайний при вирішенні нейронною мережею задачі двозначної класифікації (тобто з двома класами): двозначній змінній відповідає один вихідний елемент, який видаватиме значення 1 для одного з класів і 0 - для іншого. Всі проміжні значення виражають ту чи іншу

ступінь впевненості в тому, що спостереження належить даному класу.

На практиці стандартний багат шаровий персептрон на виході ніколи не видасть точно 0 або 1, а лише, можливо, близькі до них значення. Таким чином, природно буде встановити довірчі рівні або межі. Наприклад, довірчі рівні можна задати так, щоб точки вище 0,95 оголошувалися такими, що належать до одного класу, а точки нижче 0,05 - до іншого.

1.2.2.1 Установка довірчих рівнів

Довірчі межі для класифікації встановлюються в пакеті ST Neural Networks в редакторі мережі на вкладці *Пороги*, яке відкривається командою *Правка*. Інший спосіб доступу до цього параметру - в діалозі *Обучение* на вкладці *Классификация*. У полі *Принять* задається мінімальне значення виходу, при якому спостереження буде вважатися, що належить до «позитивного» класу; поле *Отвергнуть* містить максимальне значення виходу, при якому спостереження буде віднесено до «негативного» класу.

1.2.2.2 Інтерпретація результатів класифікації

При прогоні мережі з вікна *Прогноз для наблюдения пользователя* або інструментом *Запустить модель* програма ST Neural Networks виробляє постпроцесування вихідних значень з урахуванням довірчих меж та присвоює значення вихідної змінної за такими правилами:

- якщо вихід перевищив поріг прийняття, то спостереження класифікується позитивно і видається друге з двох номінальних значень (в нашому прикладі *True*);

- якщо вихід був нижче порога відкидання, спостереження класифікується негативно і видається перше з двох номінальних значень (в нашому прикладі *False*);

- якщо вихід лежить між двома порогамі, класифікація оголошується невизначеною і вихідна змінна набуває «пропущене» значення (відображається пропуском в таблиці передбачених значень).

За умовчанням в пакеті ST Neural Networks видаються саме ці номінальні значення вихідних змінних. Однак, цілком імовірно, що ви захочете дізнатися справжні рівні довіри, які мережа привласнила спостереженнями до постпроцесування (тобто рівні активації вихідного елемента). Встановіть прапорець у опції *Уровни доверия* у

вікні *Результаты - Предсказанные* і в таблиці будуть показані рівні активації вихідного елемента мережі.

1.2.2.3 Статистики класифікації

Таблиця із статистиками, що відображають результати і якість класифікації, виводиться натисканням кнопки *Описательные статистики* у вікні *Результаты - Быстрый* або *Результаты - Описательные*.

Описові статистики розбиті на дві частини: сумарні статистики та статистики присвоювання.

Підсумкові статистики: *Всего. Загальне число спостережень в цьому наборі даних*

1. *Правильно.* Кількість спостережень з цього класу, які мережа класифікувала правильно.

2. *Помилково.* Кількість спостережень з цього класу, які мережа неправильно класифікувала як відносяться до другого класу.

3. *Невідомо.* Кількість спостережень з цього класу, які мережа не змогла класифікувати.

1.2.2.4 Матриця помилок

Тут показано, скільки спостережень з кожного класу було віднесено до кожного з можливих класів, включаючи правильне. Неясні спостереження не відображені. Ця інформація дозволяє більш докладно з'ясувати причини неправильних класифікацій.

Зауваження. Статистики класифікації автоматично оновлюються після запуску мережі і для їх перерахування не потрібно ніяких спеціальних дій.

1.2.3 Радіальні базисні функції

У пакеті ST Neural Networks є можливість працювати з мережами різних архітектур, з яких, ймовірно, найкраще відомі багат шарові персептрони. Інший, теж часто використовуваний тип елементів - Радіальна базисна функція (РБФ) (Haykin, 1994; Bishop, 1995).

Кожен елемент багат шарового персептрона виконує лінійне перетворення вхідного вектору (набору значень, що подається на вхід елемента); більш точно - береться зважена сума входів, і з неї вираховується порогове значення. У пакеті ST Neural Networks таке перетворення називається лінійною PSP (постсинаптичною потенційною) функцією. Результат потім пропускається через

нелінійну функцію активації. Лінійність PSP-функції означає, що багатошаровий персептрон фактично розділяє простір вхідних векторів на частини за допомогою гіперплощин (у двовимірному просторі гіперплощина - це звичайна пряма).

На відміну від багатошарового персептрона з його лінійним підходом, у мережі на радіальних базисних функціях використовується радіальна PSP-функція. У кожному елементі обчислюється квадрат відстані від вхідного вектора до вектора ваг. Потім ця відстань множиться на «порогове значення» (насправді, на величину відхилення), після чого результат пропускається через функцію активації. Таким чином, мережа на радіальних базисних функціях ділить простір входів за допомогою гіперсфери (у двовимірному просторі гіперсфери - це коло).

Кожен з двох описаних підходів має свої переваги і недоліки. Дія радіальних функцій дуже локальна, в той час як при лінійному підході охоплюється весь простір входів. Тому, як правило, РБФ-мережі мають більше елементів, ніж БЛП-мережі, однак БЛП може робити необґрунтовані узагальнення в ситуаціях, коли йому попадається набір даних, несхожий ні на які набори з навчальної множини, в той час як РБФ в такому випадку завжди буде видавати майже нульовий відгук.

У теорії наголошується також, що для вирішення деяких класів задач за допомогою БЛП потрібно два проміжних шарів, а іноді для більшої ефективності доводиться брати навіть більше число проміжних шарів. На відміну від цього, для РБФ-мережі завжди достатньо одного проміжного шару.

РБФ-мережа завжди складається з трьох шарів: вхідного, проміжного, який складається з радіальних елементів, і лінійного вихідного шару. Завдяки нелінійності радіальних елементів цей вихідний шар можна брати вже чисто лінійним (тобто з лінійною PSP-функцією і лінійною функцією активації). У програму ST Neural Networks закладено стандартні методи лінійної оптимізації, що дозволяють оптимізувати параметри вихідного лінійного шару при будь-яких заданих параметрах попередніх шарів мережі.

Таким чином, спосіб навчання РБФ-мережі принципово відрізняється від того, який ми мали в разі БЛП. Перш за все, за допомогою навчання без вчителя (тобто коли в навчальних даних в розрахунок приймаються тільки значення вхідних змінних)

визначаються радіальні центри та відхилення (або протяжність). По суті, ідея полягає в тому, щоб помістити центри в середини кластерів навчальних даних, а відхилення обрати так, щоб вони відображали щільність даних у кластерах. Потім відбувається оптимізація лінійного вихідного шару методом псевдозворотних матриць.

