

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни «Основи теорії інтелектуальних систем»

для студентів спеціальності 123 «Комп'ютерна інженерія»
за освітньою програмою «Комп'ютерні системи та мережі»
денної форми навчання

Методичні вказівки до лабораторних робіт з дисципліни «Основи теорії інтелектуальних систем» для студентів спеціальності 123 «Комп'ютерна інженерія» за освітньою програмою «Комп'ютерні системи та мережі» денної форми навчання / Укл.: М.Ю. Тягунова – Запоріжжя: НУ «Запорізька політехніка», 2022. – 58 с.

Укладач: М.Ю. Тягунова, к.т.н., доцент

Рецензент: Г.Г. Киричек, к.т.н., доцент

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

Затверджено:
На засіданні кафедри
«Комп'ютерні системи та мережі»
Протокол № 2
від 23 вересня 2022 р.

Рекомендовано до видання
НМК факультету КНТ
Протокол № 2
від 29 вересня 2022 р.

ЗМІСТ

1	ЛАБОРАТОРНА РОБОТА №1. НЕЙРОННІ МЕРЕЖІ У СЕРЕДОВИЩІ MATLAB.....	3
1.1	Теоретичні відомості	5
1.2	Керуючі елементи <i>NNTool</i>	6
1.3	Поділ лінійно невіддільних множин	15
1.4	Вирішення задачі апроксимації	20
1.5	Імпорт/експорт даних	25
1.6	Завдання до виконання	27
1.7	Зміст звіту	27
1.8	Контрольні питання	28
2	ЛАБОРАТОРНА РОБОТА №2. ЗАСТОСУВАННЯ НЕЙРОННИХ МЕРЕЖ КОХОНЕНА	29
2.1	Теоретичні відомості	29
2.2	Задача кластеризації	31
2.3	Вирішення задачі кластеризації валютного курсу.....	35
2.4	Завдання до виконання	35
2.5	Зміст звіту	35
2.6	Контрольні питання	35
3	ЛАБОРАТОРНА РОБОТА №3. РОЗПІЗНАВАННЯ ОБРАЗІВ	36
3.1	Теоретичні відомості	36
3.2	Задача на розпізнавання літер.....	36
3.3	Завдання до виконання	40
3.4	Зміст звіту	41
3.5	Контрольні питання	41
4	ЛАБОРАТОРНА РОБОТА №4. РОЗПІЗНАВАННЯ ОБЛИЧЧЯ	42
4.1	Теоретичні відомості	42
4.2	Налаштування програмного середовища	43
4.3	Розпізнавання обличчя з відеопотоку	45
4.4	Завдання до виконання	47
4.5	Зміст звіту	47
4.6	Контрольні запитання.....	48
5	ЛАБОРАТОРНА РОБОТА №5. ЗАСТОСУВАННЯ ГЕНЕТИЧНИХ АЛГОРИТМІВ.....	49

5.1	Теоретичні відомості	49
5.2	Задача для мінімізації функції	51
5.3	Задача для максимізації функції.....	55
5.4	Завдання до виконання	56
5.5	Зміст звіту	57
5.6	Контрольні запитання.....	57
ЛІТЕРАТУРА		58

1 ЛАБОРАТОРНА РОБОТА №1

НЕЙРОННІ МЕРЕЖІ У СЕРЕДОВИЩІ *MATLAB*

Мета роботи: набути навички роботи у програмному середовищі *MATLAB* при роботі з нейронними мережами.

1.1 Теоретичні відомості

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 середовищі *MATLAB*.

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

Обробка сигналів в технологіях *NN* виконується за допомогою *NN* без пам'яті або *NN* із пам'яттю. І в одному, 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.

Далі буде розглянуто *NN* прямого поширення, тобто без зворотних зв'язків, особливістю яких є їх стійкість.

Після того, як структуру *NN* обрано, повинні бути встановлені її параметри, здійснено вибір 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 обирати структури *NN* з великого переліку і надає безліч алгоритмів навчання для кожного типу мережі.

Лабораторна робота сприяє опрацюванню та засвоєнню наступних питань:

- призначення графічних керуючих елементів *NNTool*;
- підготовка даних для створення нейронної мережі;
- створення нейронної мережі;
- навчання мережі;
- перевірка створеної та навченої мережі (прогін мережі).

1.2 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» (*Neural Network/Data Manager*) (рис. 1.1).

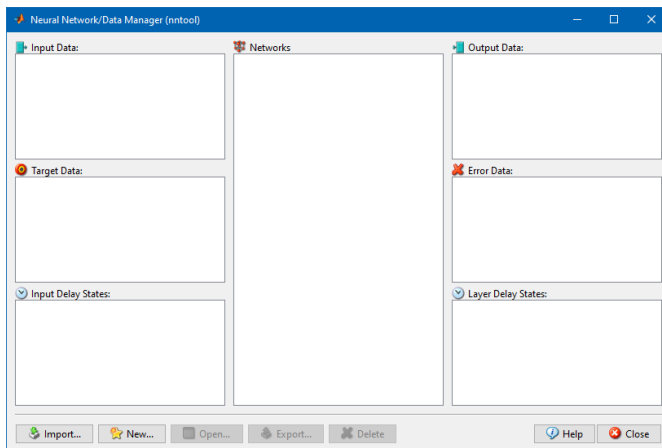


Рисунок 1.1 – Головне вікно *NNTool*

Вікно «*Neural Network/Data Manager*» має функціональні клавіші з наступними призначеннями:

- імпорт (*Import ...*) – імпорт даних з робочого простору MATLAB або з зовнішнього файлу в простір змінних NNTool;
- новий (*New...*) – виклик вікна, що дозволяє створювати нові мережі та набори даних;
- відкрити (*Open...*) – виклик вікна для перегляду інформації про обраний елемент;
- експорт (*Export ...*) – експорт даних з простору змінних NNTool в робочий простір MATLAB або в зовнішній файл;
- видалити (*Delete*) – видалення обраного об'єкту;
- допомога (*Help*) – короткий опис керуючих елементів цього вікна;
- закрити (*Close*) – закрити поточне вікно.

При роботі з *NNTool* важливо пам'ятати, що клавіші *Open*, *Export* та *Delete* (зображені на рис. 1.1 як неактивні) діють стосовно того об'єкту, який у даний момент позначено виділенням. Якщо такого об'єкту немає або над виділеним об'єктом неможливо здійснити вказану дію, відповідна клавіша неактивна.

Розглянемо створення нейронної мережі за допомогою *NNTool* на прикладі.

ПРИКЛАД 1. Нехай потрібно створити нейронну мережу, що реалізує логічну функцію «AND».

Крок 1. Створення мережі. Оберемо 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 підготувати набори навчальних і цільових даних. Складемо таблицю достовірності для логічної функції «AND», де *P1* і *P2* – входи, а *Res* – бажаний вихід (таблиця 1.1).

Таблиця 1.1 - Таблиця достовірності логічної функції «AND»

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

Щоб задати матрицю, що складається з чотирьох векторів-рядків, у якості вхідних даних, скористаємося кнопкою *New*. У вікні слід обрати вкладку *Data* та занести зміни у відповідності із рис. 1.2 і натиснути клавішу «Створити» (*Create*).

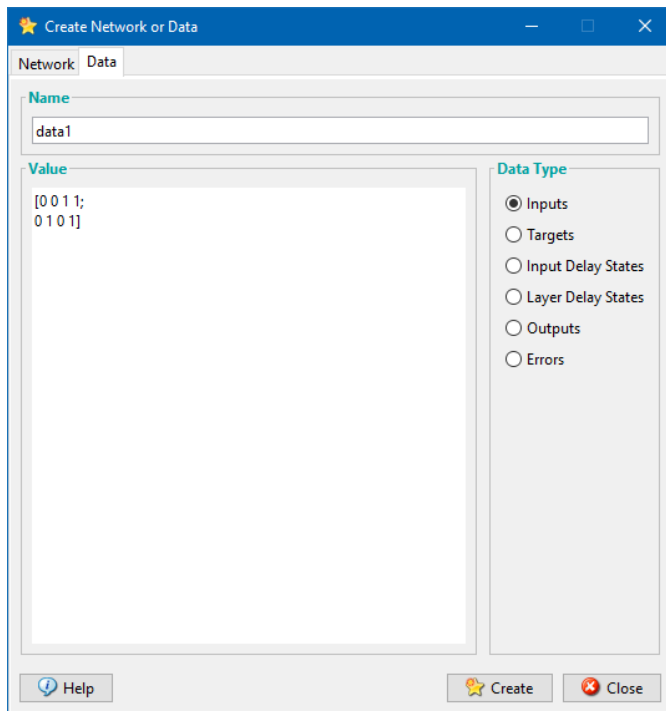


Рисунок 1.2 – Створення вхідних векторів

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

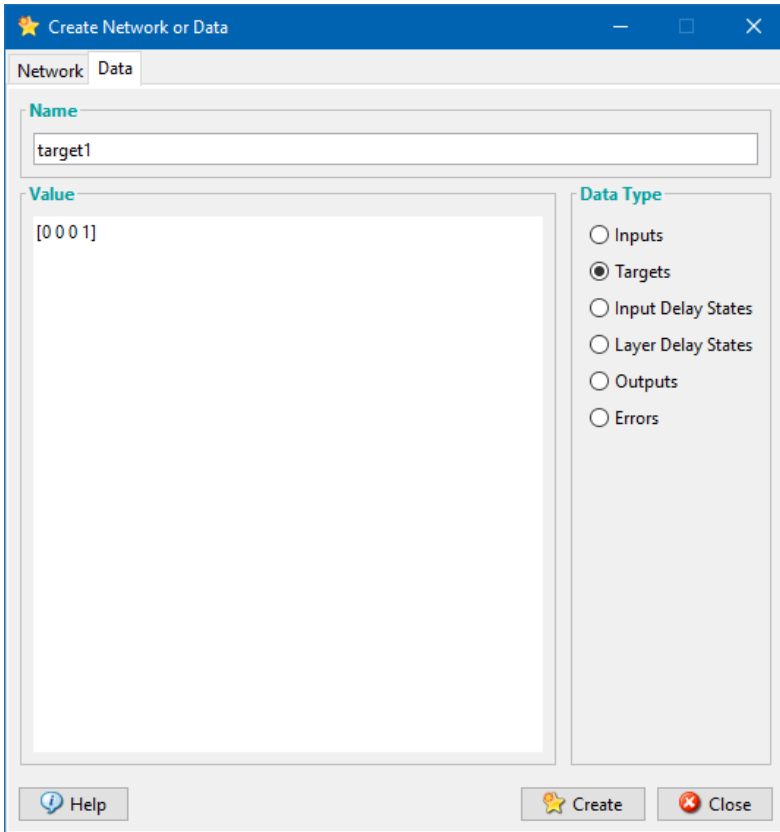


Рисунок 1.3 – Створення цільового вектору

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

Тепер слід перейти до створення нейронної мережі. Натискаємо кнопку *New*, обираємо вкладку *Network* і заповнюємо форму згідно рис. 1.4.

