Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет Інформатики та Обчислювальної Техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №6

з дисципліни «Методи та технології штучного інтелекту»

на тему

«Нейро-нечітке моделювання»

Виконала:
студентка групи ІС-12
Павлова Софія

Викладач:
Шимкович В. М.

Київ – 2023

1. Постановка задачі

Мета: Отримання і закріплення знань про методи моделювання та принципи функціонування нейронечітких систем, а також формування практичних навичок з конструювання нейронечітких мереж.

Завдання:

1. Сформулювати завдання в галузі обчислювальної техніки, для вирішення якої було б обґрунтовано застосування гібридної нейронечіткої мережі.
2. Сформувати вибірку для навчання гібридної нейронної мережі.
3. Згенерувати і візуалізувати структуру гібридної нейронної мережі.
4. Навчити гібридну нейронну мережу, при цьому задати і обґрунтувати параметри її навчання.
5. Виконати перевірку адекватності побудованої нечіткої моделі гібридної мережі.

2. Виконання

Постановка задачі:

Сформулюємо задачу в галузі машинного навчання наступним чином:

Здача – Прогнозування курсу валюти.

Необхідно прогнозувати курс валюти (EUR до USD) за допомогою гібридної нейронної мережі.

Задача полягає у створенні та тренуванні моделі, яка може адекватно визначати залежність між показниками відкриття (*'BO'* або *'Open'*), високої (*'BH'* або *'High'*), низької (*'BL'* або *'Low'*) ціни та кінцевою (*'BC'* або *'Close'*) ціною валютного обміну.

Набір даних:

Для нашої задачі використаємо публічний датасет EUR USD Forex Pair Historical Data (2002 - 2020).

Date	Time	BO	BH	BL	BC
2005-05-02	00:00	1.2852	1.2852	1.284	1.2844
2005-05-02	01:00	1.2844	1.2848	1.2839	1.2842
2005-05-02	02:00	1.2843	1.2854	1.2841	1.2851
2005-05-02	03:00	1.2851	1.2859	1.285	1.2851
2005-05-02	04:00	1.2852	1.2859	1.2849	1.2855
2005-05-02	05:00	1.2854	1.2858	1.2853	1.2854
2005-05-02	06:00	1.2854	1.286	1.2852	1.28585

Рисунок 1 – Вигляд датасету

Датасет містить наступні стопці:

Date – **Дата**:

Містить дані з *2005-05-02* (2 травня 2005) по *2020-04-29* (29 квітня 2020).

Time – **Час**:

Година, в якій була виміряна ціна. Містить значення з *00:00* по *23:00*.

BO – **Open**:

Ціна початкової пропозиції.

BH – **High**:

Найвища ціна пропозиції за годину.

BL – **Low**:

Найнижча ціна пропозиції за годину.

BC – **Close**:

Ціна закриття пропозиції.

Створення гібридної нейронної мережі:

Завантаження даних

Оскільки набір даних містить *93 084* зразка – що є завеликою кількістю для навчання нейромережі, використаємо перші *3794* зразка, завантаживши дані до *2006-02-01* (1 лютого 2006).

Завантажимо наш датасет та розділимо його на тренувальний і тестовий у співвідношенні 80:20. Відобразимо датасет у вигляді графіка та таблиць DataFrame.

За ознаки візьмемо стовпці **BO (Open)**, **BH (High)**, **BL (Low)**, а за мітки – **BC (Close)**.