Можна діяти інакше: змінити вигляд функцій активації елементів вихідного шару на логістичні (Logistic) і потім навчити цей шар за допомогою методу зворотного поширення, спряжених градієнтів або Левенберга-Маркара, з'єднавши таким чином переваги нелінійності радіальних функцій і стійкості логістичних функцій активації, правда, ціною збільшення часу навчання.

Щоб побудувати РБФ-мережу, необхідно скористатися, як завжди, *Конструктором мереж* і вказати в ньому тип мережі *Радіальна базисна функція*. Кожного разу, коли ви міняєте тип мережі, програма ST Neural Networks автоматично пропонує відповідний режим пре-і постпроцесування і параметри шарів. Як і у випадку багат шарового персептрона, необхідно тільки скорегувати число прихованих елементів. Для даної задачі розумно взяти десять прихованих елементів.

Навчається мережа через діалогове вікно *Радіальные базисные функции - Обучение*, яке відкривається командою *Радіальные базисные функции - Radial Basis Functions, Обучить - Train*.

Навчання РБФ-мережі можна зробити наступним чином:

1. У полі *Радіальное задание* оберемо *K-средние*. Центри радіальних елементів будуть розташовані в центрах кластерів.

2. У полі *Радіальное распространение* оберемо *Изотропные*. Відхилення будуть обрані з урахуванням числа елементів і розкиду навчальних даних.

Як можна помітити, радіальна базисна функція має ще одну перевагу - вона навчається дуже швидко.

Один з недоліків цього методу полягає в тому, що алгоритм псевдозворотних, використовуваний для навчання, може давати велику помилку при дуже малих радіальних відхиленнях, тому методи *Изотропные - Isotropic* і *K-средние - K-Means* потрібно використовувати з обережністю.

1.2.4 Імовірнісні та узагальнено-регресійні нейронні мережі

Імовірнісні нейронні мережі (PNN) застосовуються в задачах класифікації, тому набір даних для них повинен мати одну номінальну вихідну змінну (Patterson, 1996; Bishop, 1995).

Узагальнено-регресійні нейронні мережі (GRNN) використовуються в задачах регресії. Їх можна застосовувати в задачах з вихідними змінними будь-яких типів, але, як правило, з однією числовою вихідною змінною (Patterson, 1996; Bishop, 1995).

Стандартні архітектури мереж, розраховані на кероване навчання (багатошаровий персептрон і радіальна базисна функція), припускають побудову параметричної моделі по наявних навчальних даних, де в якості параметрів виступають ваги. Параметрична модель (мережа) за обсягом зазвичай виявляється набагато менше, ніж набір навчальних даних, і працює досить швидко, хоча для навчання мережі іноді може вимагатися значний час.

До завдання можна підійти інакше: спробувати моделювати шукане відображення більш-менш безпосередньо по навчальних даних. Перевага такого підходу в тому, що тут не потрібно навчання (не рахуючи, можливо, деяких простих дій, які практично зводяться до змін форми подання навчальних даних), а недолік у тому, що в результаті може вийти дуже громізка модель, яка буде займати багато пам'яті і повільно працювати.

PNN- і GRNN-мережі саме і являють собою методи такого типу, «замасковані» під нейронні мережі, призначені відповідно для задач класифікації і регресії. Перший проміжний шар в мережах цих типів складається з радіальних елементів, кожен з яких у разі PNN-мережі зберігає в собі кожне навчальне спостереження, а в разі GRNN-мережі - велике число центрів кластерів (яке, як правило, буває приблизно того ж порядку, що і обсяг навчальної множини). Вихідний сигнал такого радіального елемента являє собою гауссову функцію активації з центром в тій точці, яка зберігається в даному елементі (тим самим «засвідчуючи» про те, що у моделюємій функції є деяка щільність ймовірності в околиці даної точки), а в наступних шарах з цих сигналів складаються оцінки для щільності ймовірностей класів (для PNN) або залежної змінної регресії (для GRNN).

Завдання 3.

Виконати приклади 1, 2 в пакеті *STATISTICA NeuralNetworks* із заданими у прикладах вихідними даними (у тому числі,

номінальними), далі змінити значення вихідних даних завдань, наприклад, задати замість 0 число 0,1, а замість 1 – число 0,9, та проаналізувати результат.

1.3 Контрольні питання

1. Які існують програмні пакети для моделювання нейронних мереж?
2. Які особливості використання пакета Matlab?
3. Які особливості використання пакета STATISTICA NeuralNetworks?
4. Яким чином можна створити нейронну мережу у пакеті Matlab?
5. Яким чином можна створити нейронну мережу у пакеті STATISTICA NeuralNetworks?
6. Яким чином нейронні мережі реагують на малі зміни (перешкоди) вихідних значень?

2 ЛАБОРАТОРНА РОБОТА № 2

Застосування нейронних мереж Кохонена

Мета роботи: навчитися застосовувати нейронні мережі Кохонена на практиці.

2.1 Загальні відомості

Мережа Кохонена сильно відрізняється від всіх інших типів мереж, реалізованих в пакеті *STATISTICA Neural Networks*. У той час як всі інші мережі призначені для задач з керованим процесом навчання, мережі Кохонена головним чином розраховані на некероване навчання.

При керованому навчанні спостереження дані, що становлять навчальні, разом із вхідними змінними містять також і відповідні їм вихідні значення, і мережа повинна відновити відображення, переводячи перші в другі. У разі ж некерованого навчання навчальні дані містять тільки значення вхідних змінних. На перший погляд це може здатися дивним. Як мережа може чомусь навчитися, не маючи вихідних значень? Відповідь полягає в тому, що мережа Кохонена навчається приймати саму структуру даних.

Одне з можливих застосувань таких мереж – розвідувальний аналіз даних. Мережа Кохонена може розпізнавати кластери в даних, а також установлює 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 задачі класифікації. Мережі Кохонена можна використовувати і в тих задачах класифікації, де класи вже задані, тоді перевага представленого буде в тому, що мережа зможе виявити подібність між різними класами.

Інша можлива область застосування – виявлення нових явищ. Мережа Кохонена розпізнає кластери в навчальних даних і відносить всі дані до тих чи інших кластерів. Якщо після цього мережа зустрінеться із спостереженнями, несхожими ні на один з відомих зразків, то вона не зможе класифікувати такий набір і тим самим виявить його новизну.

Мережа Кохонена має всього два шари: вхідний і вихідний, складений з радіальних елементів (вихідний шар називають також шаром топологічної карти). Елементи топологічної карти розташовуються в певному просторі, як правило, двовимірному (в пакеті *STATISTICA Neural Networks* реалізовані також одномірні мережі Кохонена).

Навчається мережа Кохонена методом послідовних наближень. Починаючи з випадковим чином обраним початковим розташуванням центрів, алгоритм поступово покращує його так, щоб уловлювати кластеризацію навчальних даних.