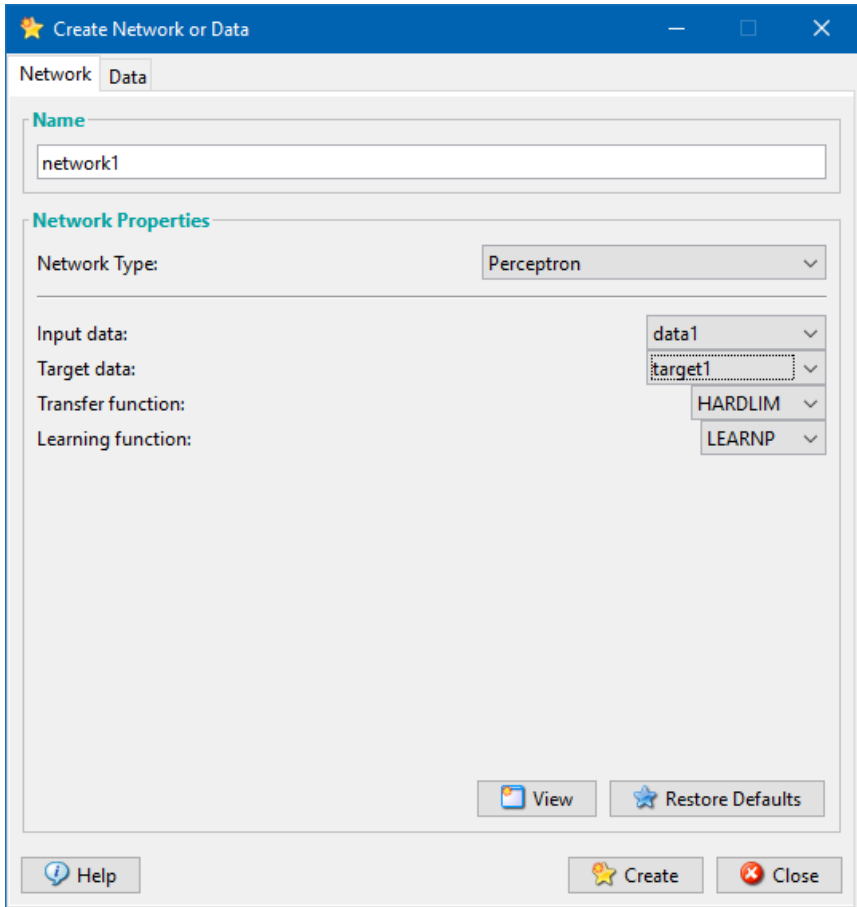


Рисунок 1.4 – Вікно «Створення мережі»

Поля форми (рис. 1.4) несуть наступні смислові навантаження:

- ім'я мережі (Name) – це ім'я об'єкту створюваної мережі;
- тип мережі (Network Type) – визначає тип мережі і, в контексті обраного типу, обумовлює відповідні параметри мережі, представлені полями для введення, розташованими нижче. Таким чином, для різних типів мереж вікно змінює свій зміст;
- вхідні дані (Input data) – набір вхідних даних нейронної мережі;
- цільові дані (Target data) – набір даних цільової функції;

- передаточна функція (Transfer function) – в цьому пункті обирається 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, що всі дії було проведено вірно. На рис. 1.5 зображено персептронну мережу з вихідним блоком, який реалізує передаточну функцію із жорстким обмеженням. Кількість нейронів у шарі дорівнює одному, що символічно відображається розмірністю вектора-стовпця на виході шару і вказується числом безпосередньо під блоком передаточної функції. Розглянута мережа має два входи, бо розмірність вхідного вектора-стовпця дорівнює двом.

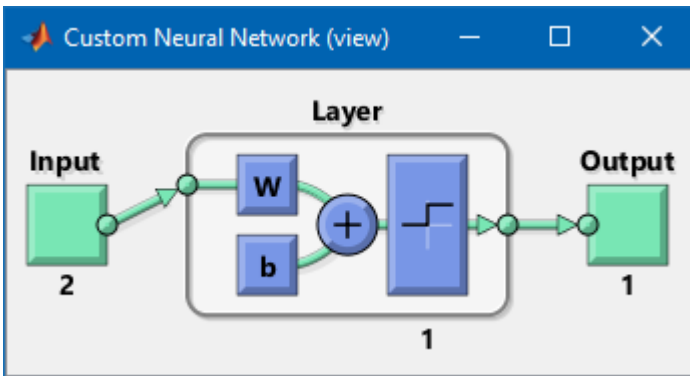


Рисунок 1.5 – Попередній перегляд створюваної мережі

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

У результаті виконаних операцій в розділі «Мережі» (*Networks*) головного вікна *NNTool* з'явиться об'єкт під назвою *network1*.

Крок 2. Навчання мережі. Наша мета – побудувати нейронну мережу, яка виконує функцію логічного «AND». 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. У *MATLAB* реалізовано більшість відомих алгоритмів навчання нейронних мереж, серед яких для персептронних мереж розглянутого виду представлено два. Створюючи мережу, ми вказали *LEARNP* як функцію, що реалізує алгоритм навчання (рис. 1.4).

Повернемося до головного вікна *NNTool*. На даному етапі інтерес представляє створена мережа *network1*. Обравши її, подвійним натисканням або натисканням кнопки *Open* відкривається вікно перегляду мережі. Необхідно обрати вкладку *Train*, що містить, у свою чергу, ще одну панель вкладок (рис. 1.6).

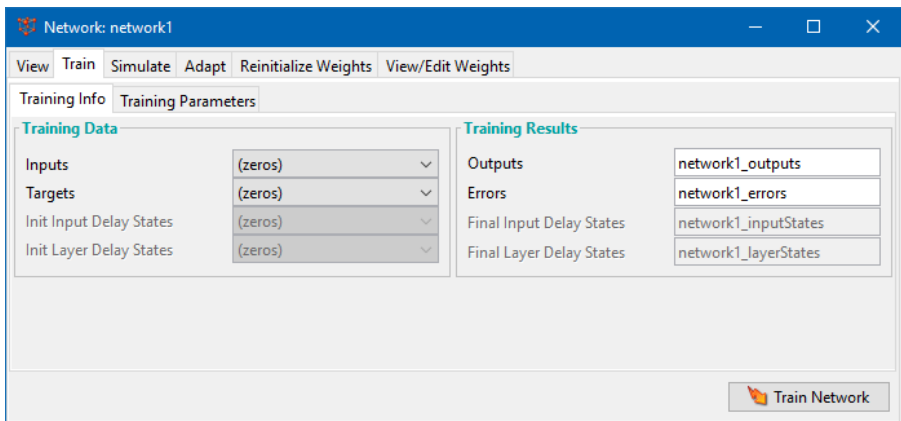


Рисунок 1.6 – Вікно параметрів мережі, відкрите на вкладці «Навчання» (Train)

Щоб почати навчання, необхідно обрати вхідні дані (*Inputs*) та цільову функцію (*Targets*), після чого потрібно натиснути кнопку «Навчити мережу» (*Train Network*).

Після чого відкриється вікно тренування нейронних мереж («Neural Network Training») (рис. 1.7).

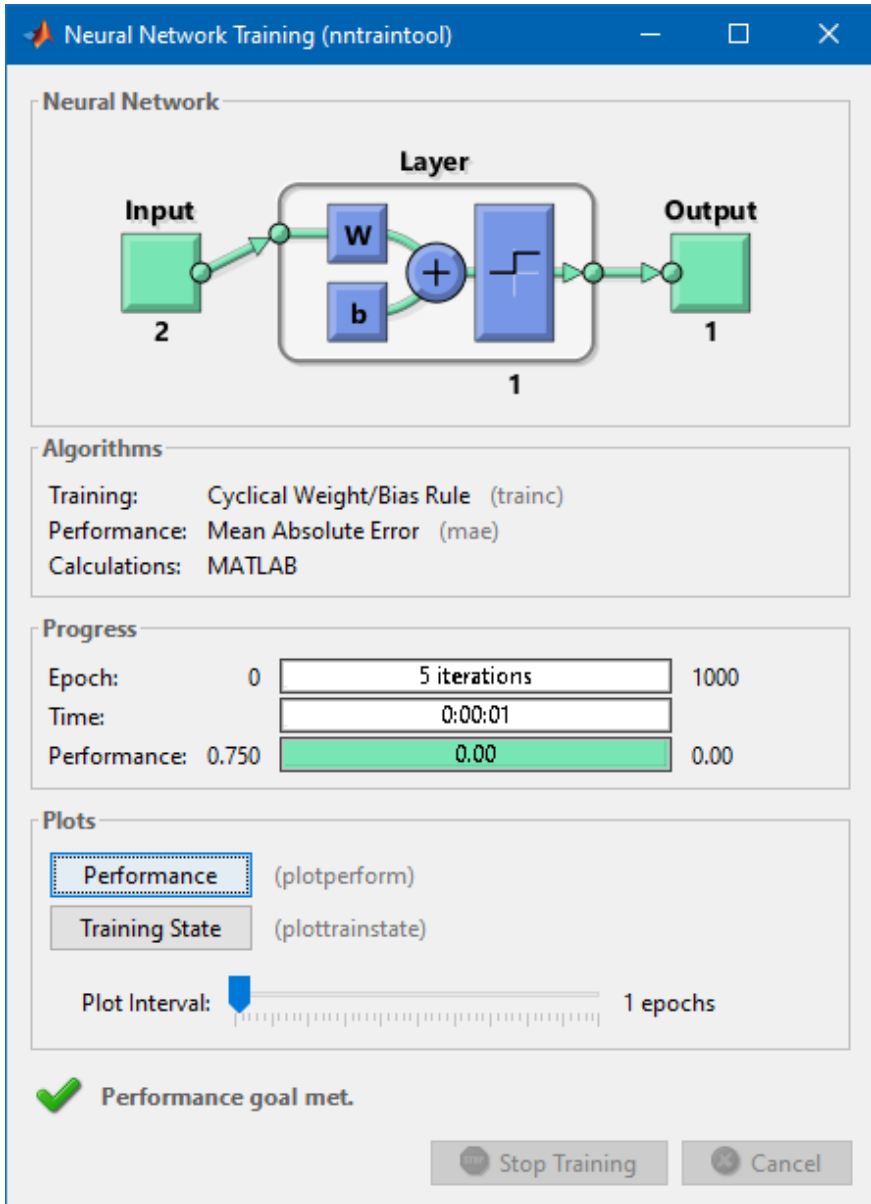


Рисунок 1.7 – Вікно тренування нейронних мереж («Neural Network Training»)

Для перегляду графіку навчання нейронної мережі у поточному вікні необхідно натиснути кнопку «*Performance*». Після чого відкриється вікно з результатами навчання нейронної мережі (рис. 1.8).

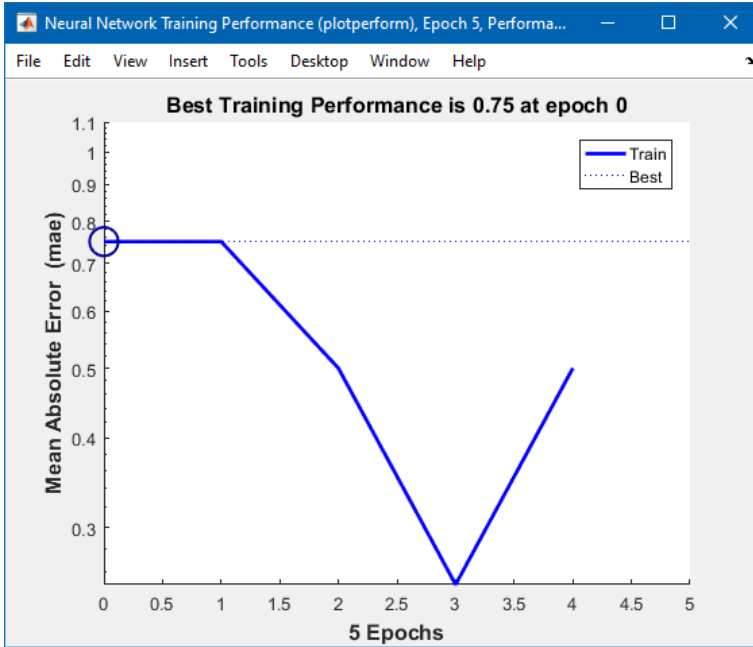


Рисунок 1.8 – Крива навчання

З рисунку 1.8 видно, що навчання було зупинене, коли цільова функція досягла встановленої величини ($goal = 0$) і подальше тренування не принесло результату.