Лістинг:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import anfis
import membership.membershipfunction
import warnings
# Ігнорувати RuntimeWarning: divide by zero encountered in scalar divide
warnings.filterwarnings("ignore", category=RuntimeWarning)
# Функція завантаження даних
def load_data(file_name, end_date):
    # Завантаження даних з CSV-файлу
    dataset = pd.read_csv(file_name)
    dataset['Date'] = pd.to_datetime(dataset['Date'])
    dataset = dataset[dataset['Date'] <= end_date]

    # Розбиття тестових даних
    X_test = dataset[['BO', 'BH', 'BL']].tail(1).to_numpy()
    y_test = dataset['BC'].tail(1).to_numpy()

    # Розбиття тренувальних даних
    dataset.drop(dataset.index[-1], inplace=True)
    X_train = dataset[['BO', 'BH', 'BL']].to_numpy()
    y_train = dataset['BC'].to_numpy()

    # Візуалізація даних на графіках
    plot_data(dataset)

    return X_train, X_test, y_train, y_test

# Головні виклики
if __name__ == "__main__":
    file_name = 'eur_usd_hour.csv'
    end_date = '2006-02-01'

    # Завантаження даних
    X_train, X_test, y_train, y_test = load_data(file_name, end_date)

    # Візуалізація вхідних даних
    df_X_train = pd.DataFrame(X_train, columns=['Open', 'High', 'Low'])
    df_y_train = pd.DataFrame(y_train, columns=['Close'])
    df_train = pd.concat([df_X_train, df_y_train], axis=1)
    print('\nТренувальні дані:')
    print(df_train)

    df_X_test = pd.DataFrame(X_test, columns=['Open', 'High', 'Low'])
    df_y_test = pd.DataFrame(y_test, columns=['Close'])
    df_test = pd.concat([df_X_test, df_y_test], axis=1)
    print('\nТестові дані:')
    print(df_test)
```

Результат:

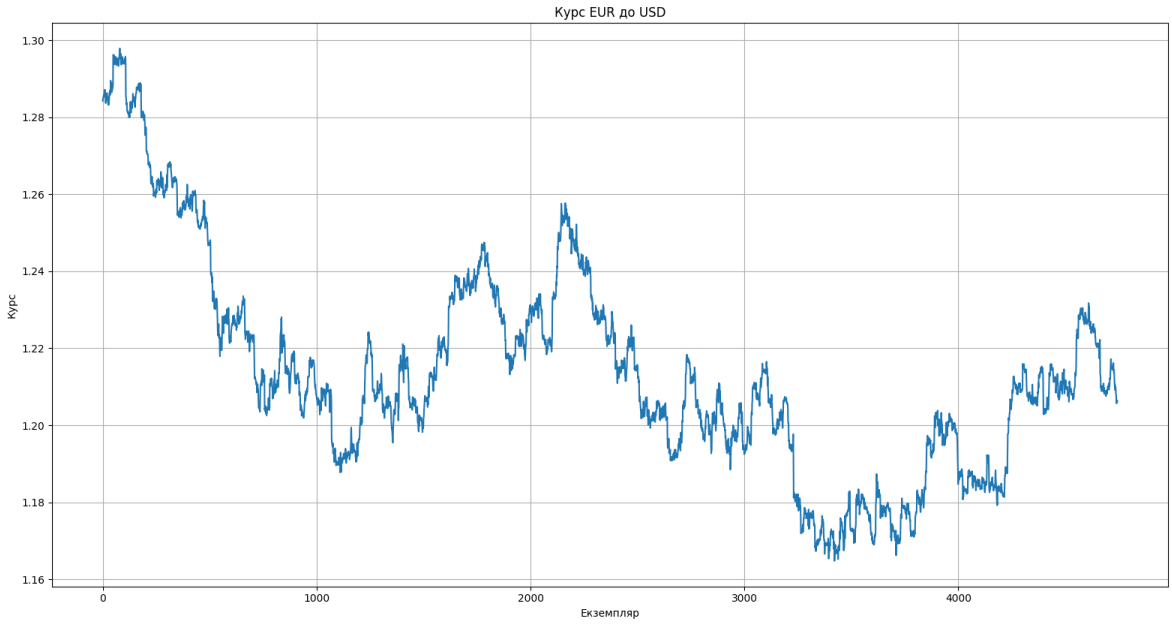

Рисунок 2 – Візуалізація датасету