У результаті ітеративної процедури навчання мережа організується таким чином, що елементи, що відповідають центрам, розташованим близько один від одного в просторі входів, будуть розташовуватися близько один від одного і на топологічній карті. Топологічний шар мережі можна уявляти собі як двовимірні ґрати, які потрібно так відобразити в N -мірний простір входів, щоб по можливості зберегти вихідну структуру даних. Звичайно ж, при будь-якій спробі представити N -мірне простір на площині будуть втрачені багато деталей, а проте, такий прийом іноді корисний, тому що він дозволяє користувачеві візуалізувати дані, які ніяким іншим способом зрозуміти неможливо.

Основний ітераційний алгоритм Кохонена складається з декількох послідовних епох, на кожній з яких потрібно обробити кожен з навчальних прикладів і виконати певний набір операцій:

- обрати нейрон-переможець (тобто той, який розташований найближче до вхідного, наприклад);
- скорегувати нейрон-переможець так, щоб він став більш схожий на цей вхідний приклад (взявши зважену суму колишнього центру та навчального прикладу).

В алгоритмі при обчисленні зваженої суми використовується поступово спадний коефіцієнт швидкості навчання, для того щоб на кожній новій епісї корекція ставала все більш точною. В результаті положення центру встановиться в деякій позиції, яка задовільним чином представляє ті спостереження, для яких даний нейрон виявився переможцем.

Властивість топологічної впорядкованості досягається в алгоритмі за допомогою додаткового використання поняття околиці.

Околиця – це кілька нейронів, що оточують нейрон-переможець. Подібно швидкості навчання розмір околиці убуває з часом, так що спочатку до неї належить досить велике число нейронів (можливо, майже вся топологічна карта); на самих останніх етапах околиця стає нульовою (тобто складається тільки з самого нейрона-переможця). Насправді в алгоритмі Кохонена коригування застосовується не тільки до нейрону-переможця, але й до всіх нейронів з його поточної околиці.

Результатом такої зміни околиць є те, що спочатку досить великі ділянки мережі «перетягуються» – і при тому помітно – у бік навчальних прикладів. Мережа формує грубу структуру топології чеського порядку, при якій схожі спостереження активують групи нейронів, що близько лежать на топологічній карті. З кожною новою епохою швидкість навчання і розмір околиці зменшуються, тим самим усередині ділянок карти виявляються все більш тонкі відмінності, що, врешті-решт, призводить до тонкого налаштування кожного нейрона. Часто навчання навмисне розбивають на дві фази: більш коротку – з великою швидкістю навчання і великими околицями, і довшу – з малою швидкістю навчання і нульовими або майже нульовими околицями.

Після того, як мережа навчена розпізнаванню структури даних, її можна використовувати як засіб візуалізації при аналізі даних. За допомогою частот виграшів, які обчислюються для кожного нейрона і показують скільки разів він вигравав при обробці навчальних прикладів, можна визначити, чи розбивається карта на окремі кластери. Можна також дивитися, як при цьому змінюється топологічна карта.

Після того, як кластери виявлені, нейрони топологічної карти позначаються змістовними мітками (в деяких випадках можуть бути помічені і окремі спостереження). Коли топологічна карта в описаному вигляді побудована, на вхід мережі можна подавати нові спостереження. Якщо при цьому нейрон-переможець був раніше позначений ім'ям класу, то мережа здійснює класифікацію. В іншому випадку вважається, що мережа не прийняла ніякого рішення.

При вирішенні задач класифікації в мережі Кохонена використовується так званий поріг доступу. З огляду на те, що в такій мережі рівень активації нейрона є відстань від нього до вхідного прикладу, поріг доступу грає роль максимальної відстані, на якому

відбувається розпізнавання. Якщо рівень активації нейрона-переможця перевищує це порогове значення, то вважається, що мережа не прийняла ніякого рішення. Тому, коли всі нейрони помічені, а пороги встановлені на потрібному рівні, мережа Кохонена може служити детектором нових явищ (вона повідомляє про неприйняття рішення тільки в тому випадку, якщо подане їй на вхід спостереження значно відрізняється від всіх радіальних елементів).

Ідея мережі Кохонена виникла за аналогією з деякими відомими властивостями людського мозку. Кора головного мозку являє собою великий плоский лист площею близько $0,5 \text{ м}^2$ (щоб поміститися в черепі, вона згорнута складками) з відомими топологічними властивостями. Наприклад, ділянка, відповідальна за кисть руки, примикає до ділянки, відповідальній за рух всієї руки, і таким чином всі зображення людського тіла безупинно відображається на цю двовимірну поверхню.

Мережі Кохонена використовуються не так, як мережі інших типів, і пакет ST Neural Networks має спеціальні засоби для роботи з ними, в тому числі:

- вікно Частоти виграшів, в якому показується, де в мережі формуються кластери;
- вікно Топологічна карта, яка показує, які спостереження віднесені до того чи іншого кластеру, і допомагає користувачеві правильно позначити елементи і спостереження.

2.2 Завдання до лабораторної роботи

Завдання 1. Виконати кластеризацію обсягів продажів двох фірм (вибірка по місяцях) за таким принципом: вдалі місяці продажів обох фірм – в одну категорію, невдалі – в іншу, за допомогою програми Statistica 8.

Хід виконання

1. Запустити пакет Statistica 8.

2. Створити нову таблицю.

2.1 File **à** new **à** spreadsheet.

2.2 Поставити 2 змінні по 12 значень кожна, префікс змінних поставити «firm».

3. Заповнити таблицю обсягів продажів фірм по місяцях, як показано на рис. 2.1.

4. У головному меню вибрати «Data Mining» а «Automated Neural Networks».

5. Вибрати «Cluster Analysis».

6. Вибрати змінні Categorical inputs: «firm1», «firm2».

7. Вибрати висоту сітки рівний 2, ширину – 1.

8. Навчити мережу.

9. Вибрати «Kohonen graph» а «select all». За графіком видно, що мережа віднесла по п'ять значень вибірки в кожну категорію.

	1 firm1	2 firm2
1	800	400
2	750	350
3	900	400
4	1100	1500
5	1900	1100
6	2100	3100
7	2200	3200
8	2000	2500
9	1500	2000
10	700	1000
11	200	700
12	2000	800

Рисунок 2.1 – Дані про обсяги продажів фірм по місяцях

10. Вибрати кнопку «Kohonen», і подивитися які значення в яку категорію помістила мережу, а також значення функцій активації в змагальному процесі навчання мережі.

Мережа віднесла вдаль місяці продажів обох фірм в одну категорію, невдалі – в іншу. Помилки обумовлені малим об'ємом вибірки вхідних даних. Вибірка, об'ємом більше тисячі значень, забезпечує низьку ймовірність помилки, однак вимагає значних тимчасових і просторових витрат.