Слід зазначити, що для персептронів, що мають функцію активації із жорстким обмеженням, помилка розраховується як різниця між цільовими і одержуваними вихідними даними.

Отже, алгоритм навчання знайшов точне рішення задачі. У методичних цілях переконаємося в правильності вирішення задачі шляхом прогону навченої мережі. Для цього, у головному вікні необхідно елемент `network1_outputs` і порівняти результати з записаними значеннями цільової функції (рис. 1.9).

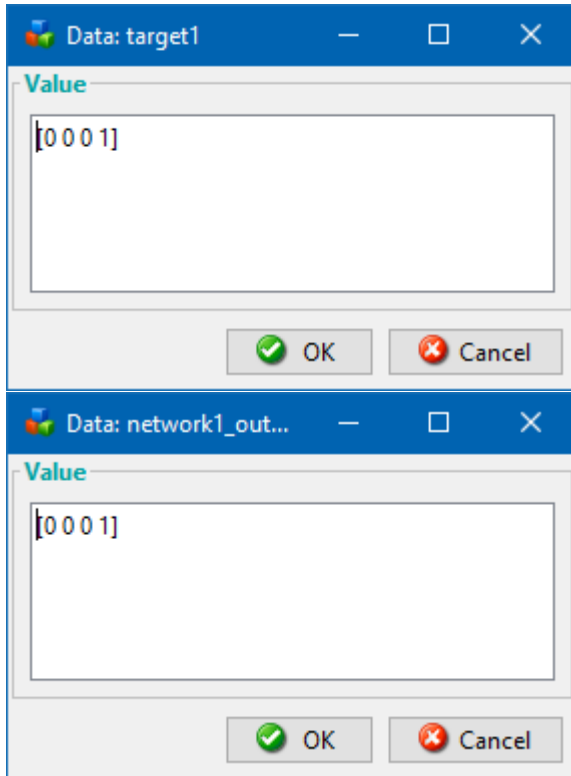


Рисунок 1.9 – Порівняння значень цільової функції із вихідними значеннями мережі

Крім цього, можна переглянути елемент *network1_errors*. Якщо всі значення матриці відповідають «0», то мережа навчена правильно і помилок, під час тренування, не відбулось.

1.3 Поділ лінійно невіддільних множин

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

ПРИКЛАД 2. Нехай потрібно створити нейронну мережу, що реалізує логічну функцію «виключаюче АБО» – XOR.

Розглянемо таблицю істинності цієї функції (таблиця 1.2).

Таблиця 1.2 - Таблиця достовірності логічної функції «XOR»

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, задамо множину вихідних значень у просторі входів за наступним правилом (рис. 1.10): поєднання входів $P1$ і $P2$, при яких вихід Res обертається в нуль, позначаються кружечком, а ті, при яких Res обертається в одиницю – хрестиком.

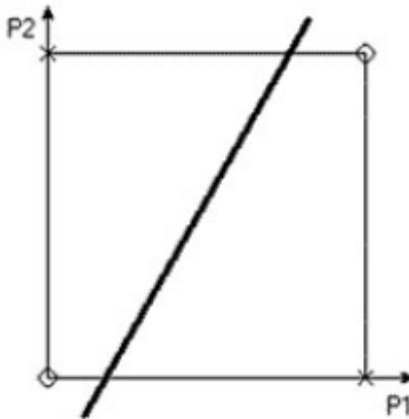


Рисунок 1.10 – Стан логічного елемента XOR

Наша мета – провести межу, яка відокремлює множину нулів від множини хрестиків. Із вмісту рис. 1.10 видно, що неможливо провести пряму лінію, яка б відокремила нулі від одиниць. Саме в цьому сенсі множина нулів лінійно невіддільна від множини одиниць, і перцептрони, розглянуті раніше, в принципі, не можуть вирішити поставлену задачу.

Якщо ж використовувати персептрони зі спеціальними нелінійними функціями активації, наприклад сигмоїдальні, то вирішення задачі можливе.

Аналогічним чином, вказуємо значення цільової функції *target2* згідно таблиці істинності (рис. 1.11).

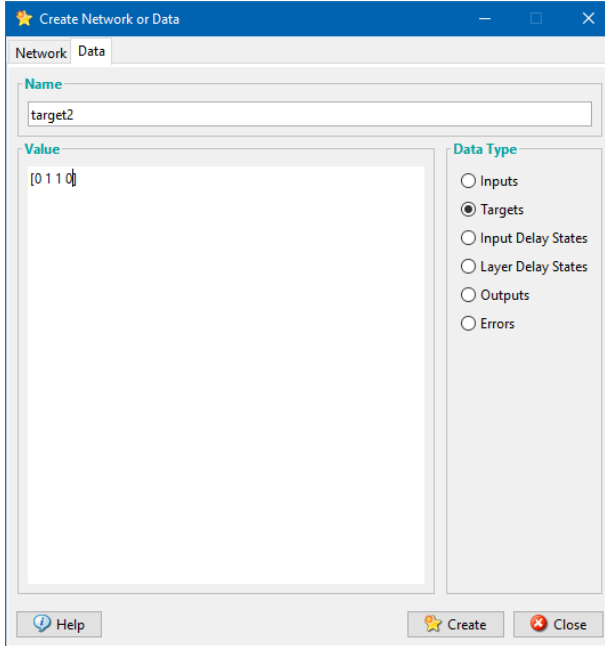


Рисунок 1.11 – Створення цільового вектору

Створюємо нову мережу. Тип мережі обираємо як *Feed-forward backprop*, вхідні дані – *data1*, цільова функція – *target2*, кількість нейронів у прихованому шарі – 2, в якості функції помилки вкажемо *MSE* (*Mean Square Error* – середньоквадратична помилка). Інші поля залишимо за замовчуванням (рис. 1.12).

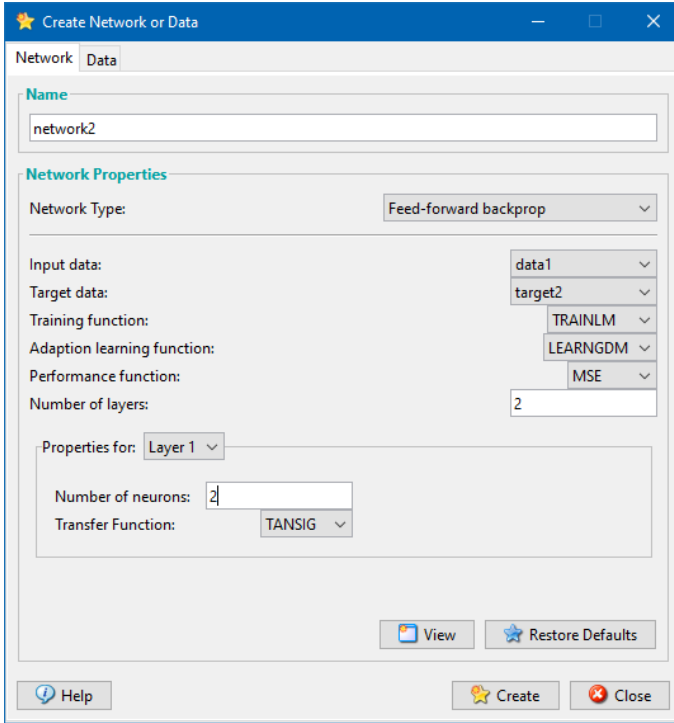


Рисунок 1.12 – Вікно «Створення мережі»

Переглянемо створену мережу. У результаті повинні отримати структуру перцептронів із двома нейронами прихованого шару, у яких функції активації сигмоїдальні, і одним вихідним нейроном з лінійною функцією активації (рис. 1.13).

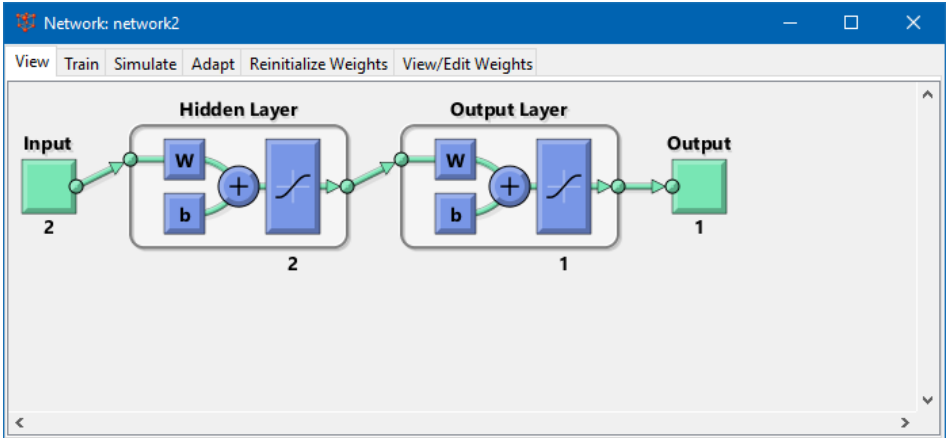


Рисунок 1.13 – Мережа для розв'язання задачі XOR

Далі відкриваємо вкладку *Reinitialize Weights* та натискаємо кнопку «Initialize Weights» (рис. 1.14)..

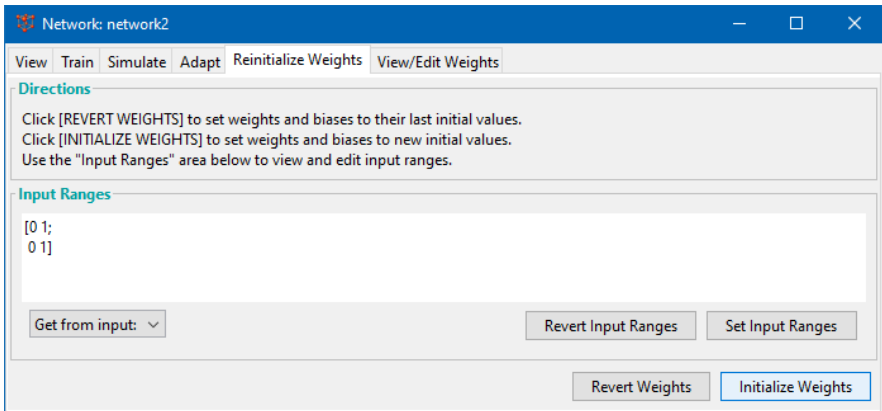


Рисунок 1.14 – Встановлення ваги з файлу вхідних параметрів

Повторюємо дії з прикладом 1, вказуємо вхідні дані та цільову функцію. Переглядаємо результат навчання (рис. 1.15).