```
Тренувальні дані:
         Open     High      Low    Close
0     1.28520  1.28520  1.28400  1.28440
1     1.28440  1.28480  1.28390  1.28420
2     1.28430  1.28540  1.28410  1.28510
3     1.28510  1.28590  1.28500  1.28510
4     1.28520  1.28590  1.28490  1.28550
...       ...      ...      ...      ...
4738  1.20813  1.20843  1.20713  1.20773
4739  1.20778  1.20783  1.20503  1.20573
4740  1.20583  1.20643  1.20553  1.20610
4741  1.20610  1.20683  1.20470  1.20643
4742  1.20643  1.20673  1.20553  1.20583

[4743 rows x 4 columns]

Тестові дані:
      Open     High      Low    Close
0  1.20563  1.20673  1.20553  1.20633
```

Рисунок 3 – Розбиття датасету на тренувальний та тестовий

Архітектура нейромережі

Розглянемо архітектуру нейромережі *ANFIS*.

Бібліотеку ANFISS взято з [Github репозиторію](#).

1. **Шар приналежності (Antecedent Layer):**

   Кількість нейронів у цьому шарі дорівнює загальній *кількості функцій приналежності*, які використовуються для моделювання вхідних даних.

   Так, як ми маємо **9 функцій приналежності**:

   Лістинг:

```
# Гаусівські функції приналежності
def gaussian_mf(X_train):
    mf = [
        [
            ['gaussmf', {'mean': np.mean(X_train[:, 0] - 1000), 'sigma':
np.std(X_train[:, 0])}],
            ['gaussmf', {'mean': np.mean(X_train[:, 0]), 'sigma': np.std(X_train[:,
0])}],
            ['gaussmf', {'mean': np.mean(X_train[:, 0] + 1000), 'sigma':
np.std(X_train[:, 0])}],
        ],
        [
            ['gaussmf', {'mean': np.mean(X_train[:, 1] - 1000), 'sigma':
np.std(X_train[:, 1])}],
            ['gaussmf', {'mean': np.mean(X_train[:, 1]), 'sigma': np.std(X_train[:,
1])}],
            ['gaussmf', {'mean': np.mean(X_train[:, 1] + 1000), 'sigma':
np.std(X_train[:, 1])}],
        ],
        [
            ['gaussmf', {'mean': np.mean(X_train[:, 2] - 1000), 'sigma':
np.std(X_train[:, 2])}],
            ['gaussmf', {'mean': np.mean(X_train[:, 2]), 'sigma': np.std(X_train[:,
2])}],
            ['gaussmf', {'mean': np.mean(X_train[:, 2] + 1000), 'sigma':
np.std(X_train[:, 2])}],
        ],
    ]

    return mf
```

   Цей шар буде мати **9 нейронів**.

2. **Шар виводу (Consequent Layer):**

Кількість нейронів у цьому шарі визначається *комбінацією функцій належності для кожної змінної.*

Так, як ми використовумо **3 змінні**, і для кожної змінної – **3 функції** *(low, medium, high*). Тоді ми маємо на цьому шарі **3^3 = 27 нейронів**.

Навчання нейромережі:

Навчимо нейромережу на **5 епохах**.

За критерій оцінки точності мережі візьмемо стандартний для *ANFIS*.

Стандартний критерій оптимальності моделі в *ANFIS* базується на мінімізації **середньоквадратичної похибки** між виходом нейромережі та відомими вихідними значеннями.

Лістинг:

```
# Головні виклики
if __name__ == "__main__":
    [...]

    # Створення і навчання нечіткої нейромережі ANFIS
    print('\nНавчання:')
    mf = gaussian_mf(X_train)
    mfc = membership.membershipfunction.MemFuncs(mf)
    anf = anfis.ANFIS(X_train, y_train, mfc)
    anf.trainHybridJangOffLine(epochs=5)

    # Візуалізація графіка функції втрат та результатів навчання
    # Графік втрат
    anf.plotErrors()
    # Результати навчання
    anf.plotResults()
```

Результат:

```
Навчання:
current error: 0.0017589694126659676
current error: 0.0017589694126659676
current error: 0.0017589694126659676
current error: 0.0017589694126659676
```

Рисунок 5 – Навчання нейромережі

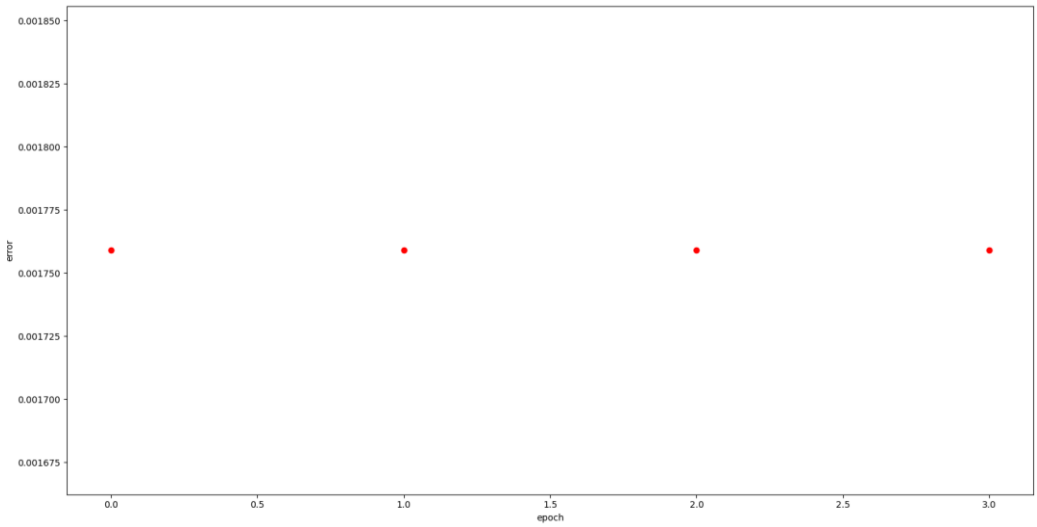

Рисунок 6 – Функція втрат

З результатів похибки навіть на 5 епохах бачимо, що значення не змінюються.

Отже можемо зробити припущення, що модель збіглась на 1 епосі.

Це можна пояснити стохастичністю оптимізаційного алгоритму ANFIS, оскільки параметри, що впливають на збіжність моделі такі, як *learning rate* та *початкові ваги* змінюються в алгоритмі динамічно.

Іншою ж причиною можуть бути особливості вхідних даних.

Але перевірити припущення про збіжність можна буде лише протестувавши модель, що ми й зробимо.

Передбачення:

Перевіримо працездатність навченої нейромережі на тестовому датасеті. Заодно й перевіримо наше припущення про збіжність моделі.

Лістинг:

```
# Головні виклики
if __name__ == "__main__":
    [...]

    # Передбачення та порівняння результатів
    y_pred = anfis.predict(anf, X_test)[0][0]
    y_real = y_test[0]
    df_combined = pd.DataFrame(np.column_stack((y_real, y_pred)), columns=['Real',
```

```
'Predicted'])

    print('\nРезультати передбачення:')
    print(df_combined)
```

Результат:

```
Результати передбачення:
      Real  Predicted
0  1.20633   1.206254
```

Рисунок 7 – Порівняння прогнозованих та реальних значень

Бачимо, що значення і графіки курсу EUR до USD доволі подібні. Отже наше припущення про збіжність нейромережі істинне. Як бачимо, нейромережа справляється з поставленою задачею.

Висновок:

Під час виконання лабораторної роботи було отримано теоретичні знання та практичні навички.

Досліджено застосування методів моделювання та принципів функціонування нейронечітких систем. Опановано засоби нейрочіткого моделювання нейромереж ANFIS. Сформульовано задачу, яка являє собою розроблення гібридної нейронної мережі, що вирішує завдання прогнозування курсу валют.

Обрано датасет з даними курсів EUR до USD.

Створено та навчено нейрочітку гібридну нейронну мережу з використанням бібліотеки ANFIS, що виконує поставлену задачу.

У результаті, створена нейромережа показала гарні результати в прогнозуванні значень показавши невелику похибку.