Завдання 2. Задати змінну – значення курсу гривні до долара США (домножить число до цілого) за півроку з кроком в один день. Виконати кластеризацію вхідних даних за допомогою карт Кохонена.

2.3 Контрольні питання

1. Наведіть приклад структури карти Кохонена.
2. Опишіть принцип функціонування мереж, які самоорганізуються.
3. Опишіть принципи ітераційного алгоритму Кохонена.
4. Який тип навчання використовують мережі Кохонена?
5. Яке застосування мереж Кохонена?

3 ЛАБОРАТОРНА РОБОТА № 3 Застосування нейронних мереж Хопфілда

Мета роботи: навчитися застосовувати нейронні мережі Хопфілда на практиці.

3.1 Загальні відомості

Американський дослідник вимірів у 80-х роках 20-го століття запропонував спеціальний тип нейромереж. Названі на його честь мережі Хопфілда є рекурентним або мережами із зворотними зв'язками і призначені для розпізнавання образів. Узагальнена структура цієї мережі рекомендується, як правило, у вигляді системи зі зворотним зв'язком виходу з входом.

У мережі Хопфілда вхідні сигнали нейронів є одночасно і вихідними сигналами мережі: $x_i(k) = y_i(k-1)$, при цьому збудливий вектор особливо не виділяється. У класичній системі Хопфілда відсутній зв'язок нейрона з власним виходом, що відповідає $w_{ii} = 0$, а вся матриця ваг є симетричною: $w_{ij} = w_{ji}$

$$\hat{W} = \hat{W}^T. \quad (3.1)$$

Симетричність матриці ваг гарантує збіжність процесу навчання. Процес навчання мережі формує зони тяжіння деяких точок рівноваги, відповідних навчальним даним. При використанні асоціативної пам'яті маємо справу з навчальним вектором, або з множиною цих векторів, які в результаті проведеного навчання визначають розташування конкретних точок тяжіння (атракторів).

Кожен нейрон має функцію активації Сігнум зі значеннями ± 1 :

$$\text{sign}(a) = \begin{cases} 1, & a \geq 0 \\ -1, & a < 0 \end{cases}. \quad (3.2)$$

Це означає, що вихідний сигнал i -го нейрона визначається функцією:

$$y_i = \text{sign}\left(\sum_{j=1}^N w_{ij}x_j + b_i\right), \quad (3.3)$$

де N позначає кількість нейронів, $N = n$. Часто постійна складова b_i , яка визначає поріг спрацьовування окремих нейронів, дорівнює 0. Тоді циклічне проходження сигналу в мережі Хопфілда можна надати співвідношенням:

$$y_i(k) = \text{sign}\left(\sum_{j=1}^N w_{ij}y_j(k-1)\right), \quad (3.4)$$

з початковою умовою $y_j(0) = x_j$.

У процесі функціонування мережі Хопфілда можна виділити два режими: навчання та класифікації. У режимі навчання на основі відомих навчальних вибірок підбираються вагові коефіцієнти w_{ij} . У режимі класифікації при зафіксованих значеннях ваг і введенні конкретного початкового стану нейронів $\mathbf{y}(0) = \mathbf{x}$ виникає перехідний процес, що протікає відповідно з виразом (3) і закінчується в одному з локальних стійких положень, що задається біполярним вектором зі значеннями, $y_j = \pm 1$ для якого $\mathbf{y}(k) = \mathbf{y}(k-1)$.

Навчання не носить рекурентного характеру. Досить ввести значення (правило Хебба) ваг, висловивши їх через проекції вектора точки тяжіння еталонного образу:

$$w_{ij} = \frac{1}{N} x_i x_j, \quad (3.5)$$

Згідно з цим правилом мережа дає правильний результат при вхідному прикладі, що збігається з еталонним зразком, оскільки:

$$\sum_{j=1}^N w_{ij}x_j = \frac{1}{N} \sum_{j=1}^N x_i(x_j x_j) = x_i \sum_{j=1}^N 1 \cdot \frac{1}{N} = x_i, \quad (3.6)$$

бо внаслідок біполярності значень елементів вектора $\mathbf{x}$ завжди $x_j^2 = 1$.

При введенні великої кількості навчальних вибірок $\mathbf{x}^{(k)}$ для $k = 1, 2, \dots, p$ ваги w_{ij} підбираються згідно узагальненому правилу Хебба відповідно до яких:

$$w_{ij} = \frac{1}{N} \sum_{k=1}^l x_i^{(k)} x_j^{(k)}. \quad (3.7)$$

Завдяки такому режиму навчання ваги беруть значення, що визначаються усередненням множини вибірок, які здатні до навчання. У разі множини вибірок, які здатні до навчання, актуальним стає питання про стабільність асоціативної пам'яті.

Мережа Хопфілда є автоасоціативною мережею. Дискретна мережа Хопфілда має наступні характеристики: вона містить один шар елементів, кожен елемент зв'язується з усіма іншими елементами, але не пов'язаний з самим собою; за один крок роботи оновлюється лише один елемент мережі; елементи оновлюються у випадковому порядку; вихід елемента обмежений значеннями 0 або 1.

3.2 Завдання до лабораторної роботи

Приклад 1. Розглянемо мережу Хопфілда з чотирма нейронами і визначимо чотири точки рівноваги:

```
T = [+1 -1; -1 +1; +1 +1; -1 -1];
T = T';
plot ( T (1,:), T (2,:), 'rh ', ' MarkerSize
', 13), hold on;
axis ([-1.1 1.1 -1.1 1.1]);
title (' Hopfield Network State Space ');
xlabel ('a (1)');
ylabel ('a (2)');
net = newhop (T);
[Y, Pf, Af] = sim (net, 4, [], T);
Y
Pf
Af
pause
color = 'rgbmy';
```

```

for i = 1:25
    a = (rands (2,1));
    [y, Pf, Af] = sim (net, (січня 1920 },{ },
a);
    record = [ cell2mat (a) cell2mat (y)];
    start = cell2mat (a);
    plot (start (1,1), start (2,1), 'kx',
record (1,:), record (2,:) , color (rem (i, 5)
+1), 'LineWidth', 5)
end

```

На рис. 3.1 показано поведінку навченої мережі при випадкових початкових умовах *a*.