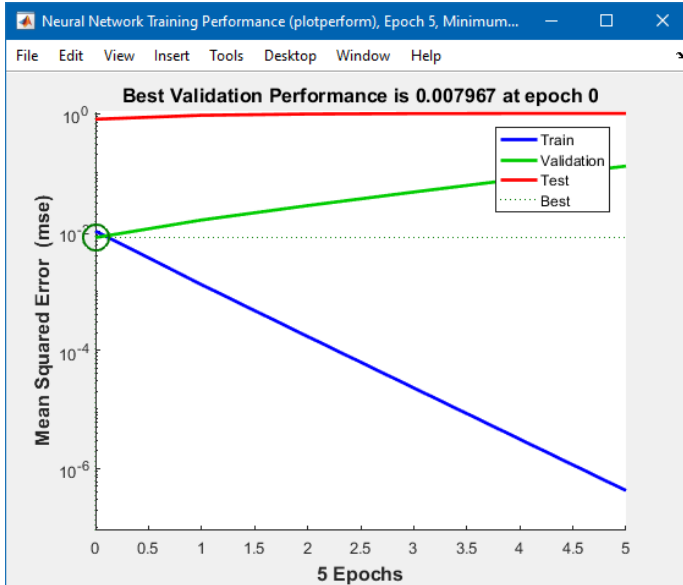


Рисунок 1.15 – Крива навчання в задачі XOR

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

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

Задача синтезу елемента XOR є також прикладом задачі класифікації. Вона відображає спільний підхід до вирішення подібного роду задач.

1.4 Вирішення задачі апроксимації

Одною з найбільш цікавих 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:

$$\sin\left(\frac{5\pi x}{N} + \sin\left(\frac{7\pi x}{N}\right)\right)$$

Підготуємо цільові дані (*target3*), ввівши в поле «Значення» (*Value*) вікна створення нових даних вираз:

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

Тепер підготуємо набір навчальних даних(*data2*), задавши їх наступним виразом: 1:100.

Оберемо перцептрон (*Feed-Forward Back Propagation*) з тринадцятьма сигмоїдальними (*TANSIG*) нейронами прихованого шару і одним лінійним нейроном вихідного шару. Навчання будемо здійснювати, використовуючи алгоритм Левенберга- Марквардта (*Levenberg-Marquardt*), який реалізує функція *TRAINLM*. Функція помилки – *MSE*. Отримана мережа має вигляд, зображений на рисунку 1.16.

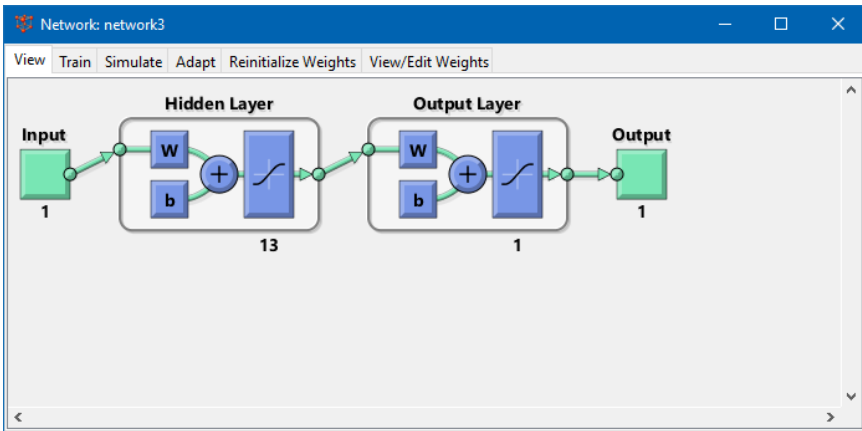


Рисунок 1.16 – Архітектура мережі для вирішення задачі апроксимації

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. 1.17).

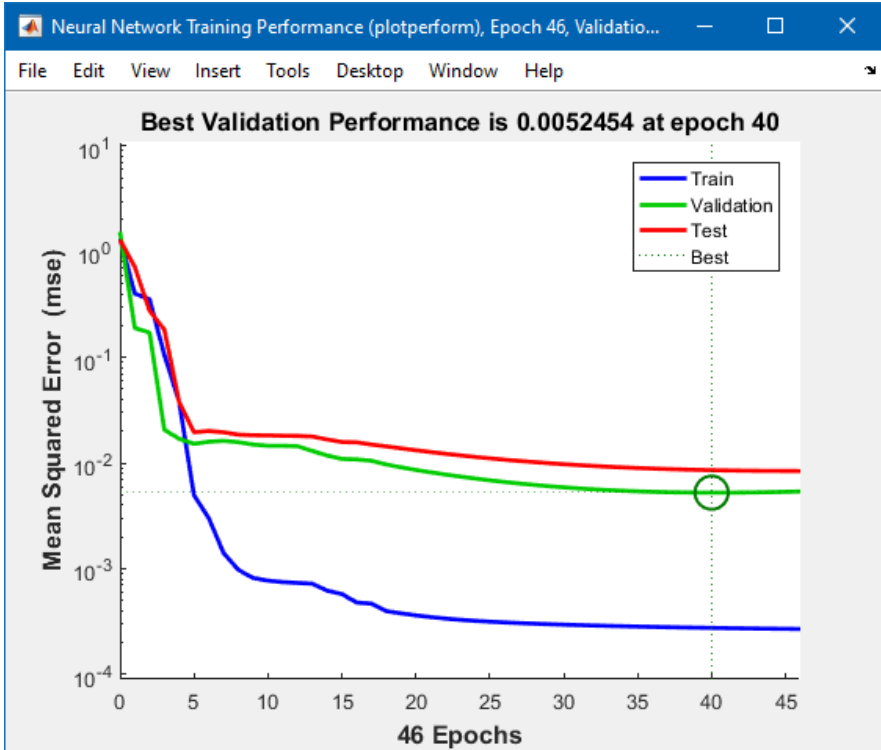


Рисунок 1.17 – Крива навчання в задачі апроксимації

Для перевірки коректності навчання мережі, порівняємо вихідні значення мережі зі значеннями цільової функції (рис. 1.18).

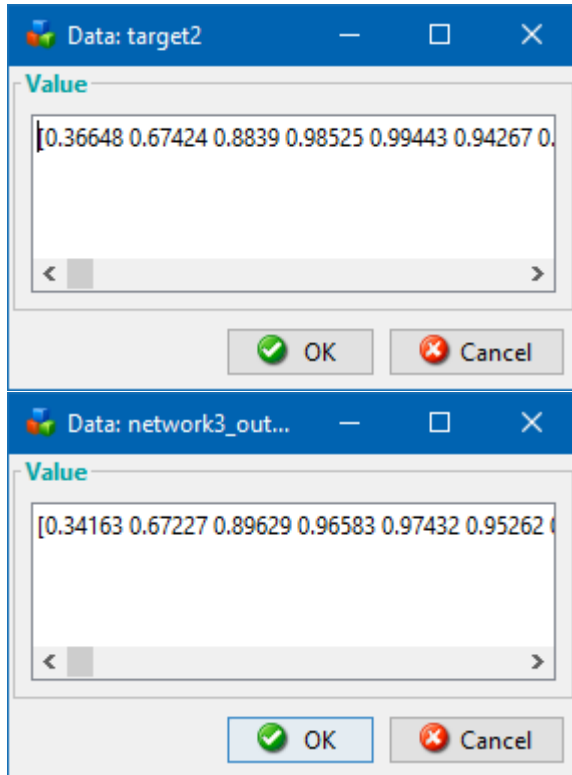


Рисунок 1.18 – Порівняння значень цільової функції із вихідними значеннями мережі

Рисунок 1.19 ілюструє різницю між цільовими даними та отриманою апроксимуючою кривою.

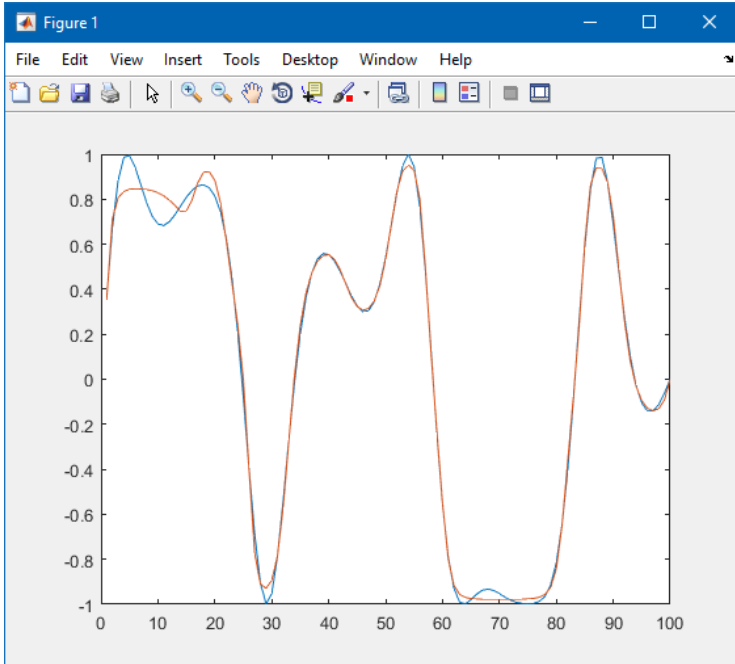


Рисунок 1.19 – Червона крива – цільові дані, синя крива – апроксимуюча функція

З рис. 1.17 та рис. 1.19 видно, наскільки зменшилася помилка апроксимації за 100 епох навчання. Форма кривих навчання на останніх епохах свідчить також про те, що точність наближення може бути підвищена. Щоб отримати графік співставлення кривих навчання, як наведено на рис. 1.15, слід виконати наступні дії:

–отримати файли результатів та помилок за допомогою команди Simulate;

–виділити усі файли у головному вікні NNTool та виконати команду Export (докладний опис команди – див. п. 1.5);

–в командному вікні MATLAB описати функції f1 (цілі) та f2 (результату) в залежності від вхідних даних (data1), а також команди побудови відповідних графіків, а саме:

```
>>f1=target3(data2) ;
>>f2=network_outputs3(data2) ;
>>figure
>>plot(data2, f1, data2, f2)
```

Зверніть увагу на необхідність використання символу «;» для розділення описів функцій $f1, f2, \dots, fn$.

1.5 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 із 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 замовчанням тут встановлена опція завантаження з робочого простору *MATLAB*, тому в центрі вікна з'являється список змінних, що належать йому. Обравши 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 (рис. 1.20). Після того, як всі дії успішно виконано, ім'я імпортованої змінної з'явиться в одному зі списків головного вікна *NNTool*.

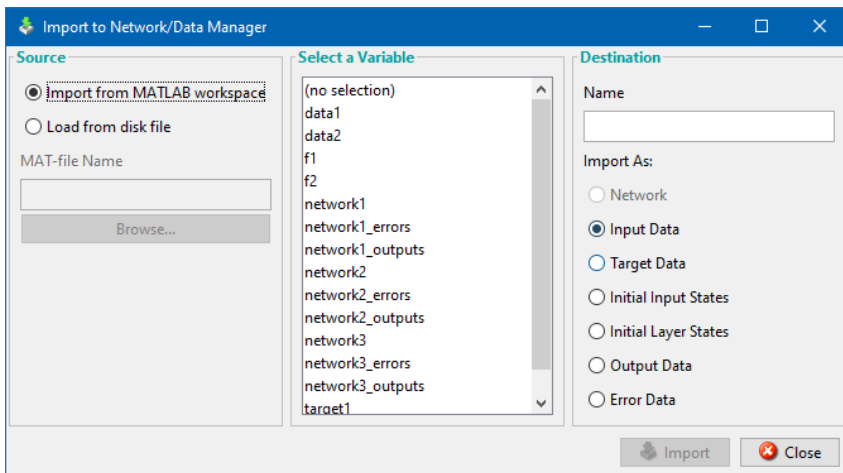


Рисунок 1.20 - Вікно імпорту та завантаження даних з *MAT*-файлу

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.

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* (рис. 1.21). Тепер виділені змінні скопійовано в робочий простір *MATLAB*.

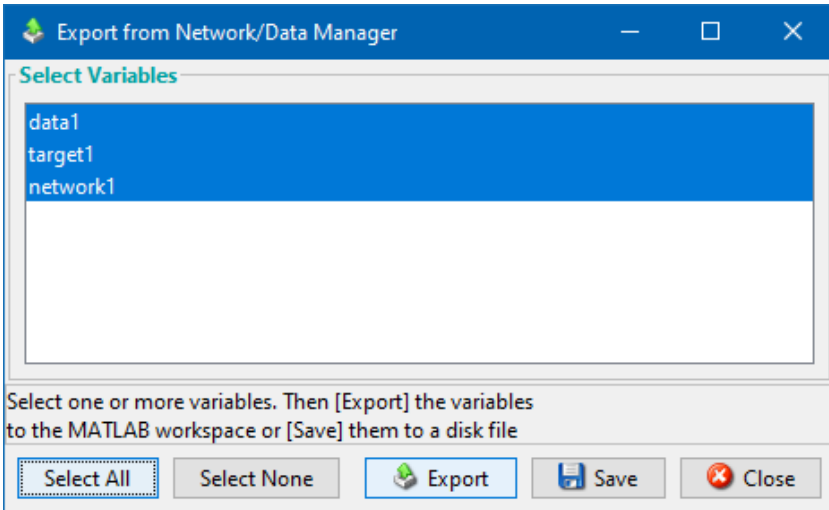


Рисунок 1.21 – Вікно експорту та збереження даних у MAT-файлі

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.

1.6 Завдання до виконання

Необхідно виконати приклади 1-3 і показати їх викладачеві. При виконанні прикладів потрібно проводити серії експериментів та вести записи щодо процесу та результатів спостережень – «журнал досліджень», в котрому слід фіксувати, як впливає на результати навчання зміна різних параметрів процесу, таких як вибір режиму навчання, тип передаточної функції, кількість епох навчання тощо.

За результатами експериментів слід зробити висновки та навести їх у звіті.

1.7 Міст звіту

Звіт повинен мати такі складові:

- титульний лист;
- мета роботи;
- результат виконання завдань;
- відповіді на контрольні питання.

1.8 Контрольні питання

1. Які задачі вирішуються нейронними мережами, приклади застосування нейронних мереж.
2. Навчання нейронної мережі – визначення поняття, призначення.
3. Особливості використання середовища *MATLAB* для моделювання нейронних мереж.
4. Керуючі елементи *NNTool*. Призначення *target*-вектору – вектору цільових даних.
5. Процес створення нейронної мережі засобами *NNTool*.

2 ЛАБОРАТОРНА РОБОТА №2

ЗАСТОСУВАННЯ НЕЙРОННИХ МЕРЕЖ КОХОНЕНА

Мета роботи: навчитися застосовувати нейронні мережі Кохонена на практиці.

2.1 Теоретичні 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.

Ідея мережі Кохонена виникла за аналогією з деякими відомими властивостями людського мозку. Кора головного мозку являє собою великий плоский лист площею близько 0,5 м² (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 12* 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 Задача кластеризації

Необхідно виконати кластеризацію обсягів продаж двох фірм (вибірка по місяцях) за принципом:

- вдалі місяці продажів обох фірм - в першу категорію;
- невдалі – в другу.

Для цього необхідно запустити середовище Statistica 12 та створити нову таблицю (New – Spreadsheet).

В поле кількість змінних (*Number of Variables*) ввести 2, у поле кількість значень (*Number of cases*) ввести 12, обрати тип даних (*Default data type*) Integer та натиснути кнопку OK (рис. 2.1).

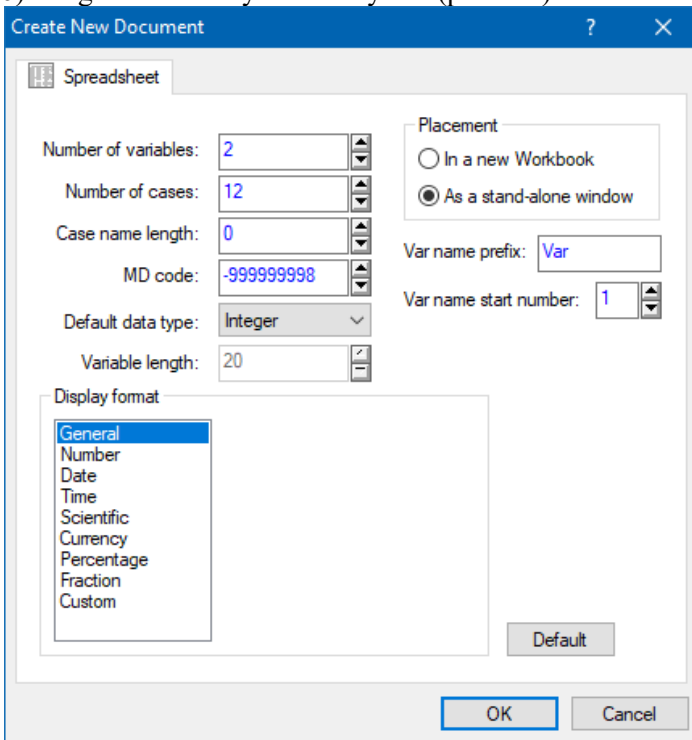


Рисунок 2.1 – Створення таблиці значень

Далі необхідно заповнити таблицю обсягів продаж по місяцях значеннями як показано на рис. 2.2.

	1 Var1	2 Var2
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 2.2 – Значення таблиці обсягів продаж

Далі необхідно відкрити вкладку *Data Mining* та обрати *Neural Networks*. У вікні, що відкрилось необхідно обрати тип аналізу *Cluster analysis* та натиснути ОК (рис. 2.3).

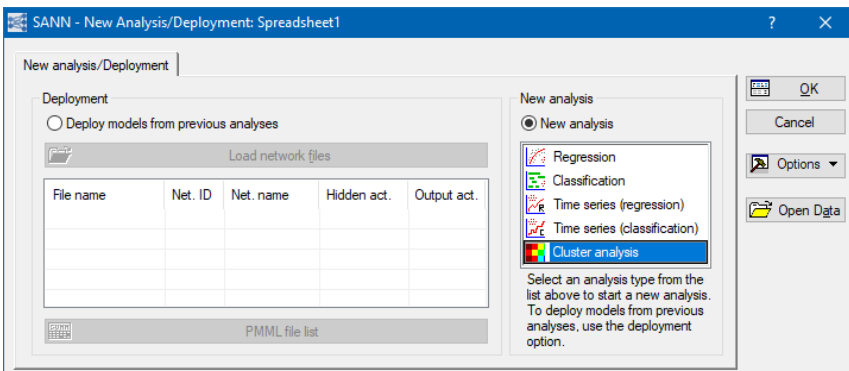


Рисунок 2.3 – Вибір типу аналізу

У вікні, що відкрилось, необхідно натиснути кнопку *Variables*. У вікні, що відкрилось натиснути *Select all* над полем *Categorical inputs* та натиснути кнопку ОК (рис. 2.4).

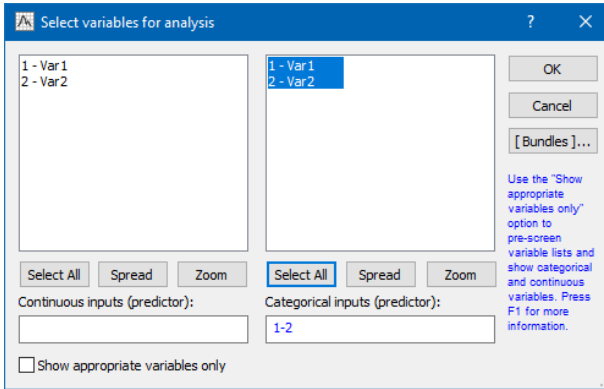


Рисунок 2.4 – Вибір значень для навчання

У вікні *Data selection* ще раз натиснути ОК. У вікні *Custom Neural Network* обрати висоту сітки 2, ширину – 1. Натиснути *Train* для навчання мережі (рис. 2.5).

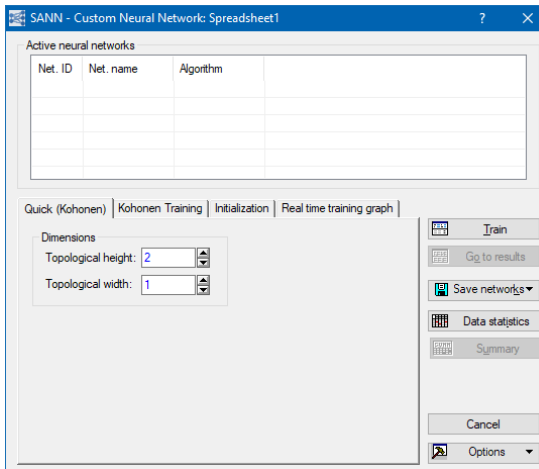


Рисунок 2.5 – Вибір розмірності сітки

Далі необхідно натиснути перейти на вкладку *Kohonen graph* та натиснути кнопку *Select All* (рис. 2.6). За графіком видно, що мережа віднесла по п'ять значень вибірки в кожну категорію.

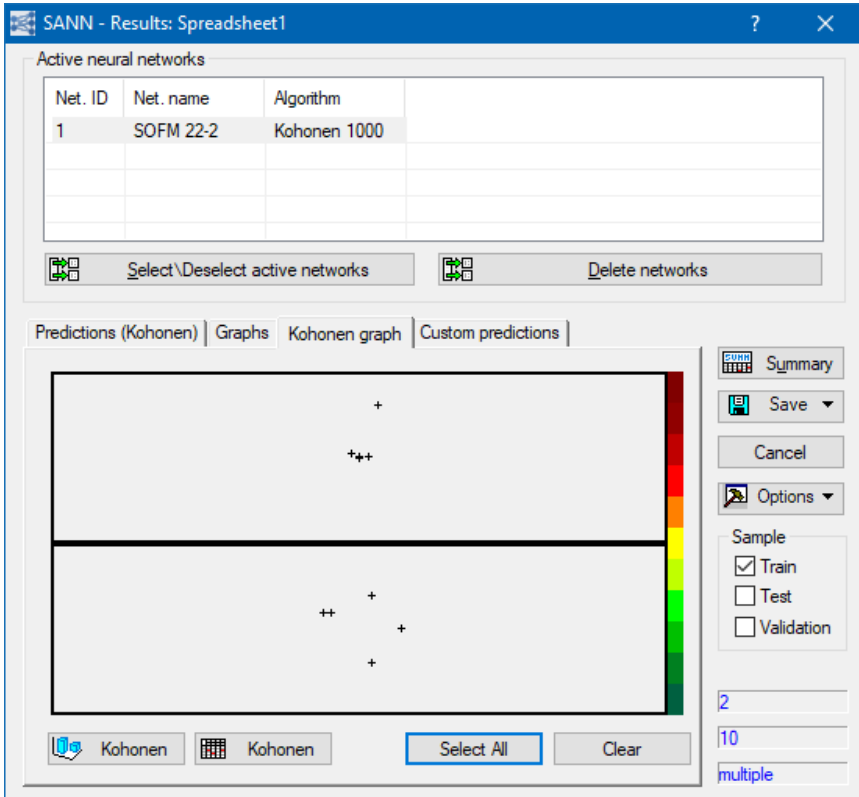


Рисунок 2.6 – Результат навчання мережі Кохонена

Наступним кроком необхідно натиснути кнопку *Kohonen* (другу зліва) та переглянути які значення в яку категорію помістила мережа, а також значення функцій активації в змагальному процесі навчання мережі.