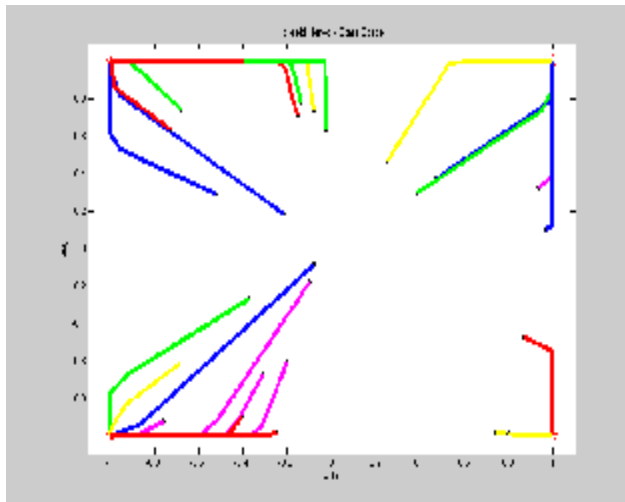


Рисунок 3.1 - Поведінка мережі Хопфілда при випадкових початкових умовах *a*

Приклад 2. Розпізнавання символів. В архіві «Hopfield_Matlab.zip» лежить приклад реалізації розпізнавання символів на основі мережі Хопфілда. Програма «Hopf0.m» виведе еталонні символи. Програма «Hopf0.m» генерує випадкову матрицю (символ) і крок за кроком звертає його в один з еталонних символів.

Завдання 1. Виконати приклад 1 та приклад 2. Представити роботу мережі в матричній формі (виключити цикл за елементами

матриці). Забезпечити роботу мережі Хопфілда при роботі з входами $(0, 1)$ і $(-1, 1)$. Здати всі завдання викладачеві.

3.3 Контрольні питання

1. Наведіть у графічному вигляді структуру нейронної мережі Хопфілда.
2. Які типи задач успішно вирішуються за допомогою мережі Хопфілда?
3. Опишіть особливості мережі Хопфілда, що дають змогу успішно вирішувати ці задачі.
4. Які режими можна виділити в процесі функціонування мережі Хопфілда?
5. Який недолік використання нейронної мережі Хопфілда для задач оптимізації?

4 ЛАБОРАТОРНА РОБОТА № 4

Розпізнавання образів

Мета роботи: навчитися використовувати апарат нейронних мереж для розпізнавання цифр, а також ідентифікації людей по особах на фотографії.

4.1 Загальні відомості

Теорія розпізнавання образів – розділ кібернетики, що розвиває теоретичні основи та методи класифікації та ідентифікації предметів, явищ, процесів, сигналів, ситуацій і т.п. об'єктів, які характеризуються кінцевим набором деяких властивостей і ознак. Такі завдання вирішуються досить часто, наприклад, при переході або проїзді вулиці за сигналами світлофора. Розпізнавання кольору лампи світлофора, яка загорілась, і знання правил дорожнього руху дозволяє прийняти правильне рішення про те, можна чи не можна переходити вулицю в даний момент.

Створення штучних систем розпізнавання образів залишається складною теоретичною і технічною проблемою. Необхідність у такому розпізнаванні виникає в різних галузях – від військової справи та систем безпеки до відцифровування аналогових сигналів.

Традиційно задачі розпізнавання образів включають до кола завдань штучного інтелекту.

Можна виділити два основних напрями в розпізнаванні образів:

1. Вивчення здібностей до розпізнавання, якими володіють живі істоти, пояснення та моделювання їх;
2. Розвиток теорії та методів побудови пристроїв, призначених для вирішення окремих завдань у прикладних цілях.

Розпізнавання образів – це віднесення вихідних даних до певного класу за допомогою виділення істотних ознак, що характеризують ці дані, із загальної маси несуттєвих даних.

Розрізняють декілька методів розпізнавання образів. Для оптичного розпізнавання образів можна застосувати метод перебору виду об'єкта під різними кутами, масштабами, зміщеннями і т. д. Для літер треба перебирати шрифт, властивості шрифту і т. д. Другий підхід – знайти контур об'єкта та дослідити його властивості (зв'язність, наявність кутів тощо). Ще один підхід – використовувати

штучні нейронні мережі. Цей метод вимагає або великої кількості прикладів задачі розпізнавання (з правильними відповідями), або спеціальної структури нейронної мережі, що враховує специфіку даної задачі.

У середовищі MATLAB набори вихідних даних (не плутати з цільовими даними) можна як завантажувати з файлів зображень, так і створювати безпосередньо. Рис. 4.1 ілюструє можливе вміст масиву вихідних даних.

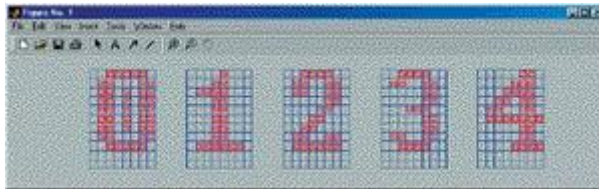


Рисунок 4.1 – Графічне представлення масиву вихідних даних

Отримати навчальні дані можна, наклавши шум на набір вихідних даних (рис. 4.2).

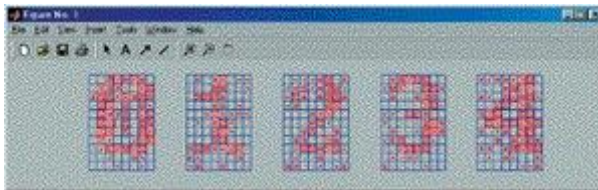


Рисунок 4.2 – Навчальні дані

У задачах класифікації, до яких відноситься даний приклад, кількість виходів мережі відповідає числу поділюваних мережею класів. Цей факт необхідно враховувати при виборі архітектури мережі і на етапі формування цільових даних.

Мережа класифікації дає найбільше значення на виході, який відповідає відповідному класу. При добре сконструйованій і навченій NN значення інших виходів будуть помітно менше.

Для вирішення цього завдання можна обрати мережу Feed-forward backprop з п'ятьма сігмоїдними нейронами першого шару і п'ятьма лінійними нейронами другого шару. Алгоритм навчання –

Левенберга-Маркара. З такою конфігурацією мережу після восьми епох навчання дає помилку порядку 10^{-30}

Відзначимо, що вже зараз успішно розробляються різноманітні системи розпізнавання образів (візуальних, звукових), що базуються на нейронних мережах. Прикладом можуть послужити програми розпізнавання сканованого тексту. Вони досить широко поширені серед власників оптичних сканерів і в реальному масштабі часу справляються з поставленими завданнями.

4.2 Завдання до лабораторної роботи

Завдання 1. Написати власну програму розпізнавання цифр від 0 до 9. Показати викладачеві.

Завдання 2. Провести розпізнавання облич на фотографії. Для цього виконати наступне:

- обрати кожному студентові фотографію з кількістю людей на ній не менше чотирьох;
- застосувати апарат знаходження особи на фотографії;
- застосувати нейронні мережі для розпізнавання людини по обличчю на фотографії.

4.3 Контрольні питання

1. Опишіть методи розпізнавання образів.
2. Які галузі використання кожного з методів розпізнавання образів?
3. Дуже стисло опишіть програмні продукти, які існують для розпізнавання облич.
4. Які основні напрями в розпізнаванні образів?
5. Які конкретні задачі успішно вирішуються в галузі розпізнавання образів?