Мережа віднесла вдалі 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.

2.3 Вирішення задачі кластеризації валютного курсу

Необхідно задати змінну – значення курсу покупки та продажу гривні до долара США / Євро (домножить число до цілого) за півроку з кроком в один тиждень. Виконати кластеризацію вхідних даних за допомогою карт Кохонена.

2.4 Завдання до виконання

Виконати приклади 2.2, 2.3. Продемонструвати результати виконання викладачу.

За результатами виконання завдань слід зробити висновки та навести їх у звіті.

2.5 Міст звіту

Звіт повинен мати такі складові:

- титульний лист;
- мета роботи;
- результат виконання завдань;
- відповіді на контрольні питання.

2.6 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 ЛАБОРАТОРНАЯ РОБОТА №3 РОЗПІЗНАВАННЯ ОБРАЗІВ

Мета роботи: навчитися використовувати апарат нейронних мереж для розпізнавання літер та цифр.

3.1 Теоретичні відомості

Стандартне завдання в машинному і комп'ютерному зорі і обробці зображень – це виявлення характерних об'єктів, будь-яких характеристик або дій на відеоданих. Зір людини здатний впоратися з ним без особливих навантажень і практично при будь-яких умовах. При цьому, варто зазначити, що відсоток розпізнавання об'єктів у нього буде достатньо високий. Однак в області комп'ютерного зору розпізнавання випадкових об'єктів в випадкових ситуаціях залишається невирішеною проблемою, з якою ще належить розібратися.

Безумовно, вже в наявності присутні різні методи, які дозволяють знаходити об'єкти і справляються з цим успішно. Але вони працюють тільки для конкретних предметів: прості геометричні об'єкти, різного виду символи, людські обличчя і автомобілі. При цьому їм необхідні певні умови для якісного розпізнавання. Як правило, це стосується освітлення, фону і розташування об'єкта щодо детектора. подібні проблеми широко висвітлені в літературі.

3.2 Задача на розпізнавання літер

Необхідно написати власну програму розпізнавання літер від А до D у середовищі *MATLAB*.

Для цього, у середовищі *MATLAB* необхідно створити новий скрипт (Home – New Script) та ввести код, що наведено у лістингу 3.1.

Лістинг 3.1 – Створення та відображення бази даних літер

```
A1=[1 -1 -1 -1 1;-1 1 1 1 -1;-1 -1 -1 -1 -1;-1 1 1 1 -1;-1 1 1 1 -1];
B1=[-1 -1 -1 -1 1;-1 1 1 1 -1;-1 -1 -1 -1 1;-1 1 1 1 -1;-1 -1 -1 -1 1];
C1=[-1 -1 -1 -1 -1;-1 1 1 1 1;-1 1 1 1 1;-1 1 1 1 1;-1 -1 -1 -1 -1];
D1=[-1 -1 -1 1 1;-1 1 1 -1 1;-1 1 1 -1 1;-1 1 1 -1 1;-1 -1 -1 1 1];
% Display images of stored patterns
hold on;
```

```
figure(1);
subplot(221);
imagesc(A1);
title('Stored A');
subplot(222);
imagesc(B1);
title('Stored B');
subplot(223);
imagesc(C1);
title('Stored C');
subplot(224);
imagesc(D1);
title('Stored D');
```

Після чого, необхідно зберегти файл та натиснути кнопку *Run*. Якщо використовувати MATLAB R2022a, то після натиснення кнопки *Run* у вікні, що відкриється (рис. 3.1) обрати Add to Path.

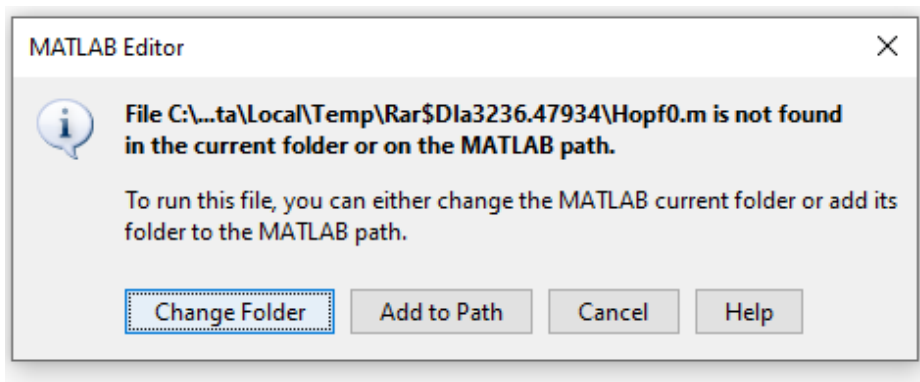


Рисунок 3.1 – Вікно додаткових налаштувань

Результатом буде відображення введених чотирьох літер (рис.3.2).

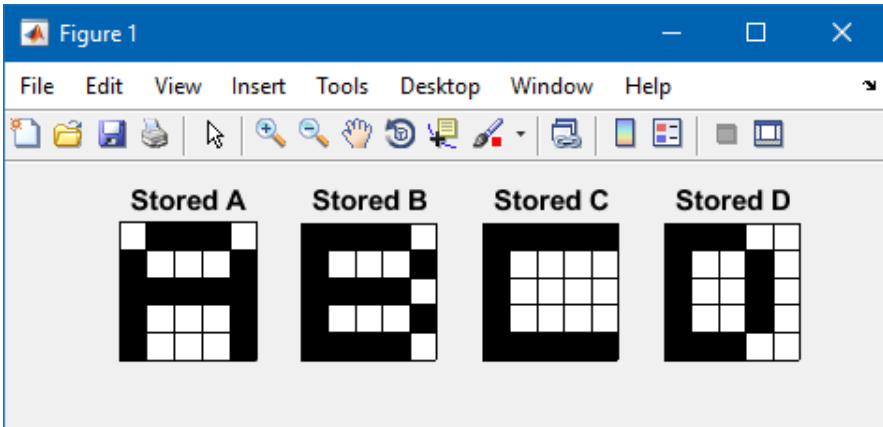


Рисунок 3.2 – Відображення літер

Далі необхідно доповнити скрипт для розпізнавання образу, ввівши код, наведений у лістингу 3.2

Лістинг 3.2 – Розпізнавання образу літери

```
clc
clear
% Initialize equilibrium characters
A1=[1 -1 -1 -1 1;-1 1 1 1 -1;-1 -1 -1 -1 -1;-1 1 1 1 -1;-1 1 1 1 -1];
B1=[-1 -1 -1 -1 1;-1 1 1 1 -1;-1 -1 -1 -1 1;-1 1 1 1 -1;-1 -1 -1 -1 1];
C1=[-1 -1 -1 -1 -1;-1 1 1 1 1;-1 1 1 1 1;-1 1 1 1 1;-1 -1 -1 -1 -1];
D1=[-1 -1 -1 1 1;-1 1 1 -1 1;-1 1 1 -1 1;-1 1 1 -1 1;-1 -1 -1 1 1];
% Display equilibrium characters
hold on;
figure(1);
subplot(221);
imagesc(A1);
title('Stored A');
subplot(222);
imagesc(B1);
title('Stored B');
subplot(223);
imagesc(C1);
title('Stored C');
subplot(224);
imagesc(D1);
title('Stored D');
% Initialize matrices to be used in calculations
```

```

V = [A1(:) B1(:) C1(:) D1(:)];
alpha = 0.8;
iter = 10;
% Calculate weight matrix
W = V*V';
W = W - diag(diag(W));
% Select integer initial matrix and reshape to column vector
x = sign(2*rand(5,5)-1);
x = reshape(x,25,1);
% Iterate calculations to update and display matrix
for k = 1:iter
    figure(2)
    x = reshape(x,5,5);
    subplot(3,5,k);
    abcd(x);
    x = reshape(x,25,1);
    xtemp = alpha*W*x;
    for j = 1:25
        if xtemp(j)>0
            x(j) = 1;
        else
            x(j)=-1;
        end
    end
end
end

```

Результатом запуску коду буде формування випадкового набору символів розмірністю 5 на 5 елементів. Нейронна мережа, на основі отриманої бази, намагатиметься розпізнати, до якої з літер найбільш схожий сформований набір клітинок (рис. 3.3).

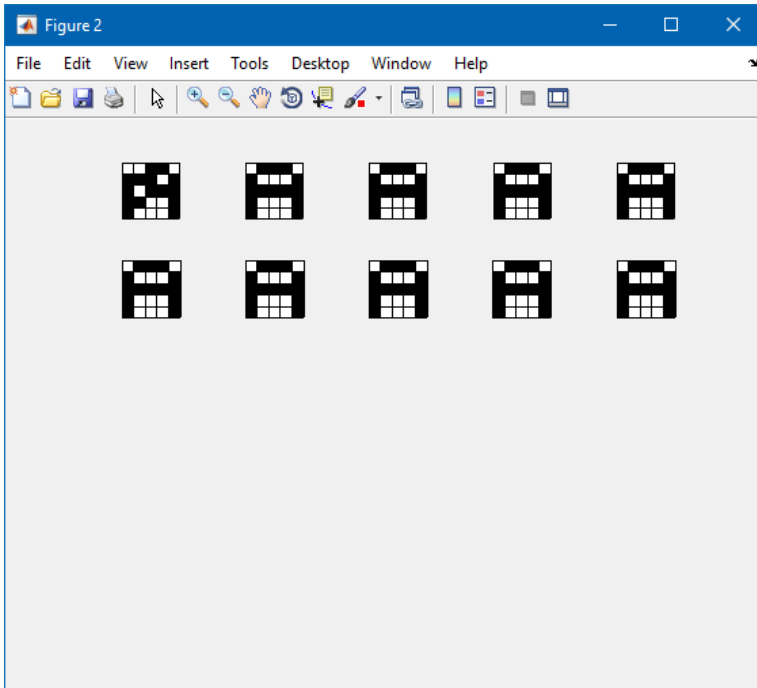


Рисунок 3.3 – Результат розпізнавання образу літери

Як видно, уже на другій ітерації образу, нейронна мережа розпізнала в ній літеру А. Ви можете проєкспериментувати з новим набором значень, які будуть випадково генеруватись під час кожного запуску.

3.3 Завдання до виконання

Виконати приклади 3.2. Додатковим завданням є створення програми для розпізнавання образів цифр від 0 до 9. Продемонструвати результати виконання викладачу.

За результатами виконання завдань слід зробити висновки та навести їх у звіті.

3.4Зміст звіту

Звіт повинен мати такі складові:

- титульний лист;
- мета роботи;
- результат виконання завдань;
- відповіді на контрольні питання.

3.5Контрольні питання

1. Опишіть методи розпізнавання образів.
2. Які галузі використання кожного з методів розпізнавання образів?
3. Які основні напрями в розпізнаванні образів?
4. Які конкретні задачі успішно вирішуються в галузі розпізнавання образів?

4 ЛАБОРАТОРНА РОБОТА №4 РОЗПІЗНАВАННЯ ОБЛИЧЧЯ

Мета роботи: навчитися створювати програми для розпізнавання обличчя мовою Python за допомогою бібліотеки OpenCV.

4.1 Теоретичні відомості

Автоматична ідентифікація особистості людини по його зображенню на фотографії або в відеопотоці має широке комерційне та наукове застосування. Дана тематика з'явилася на початку 80-х років, однак, її бурхливий розвиток почалося в 90-х, після створення нових технологій в сфері обробки зображень та обчислювальних машин. Дана технологія має особливий інтерес у зв'язку з тим, що може відбуватися безконтактно.

В даний час зріс інтерес до різних методів ідентифікації осіб. Для вирішення цього завдання необхідно виконати два етапи: виявити особа на фотографії, виділити і розпізнати його.

При всьому різноманітті різних алгоритмів і методів розпізнавання зображень, типовий метод розпізнавання складається з трьох компонентів:

- перетворення вихідного зображення в початкове уявлення (може включати в себе як попередню обробку, так і математичні перетворення, наприклад обчислення головних компонент);

- виділення ключових характеристик (наприклад береться перші n головних компонент або коефіцієнтів дискретного косинусного перетворення);

- механізм класифікації (моделювання): кластерна модель, метрика, нейронна мережа тощо.

Крім цього, побудова методу розпізнавання спирається на апіорну інформацію про предметну область (в даному випадку - характеристики особи людини), і коригується експериментальної інформацією, що з'являється по ходу розробки методу.

4.2 Налаштування програмного середовища

Для початку необхідно завантажити та інсталивати IDE PyChar. Після чого необхідно створити новий проект. Далі необхідно інсталивати бібліотеку opencv. Для цього, у терміналі програми вводимо команду:

```
pip install opencv-python
```

Далі у вікні програми імпортуємо цю бібліотеку командою:

```
import cv2
```

Враховуючи те, що opencv – безкоштовна бібліотека з відкритим кодом, то багато класифікаторів можна знайти на github. Переходимо за посиланням <https://github.com/opencv/opencv> та відкриваємо папку data, там же обираємо вже існуючі класифікатори.

Для нашого завдання оберіть з папки haarcascade класифікатор визначення обличчя:

```
haarcascade_frontalface_default.xml
```

Далі необхідно завантажити цей файл та перенести його до поточного проекту за шляхом:

```
Lib\site-packages\cv2\data
```

Після чого, необхідно створити об'єкт у Python (у дужках прописано шлях до обраного файлу):

```
face_cascade_db =  
cv2.CascadeClassifier(cv2.data.haarcascades+"haarcascade_frontalface_  
default.xml")
```

Далі необхідно завантажити до проекту будь-яке фото з обличчям у поточну директорію проекту.

Після чого, необхідно створити об'єкт для цього зображення:

```
img = cv2.imread("img.jpg")
```

Для того, щоб наше зображення правильно розпізнало існуючим класифікатором, необхідно його перетворити у відтінки сірого:

```
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Виведемо це зображення на екран. Для цього введемо команду `imshow` (назва вікна, назва зображення) та затримку `cv2.waitKey()`:

```
cv2.imshow('rez', img_gray)
cv2.waitKey()
```

Та запускаємо проект на виконання (ПКМ - Run). У результаті ми побачимо зображення у сірих відтінках (рис. 4.1).

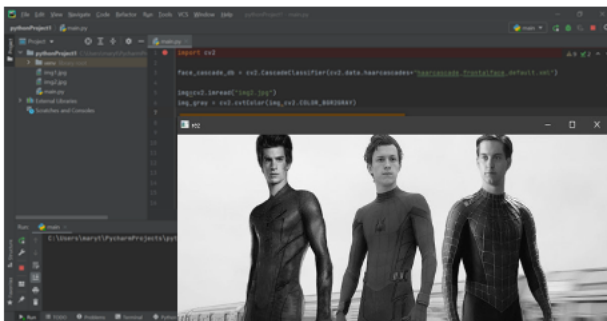


Рисунок 4.1 – Виведення зображення у сірих відтінках

Далі створимо об'єкт `faces`, у який і запишемо розпізнані обличчя за допомогою класифікатору:

```
faces = face_cascade_db.detectMultiScale(img_gray, 1.1, 19)
```

За допомогою об'єкту `face` ми отримувемо перелік знайдених координат на цьому зображенні.

Далі виведемо квадрат на знайдених обличчях та не чорно-біле, а кольорове, зображення. Повний код програми наведено у лістингу 4.1

Лістинг 4.1 – Код програми для розпізнавання обличч на фотографії

```
import cv2

face_cascade_db =
cv2.CascadeClassifier(cv2.data.haarcascades+"haarcascade_fronta
lface_default.xml")

img = cv2.imread("family.jpg")
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_cascade_db.detectMultiScale(img_gray, 1.1, 19)
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

cv2.imshow('rez', img)
cv2.waitKey()
```

Після чого запустимо проект та у результаті отримаємо програму, яка розпізнає обличчя на фотографії (рис. 4.2).

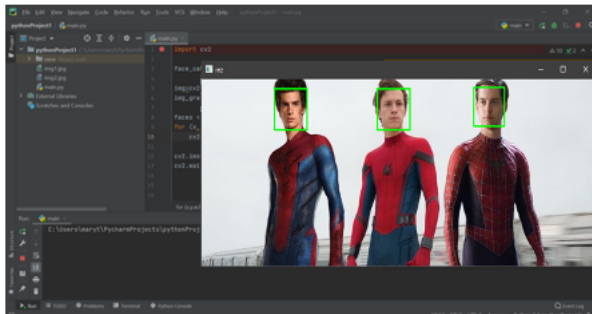


Рисунок 4.2 – Результат розпізнавання обличчя на фотографії

4.3 Розпізнавання обличчя з відеопотоку

Необхідно написати програму, яка здатна розпізнавати обличчя з потоку веб-камери.

Для цього створимо новий файл у проекті. Як джерело інформації введемо потік з веб-камери:

```
cap = cv2.VideoCapture(0)
Далі у циклі отримаємо зображення з веб-камери:

while True:
    success, img = cap.read()
```

```
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_cascade_db.detectMultiScale(img_gray, 1.1, 19)
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
```

Функція для виходу з циклу по натисканню кнопки «q»:

```
cv2.imshow('rez', img)
if cv2.waitKey() & 0xff == ord('q'):
    break
```

Повний код програми наведено у лістингу 4.2

Лістинг 4.2 – Код програми для розпізнавання обличч з потоку веб-камери

```
import cv2

face_cascade_db =
cv2.CascadeClassifier(cv2.data.haarcascades+"haarcascade_fronta
lface_default.xml")

cap = cv2.VideoCapture(0)
while True:
    success, img = cap.read()
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    faces = face_cascade_db.detectMultiScale(img_gray, 1.1, 19)
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

    cv2.imshow('rez', img)
    if cv2.waitKey() & 0xff == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

У результаті, по чергово будуть відображатись кадри з веб-камери з обведеним обличчям на ньому у разі успіху (рис. 4.3).

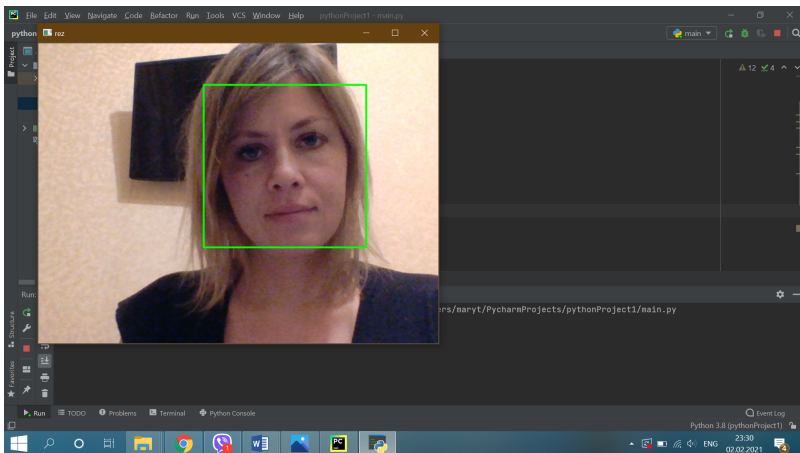


Рисунок 4.3 – Результат розпізнавання обличчя з потоку веб-камери

Як бачимо з результатів, програмою зчитано інформацію з потоку веб-камери, розпізнано на ній обличчя та обведено його зеленою рамкою.

4.4 Завдання до виконання

Виконати приклади 4.2, 4.3. Продемонструвати результати виконання викладачу.

Додатковим (необов'язковим) завданням є :

- створення програми для розпізнавання обличчя із записаного відео.

Продемонструвати результати виконання викладачу.

За результатами виконання завдань слід зробити висновки та навести їх у звіті.

4.5 Зміст звіту

Звіт повинен мати такі складові:

- титульний лист;
- мета роботи;
- результат виконання завдань;
- відповіді на контрольні питання.

4.6 Контрольні запитання

1. Дуже стисло опишіть програмні продукти, які існують для розпізнавання обличь.
2. Стисло опишіть бібліотеки Python, які призначені для розпізнавання.
3. Опишіть бібліотеки Python для роботи з нейронними мережами.
4. Особливості згорткових нейронних мереж.

5 ЛАБОРАТОРНА РОБОТА №5

ЗАСТОСУВАННЯ ГЕНЕТИЧНИХ АЛГОРИТМІВ

Мета роботи: навчитися застосовувати генетичні алгоритми на практиці.

5.1 Теоретичні 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 три механізми природної еволюції:

–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;

–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);

– 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:

- оператор репродукції (ОР);
- оператор схрещування (кросинговеру, ОК);
- 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 Задача для мінімізації функції

Необхідно мінімізувати функцію однієї змінної:

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

Для цього, у середовищі *MATLAB* необхідно створити два скрипти. У першому міститься описана вище функція (лістинг 5.1), другий містить алгоритм для підрахунку фітнес-функції, пошуку найкращої особи та середньої відстані між особинами (лістинг 5.2 та рис.).

Лістинг 5.1 – Функція мінімізації

```
function y = ex2 (x)
y=8*x-16-12*(nthroot((x+4)^2, 3));
```

Лістинг 5.2 – Функція генетичного алгоритму

```
function [X,FVAL,REASON,OUTPUT,POPULATION,SCORES] =
ex2_func
fitnessFunction = @ex2;
nvars = 1;
options = optimoptions(@ga,'PopInitRange',[-4 ; 1]);
options = optimoptions(options,'PopulationSize',10);
options =
optimoptions(options,'MutationFcn',{@mutationgaussian 1
1});
options = optimoptions(options,'Display','off');
options =
optimoptions(options,'PlotFcns',{@gaplotbestf,
@gaplotbestindiv, @gaplotdistance});
[X,FVAL,REASON,OUTPUT,POPULATION,SCORES] =
ga(fitnessFunction,nvars,options);
```

Далі необхідно запусити створені скрипти. Для цього скористайтесь інструкціями нижче.