5 ЛАБОРАТОРНА РОБОТА № 5

Застосування генетичних алгоритмів

Мета роботи: навчитися застосовувати генетичні алгоритми на практиці.

5.1 Загальні відомості

Основи теорії генетичних алгоритмів сформульовані Холландом в його основоположній роботі і в подальшому були розвинені рядом дослідників. Найбільш відомою і цитованою в даний час є монографія Д. Голдберга, де систематично викладено основні результати та області практичного застосування генетичних алгоритмів.

Генетичні алгоритми використовують принципи і термінологію, запозичені у біологічної науки – генетики. У генетичних алгоритмах кожна особина представляє потенційне рішення деякої проблеми. У класичному генетичному алгоритмі особина кодується рядком двійкових символів – хромосомою, кожен біт якої називається геном. Множина особин – потенційних рішень становить популяцію. Пошук (суб) оптимального вирішення проблеми виконується в процесі еволюції популяції – послідовного перетворення однієї кінцевої множини рішень у іншу за допомогою генетичних операторів репродукції, кросинговеру та мутації.

Еволюційні обчислення використовують наступні механізми природної еволюції:

- 1) Перший принцип заснований на концепції виживання найсильніших і природного відбору за Дарвіном який був сформульований їм у 1859 році в книзі «Походження видів шляхом природного відбору». Згідно Дарвіну особини, які краще здатні вирішувати завдання в своєму середовищі, виживають і більше розмножуються (репродуцують). У генетичних алгоритмах кожна особина представляє собою рішення деякої проблеми аналогією з цим принципом особини з кращими значеннями цільової (фітнес) функції мають великі шанси вижити і репродуцувати. Формалізація цього принципу дає оператор репродукції.

- 2) Другий принцип обумовлений тим фактом, що хромосома нащадка складається з частин, отриманих з хромосом батьків. Цей

принцип був відкритий в 1865 році Менделем. Його формалізація дає основу для оператора схрещування (кросинговеру).

3) Третій принцип заснований на концепції мутації, відкритої в 1900 році де Вре. Спочатку цей термін використовувався для опису істотних (різких) змін властивостей нащадків і набуття ними властивостей, відсутніх у батьків. За аналогією з цим принципом генетичні алгоритми використовують подібний механізм для різкої зміни властивостей нащадків і тим самим, підвищують різноманіття особин в популяції (множині рішень).

Ці три принципи складають ядро еволюційних обчислень. Використовуючи їх, популяція (множина рішень даної проблеми) еволюціонує від покоління до покоління.

Генетичний алгоритм бере множину параметрів оптимізаційної проблеми та кодує їх послідовностями кінцевої довжини в деякому кінцевому алфавіті (в найпростішому випадку двійковий алфавіт «0» і «1»).

Простий генетичний алгоритм випадковим чином генерує початкову популяцію стрінгів (хромосом). Потім алгоритм генерує наступне покоління (популяцію), за допомогою трьох основних генетичних операторів:

1. Оператор репродукції (ОР);
2. Оператор схрещування (кросинговеру, ОК);
3. Оператор мутації (ОМ).

Генетичні оператори є математичної формалізацією наведених вище трьох основоположних принципів Дарвіна, Менделя і де Вре природної еволюції.

Генетичний алгоритм працює до тих пір, поки не буде виконано задану кількість поколінь (ітерацій) процесу еволюції або на деякій генерації буде отримано задану якість або внаслідок передчасної збіжності при попаданні в деякий локальний оптимум.

У кожному поколінні множина штучних особин створюється з використанням старих і додаванням нових з хорошими властивостями. Генетичні алгоритми – не просто випадковий пошук, вони ефективно використовують інформацію, накопичену в процесі еволюції. У процесі пошуку рішення необхідно дотримувати баланс між «експлуатацією» отриманих на поточний момент кращих рішень і розширенням простору пошуку. Різні методи пошуку вирішують цю проблему по-різному.

5.2 Завдання до лабораторної роботи

Завдання 1. Мінімізувати функцію однієї змінної

$$f(x) = 8x - 16 - 12\sqrt[3]{(x+4)^2}$$

Хід виконання (робота у пакеті Matlab). Напишемо M-file для даної функції і збережемо його у поточній папці під ім'ям `ex2.m`.

```
function y = x2(x)
y = 8 * x - 16 - 12 * (x + 4) ^ (2 / 3);
```

Відкриємо вікно тулбокса за допомогою *gatool*.

У полі *fitness function* введемо ім'я цільової функції `@ex2`.

Встановимо значення параметрів ГА: кількість особин в популяції = 10, кількість поколінь = 100 (у вікні критерію зупинки алгоритму), початковий відрізок = [-4; 1]. У розділі *plots* встановимо прапорці для *best fitness*, *best individual*, *distance*. Клацнемо по кнопці *start*.

У результаті завершення процесу у вікні *final point* з'явиться значення змінної x , відповідне мінімуму функції, а у вікні *status and result* можна побачити знайдене мінімальне значення цільової функції.

Для даної задачі результати вийшли наступні мінімум функції досягається в точці $x = -2.9972$ та $f(-2.9972) = -51.99998959105959$ (рис. 5.1).

Перший рисунок відображає зміну значення цільової функції. Видно, що, починаючи з 80 популяції, алгоритм зійшовся до рішення. На другому рисунку зображена найкраща особина. Третій рисунок відповідає зсуванням відстані між особинами в поколіннях. Особини стають однакові (хеммінгова відстань = 0) в останніх 18 поколіннях. Генетичний алгоритм потрібно запустити кілька разів, а потім вибрати оптимальне рішення. Це пов'язано з тим, що початкова популяція формується з використанням генератора випадкових чисел.

Переконалися в правильності рішення можна побудувавши графік функції (рис. 5.2).