Щоб використовувати генетичний алгоритм у командному рядку, викличте функцію генетичного алгоритму `ga` із синтаксисом

`[x,fval] = ga(@fitnessfun, nvars, options)`

де

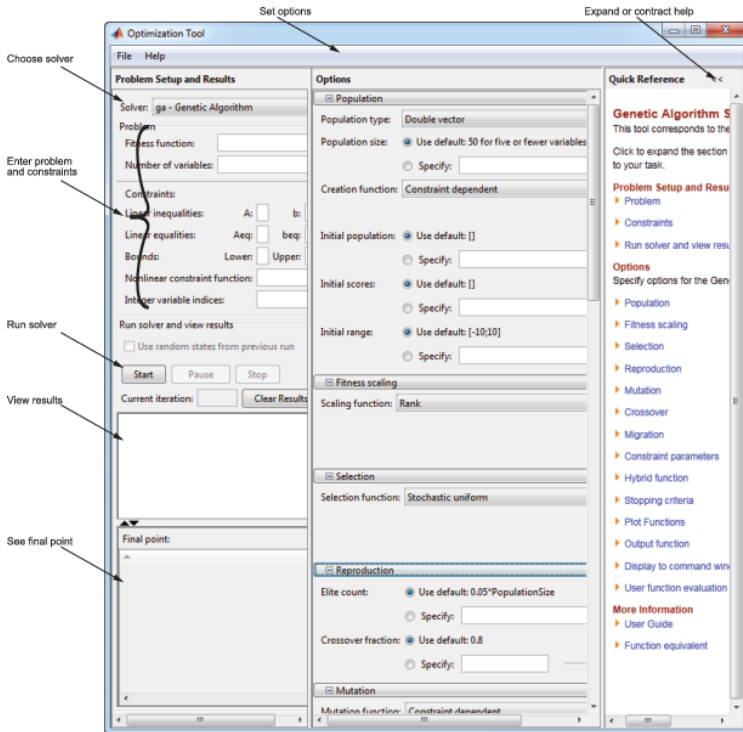
- `@fitnessfun` — це обробник для фітнес-функції.
- `nvars` — кількість незалежних змінних для фітнес-функції.
- `options` містить параметри для генетичного алгоритму. Якщо ви не передаєте цей аргумент, `ga` використовує параметри за замовчуванням.

Результати мають:

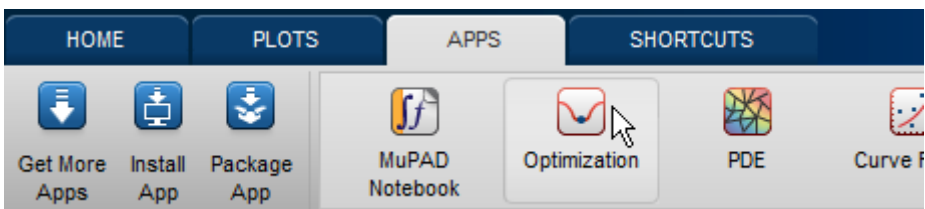
- `x` — Точка, в якій досягається кінцеве значення
- `fval` — Кінцеве значення функції придатності

Щоб відкрити програму оптимізації, введіть `optimtool('ga')`

у командному рядку або введіть `optimtool`, а потім оберіть `ga` в меню Solver.



Ви також можете запустити інструмент безпосередньо із вкладки програми MATLAB.



Щоб використовувати програмне середовище оптимізації, ви повинні спочатку ввести таку інформацію:

- Фітнес-функцію — цільова функція, яку потрібно мінімізувати. Введіть фітнес-функцію у формі `@fitnessfun`, де `fitnessfun.m` — це файл, який обчислює фітнес-функцію. Знак `@` створює маркер функції для `fitnessfun`.

• Кількість змінних — довжина вхідного вектора для відповідної функції.

Ви можете ввести обмеження або функцію нелінійного обмеження для проблеми на панелі «Обмеження». Якщо проблема необмежена, залиште ці поля порожніми.

Щоб запустити генетичний алгоритм, натисніть кнопку Start. Інструмент відображає результати оптимізації на панелі «Run solver and view results».

Ви можете змінити параметри генетичного алгоритму на панелі параметрів. Щоб переглянути параметри в одній із категорій, перелічених на панелі, клацніть знак + поруч із нею.

Для перевірки результату підрахунку фітнес-функції, пошуку найкращої особи та середньої відстані між особинами (рис. 5.1).

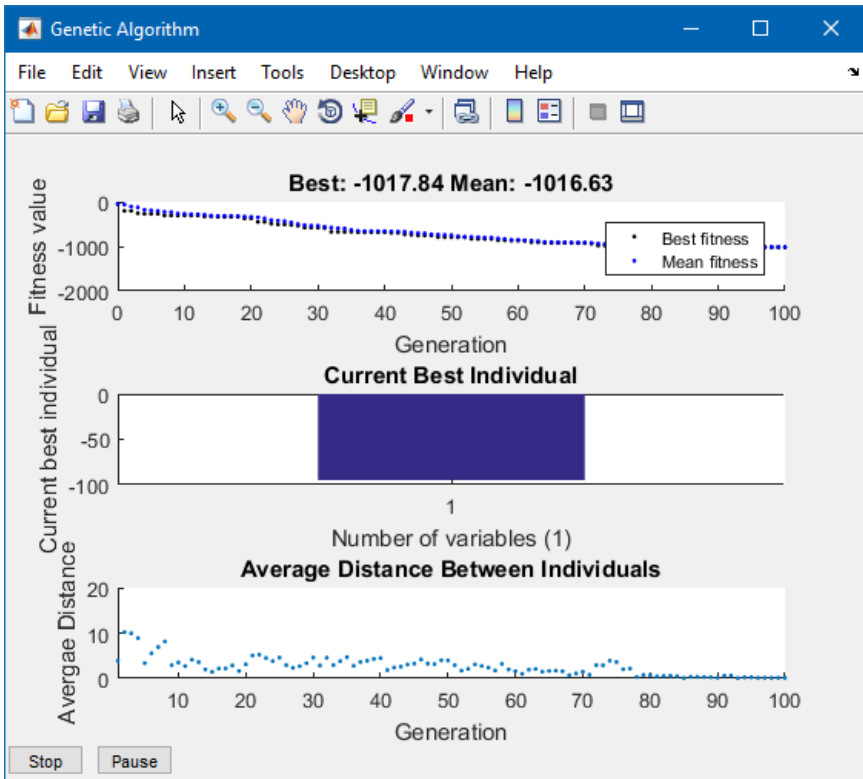


Рисунок 5.1 – Результат роботи генетичного алгоритму

Перший рисунок відображає зміну значення цільової функції. На другому рисунку зображена найкраща особина. Третій рисунок відповідає зсуванням відстані між особинами в поколіннях. Особини стають однакові (хеммінгова відстань = 0) в останніх 20 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.

5.3 Задача для максимізації функції

Необхідно максимізувати функцію двох змінних:

$$z(x, y) = \exp(-x^2 - y^2) + \sin(x + y)$$

Додаток з генетичних алгоритмів 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.

Для цього, у середовищі *MATLAB* необхідно знову створити два скрипти. У першому міститься описана вище функція (лістинг 5.3), другий містить алгоритм для пошуку фітнес функції, найкращої особини та середньої відстані між особинами (лістинг 5.4).

Лістинг 5.3 – Функція максимізації

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

Лістинг 5.4 – Функція генетичного алгоритму

```
function [X, FVAL, REASON, OUTPUT, POPULATION, SCORES] =
ex13_func
    fitnessFunction = @ex13;
    nvars = 2;
    options = optimoptions(@ga, 'PopInitRange', [-1 ; 3]);
    options = optimoptions(options, 'PopulationSize', 10);
    options =
optimoptions(options, 'MutationFcn', {@mutationgaussian 1 1});
    options = optimoptions(options, 'Display', 'off');
    options = optimoptions(options, 'PlotFcns', {@gaplotbestf,
@gaplotbestindiv, @gaplotdistance});
```

```
[X,FVAL,REASON,OUTPUT,POPULATION,SCORES] =  
ga(fitnessFunction,nvars,options);
```

Перевіряємо результат підрахунку фітнес функції, пошуку двох найкращих особин та середньої відстані між особинами (рис. 5.2).

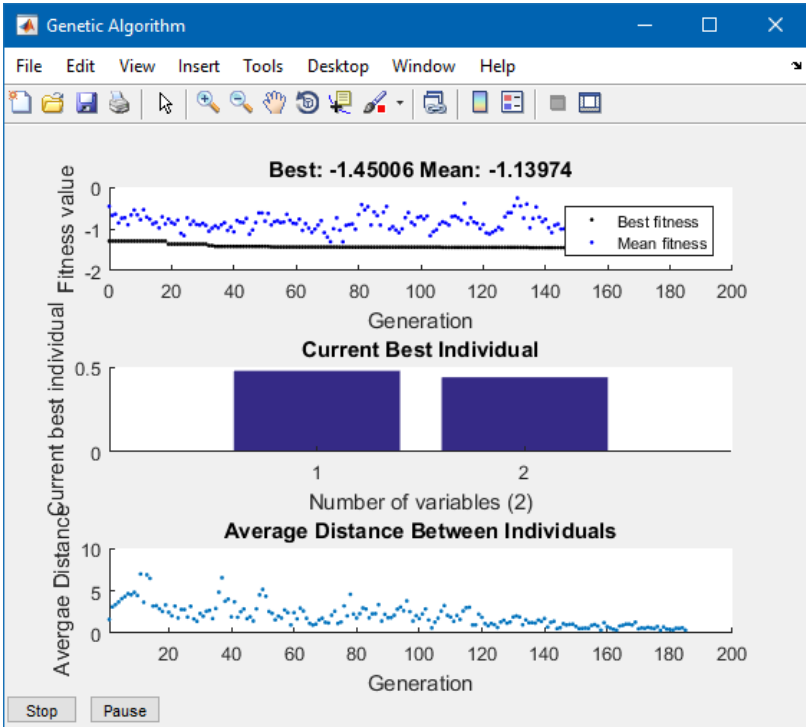


Рисунок 5.2 – Результат роботи генетичного алгоритму

5.4 Завдання до виконання

5.4.1 Виконати приклади 6.2, 6.3. Продемонструвати результати виконання викладачу.

5.4.2 Додатковими завданнями є :

– знайти максимум функції:

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

– знайти мінімум функції:

$$z(x, y) = (y - 3) * \exp(-x^2 - y^2)$$

Продемонструвати результати виконання викладачу.

5.4.3 За результатами виконання завдань слід зробити висновки та навести їх у звіті.

5.5 Міст звіту

Звіт повинен мати такі складові:

- титульний лист;
- мета роботи;
- результат виконання завдань;
- відповіді на контрольні питання.

5.6 Контрольні запитання

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».

ЛІТЕРАТУРА

1. Шаховська Н.Б. Системи штучного інтелекту / Шаховська Н.Б., Р.М. Камінський, О.Б. Вовк - Львів: Львівська політехніка, 2018. - 392 с.
2. Офіційна сторінка підтримки користувачів Matlab [Електронний ресурс] / Режим доступу: <https://www.mathworks.com/help/matlab/index.html>
3. Офіційна сторінка підтримки користувачів Statistica [Електронний ресурс] / Режим доступу: <https://docs.tibco.com/data-science/textbook>
4. Wilson H. Artificial intelligence. / H Wilson. - Grey House Publishing. 2018.