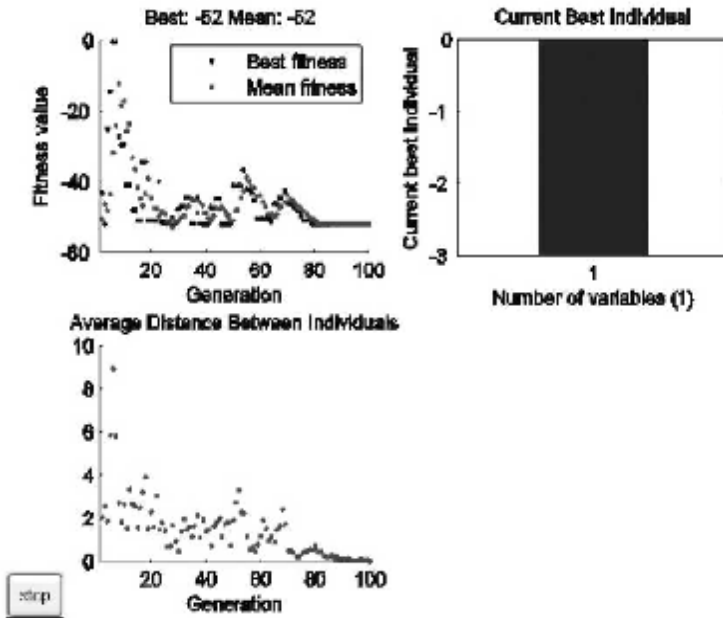


Рисунок 5.1 – Графічний аналіз рішення

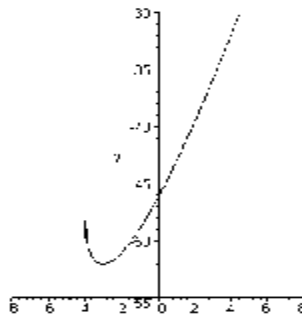


Рисунок 5.2 – Графік функції

Те ж саме можна було б отримати, використовуючи функції *gaoptimset* і *ga*. Щоб подивитися M-File виберіть в меню «File» вікна «Genetic Algorithm Tool» команду «Generate M-file», збережіть файл під іншим ім'ям і перегляньте код. Для даної задачі отримали:

```

function
[X,FVAL,REASON,OUTPUT,POPULATION,SCORES] = ex2q
% This is an auto generated M file to do
optimization
% with the Genetic Algorithm and
77% Direct Search Toolbox. Use GAOPTIMSET for
default
% GA options structure.
% Fitness function
fitnessFunction = @ex2;
% Number of Variables
nvars = 1 ;
% Start with default options
options = gaoptimset;
% Modify some parameters
options = gaoptimset(options,'PopInitRange',[-
4 ; -1 ]);
options =
gaoptimset(options,'PopulationSize',10);
options = gaoptimset(options,'MutationFcn',
{@mutationgaussian 1 1});
options = gaoptimset(options,'Display','off');
options = gaoptimset(options,'PlotFcns',
{@gaplotbestf
@gaplotbestindiv @gaplotdistance });
% Run GA
[X,FVAL,REASON,OUTPUT,POPULATION,SCORES] =
ga(fitnessFunction,nvars,options);

```

Завдання 2. Максимізувати функцію двох змінних:
 $z(x,y) = \exp(-x^2 - y^2) + \sin(x+y)$

Хід виконання. Тулбокс по генетичним алгоритмам вирішує тільки задачі мінімізації, тому для знаходження максимуму функції $f(x)$ слід мінімізувати функцію $-f(x)$. Це пояснюється тим, що точка мінімуму $-f(x)$ є деякою точкою $f(x)$, в якій досягається максимум.

Напишемо M-file для функції $z(x) = -f(x)$ і збережемо його у поточній папці під ім'ям ex13.m:

```
function z = ex13 (x)
z=-(exp(-x(1)^2-x(2)^2)+sin(x(1)+x(2)));
```

Відкриємо вікно тулбокса за допомогою *gatool*.

У полі *fitness function* введемо ім'я цільової функції *@ex13*.

Встановимо значення параметрів ГА: кількість змінних = 2, до кількостей особин у популяції = 10, кількість поколінь = 100 (у вікні критерію зупинки алгоритму), початковий відрізок = [-1; 3]. Для побудовано енія графіків у розділі *plots* встановимо прапорці для *best fitness*, *best individual*, *distance*. Клацнемо по кнопці *start*.

У результаті завершення процесу у вікні *final point* з'явиться значення змінної *x*, відповідне мінімуму функції, а у вікні *status and result* можна побачити знайдене мінімальне значення цільової функції *z (x)*.

Для даної задачі результати вийшли наступні: максимум функції досягається в точці $x = 0.46419$, $y = 0.42406$ та $f(0.46419; 0.42406) = 1.449$.

Поверхня функції представлена на рис. 5.3.

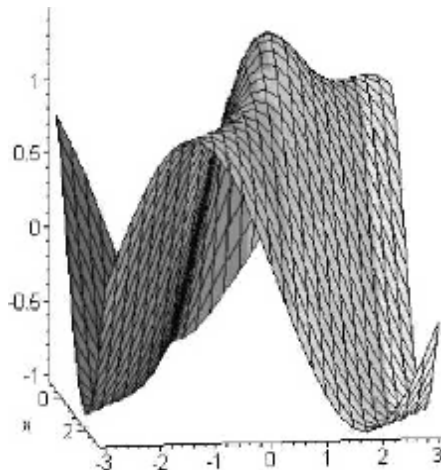


Рисунок 5.3 – Поверхня функції

Завдання 3. Знайти максимум функції $y(x) = \sqrt[3]{(x+4)^2} - 2x - 8$

Знайти мінімум функції: $z(x, y) = (y-3)\exp(-x^2 - y^2)$

5.3 Контрольні питання

1. Основі принципи еволюційних обчислень.
2. Що є генетичним алгоритмом?
3. Основні оператори генетичних алгоритмів.
4. Галузі використання генетичних алгоритмів.
5. Розкрийте поняття «паралельні генетичні алгоритми».

6 ЛАБОРАТОРНА РОБОТА № 6

Експертні системи

Мета роботи: розробити найпростіший приклад експертної системи.

6.1 Загальні відомості

Експертні системи – це напрям досліджень в галузі штучного інтелекту зі створення обчислювальних систем, що вміють приймати рішення, схожі з рішеннями експертів у заданій предметній області.

Як правило, експертні системи створюються для вирішення практичних завдань у деяких вузькоспеціалізованих областях, де велику роль грають знання «бувалих» фахівців. Експертні системи були першими розробками, які змогли залучити велику увагу до результатів досліджень в галузі штучного інтелекту.

Експертні системи мають одну велику відмінність від інших систем штучного інтелекту: вони не призначені для вирішення якихось універсальних завдань, як наприклад нейронні мережі або генетичні алгоритми. Експертні системи призначені для якісного вирішення задач у певній області, в окремих випадках – областях.

Експертне знання – це поєднання теоретичного розуміння проблеми і практичних навичок її рішення, ефективність яких доведено в результаті практичної діяльності експертів у даній області. Фундаментом експертної системи будь-якого типу є база знань, яка складається на основі експертних знань фахівців. Правильно обраний експерт і вдала формалізація його знань дозволяє наділити експертну систему унікальними та цінними знаннями. Лікар, наприклад, добре діагностує хвороби і ефективно призначає лікування, не тому, що він має якісь вроджені здібності, а тому що має якісну медичну освіту і великий досвід у лікуванні своїх пацієнтів. Тому цінність всієї експертної системи як закінченого продукту на 90% визначається якістю створеної бази знань.

Експертна система – це не проста програма, яка пишеться одним або кількома програмістами. Експертна система є плодом спільної роботи експертів у даній предметній області, інженерів по знаннях і програмістів.

Але варто відзначити, що зустрічаються випадки, коли програми пишуться самими експертами в даній області.

Експерт надає необхідні знання про ретельно відібрані приклади проблем та шляхи їх вирішення. Наприклад, при створенні експертної системи діагностики захворювань лікар розповідає інженерові по знаннях про відомі йому захворювання. Далі експерт розкриває список симптомів, які супроводжують кожне захворювання і на закінчення розповідає про відомі йому методи лікування. Інженер по знаннях, формалізує всю отриману інформацію у вигляді бази знань і допомагає програмісту в написанні експертної системи.

Зараз активний розвиток експертних систем дещо припинилося, і цьому є ряд причин:

1. Передача експертним системам «глибоких» знань про предметну область є великою проблемою. Як правило, це є наслідком складності формалізації евристичних знань експертів.

2. Експертні системи нездатні надати осмислені пояснення своїх міркувань, як це робить людина. Як правило, експертні системи всього лише описують послідовність кроків, зроблених у процесі пошуку рішення.

3. Налагодження і тестування будь-якої комп'ютерної програми є досить трудомісткою справою, але перевіряти експертні системи особливо важко. Це є серйозною проблемою, оскільки експертні системи застосовуються в таких критичних областях, як управління повітряним і залізничним рухом, системами зброї і в ядерній промисловості.

4. Експертні системи мають ще один великий недолік: вони нездатні до самонавчання. Для того, щоб підтримувати експертні системи в актуальному стані необхідно постійне втручання в базу знань інженерів по знаннях. Експертні системи, позбавлені підтримки з боку розробників, швидко втрачають свою актуальність.

6.2 Завдання до лабораторної роботи

Розробити колективно всією групою під керівництвом викладача експертну систему, що дозволяє неспеціалістові в певній області зробити правильний вибір. Наприклад, розробити експертну систему, що дозволяє неспеціалістові автотранспортних засобів придбати бажаний автомобіль.

6.3 Контрольні питання

1. Що собою являють експертні системи?
2. Призначення експертних систем.
3. Що позначає термін «експертне знання»?
4. Переваги використання експертних систем.
5. Недоліки використання експертних систем.

Вимоги до оформлення письмових звітів

Письмовий звіт необхідно надавати до кожної лабораторної роботи. При оформленні кожного звіту необхідно дотримуватися наступних вимог. Звіт повинен містити:

1. Титульний аркуш із зазначенням теми, дисципліни, студента, який виконав звіт, викладача, який приймає звіт;
2. Результати виконання завдань лабораторної роботи;
3. Відповіді на контрольні питання;
4. Висновки.

ЛИТЕРАТУРА

1. Demuth H., Beale M. Neural Network Toolbox. For Use with MATLAB. The MathWorks Inc. 1992-2000.
2. Нейронные сети. STATISTICA Neural Networks. Москва. Горячая линия – Телеком. 2000.
3. Principe J.C., Euliano N.R., Lefebvre W.C. Neural and Adaptive Systems. Fundamentals Through Simulations. N. York. J. Wiley & Sons Inc. 2000.
4. Luo F-L., Unbehauen R. Applied Neural Networks for Signal Processing. Cam-bridge University Press. 1998.
5. Официальный сайт поддержки STATISTICA Neural Networks [Электронный ресурс] / Режим доступа: <http://www.statsoft.ru/home/textbook/default.htm>
6. Медведев В.С., Потемкин В.Г. Нейронные сети. Матлаб 6. М.: Диалог МИФИ, 2002. – 496с.
7. Иванников В., Ланнэ А.. Matlab для DSP. Нейронные сети: графический интерфейс пользователя [Электронный ресурс] / Режим доступа: <http://catalog.gaw.ru/index.php?page=document&id=1438>
8. Каллан Р. Основные концепции нейронных сетей. М.: Издательский дом «Вильямс», 2001. – 287с.
9. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика. – М.: Мир, 1992.
10. Круглов В.В., Борисов В.В. Искусственные нейронные сети. М.: Горячая линия – Телеком, 2001. – 382с.
11. Осовский С. Нейронные сети для обработки информации. М.: Финансы и статистика, 2002. – 344с.
12. Тарков М.С. Нейрокомпьютерные системы. М.: БИНОМ, 2006. –142с.
13. Нейронные сети. STATISTICA Neural Networks. М.: Горячая линия – Телеком, 2001. – 182с.
14. Зайцевский И.В., Свиридов А.П., Слесарев Д.А. Нейронные сети и их приложения. М.: МЭИ, 2002. – 95с.
15. Дьяконов В.П., Круглов В.В. MATLAB 6.5 SP1/7/7 SP1/7 SP2 Simulink 5/6. Инструменты искусственного интеллекта и биоинформатики. М.: СОЛОН-ПРЕСС, 2006. – 456 с.

16. Распределение заданий в компьютерной системе на базе нейронной сети / М.Ю. Юрич, Р.К. Кудерметов // Материалы Седьмой Международной конференции-семинара «Высокопроизводительные параллельные вычисления на кластерных системах», 26–30 нояб. 2007 г., Нижний Новгород. – 2007. – с. 347–353.

17. Yurich M. An Optimal Distribution of Tasks in Parallel Computing System / M. Yurich // Proceedings of the International Conference «Modern Problems of Radio Engineering, Telecommunications and Computer Science», 19–23 Febr. 2008, Lviv–Slavsko, Ukraine. – 2008. – pp.585–587.

18. Юрич М.Ю. Гибридный алгоритм распределения задач в вычислительной системе / Юрич М.Ю. // Сборник научных трудов Донецкого национального технического университета. Серия: Информатика, кибернетика и вычислительная техника. – 2010. – Выпуск 12 (65) – с. 66–72.

19. Скобцов Ю.О. Основы эволюционных вычислений. Учебное пособие. Д.: ДонНТУ, 2008. – 326 с.

20. Ту Дж., Гонсалес Р. Принципы распознавания образов, М. 1978.

21. Панченко Т.В. Генетические алгоритмы: учебно-методическое пособие / под ред. Ю.Ю. Тарасевича. – Астрахань: Издательский дом «Астраханский университет», 2007. – 87 с.

22. Экспертные системы [Электронный ресурс] // Портал искусственного интеллекта / Режим доступа: <http://www.aiportal.ru/articles/expert-systems/expert-systems.html>

23. Нейронные сети. *STATISTICA Neural Networks*: Методология и технологии современного анализа данных / Под редакцией В. П. Боровикова. - 2-е изд., перераб. и доп. - М.: Горячая линия - Телеком, 2008. - 392 с., ил.