МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-наукового інституту атомної і теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

ЗВІТ

ДО ВИКОНАННЯ ПРАКТИЧНОГО ЗАВДАННЯ №1

з дисципліни

«МЕТОДОЛОГІЯ РОЗРОБКИ ІНТЕЛЕКТУАЛЬНИХ КОМП'ЮТЕРНИХ ПРОГРАМ »

Тема: «Штучні нейронні мережі. Моделювання формальних логічних функцій. Прогнозування часових рядів»

Варіант 16

КИЇВ 2025

Мета: Отримати початкові навички щодо створення штучних нейронних мереж, що здатні виконувати прості логічні функції, та нейронних мереж, що здатні прогнозувати часові ряди.

Опис дії алгоритму

На кожній з ітерацій кінцевою задачею є редагування синаптичних ваг так, щоб функція активації нейрона повертала прогнозовані значення часового ряду з найменшою сумарною помилкою.

Далі ми розглянемо приклад даного алгоритму зворотного поширення на прикладі першої епохи

Перш за все, згенеруємо випадкові значення від -1 до 1 синаптичних ваг. В даному випадку це:

w1 = 0.4837797265374466

w2 = 0.3713757490619345

w3 = 0.8016212947028836

Далі для кожного набору даних буде знаходитись зважена сума яка обчислюється за наступною формулою:

$$S_i = x_{i-3} \cdot w_1 + x_{i-2} \cdot w_2 + x_{i-1} \cdot w_3$$

Оскільки наша тренувальна вибірка буде мати 10 значень, то й кількість зважених сум буде 10:

S1 = 2.2653176514006814	S6 = 6.601817502617537
S2 = 6.622530916619654	S7 = 2.7813198049897765
S3 = 3.3236942738879205	S8 = 6.337875199887906
S4 = 7.159997032024997	S9 = 2.5014444751029012
S5 = 3.238728968751742	S10 = 6.3315107814866405

Після цього визначені зважені суми передаються у функцію сигмоїди, яка має наступний вигляд:

$$Y = \frac{10}{1 + e^{-S}}$$

Y1 = 9.059636395402002	Y6 = 9.98643943575469
Y2 = 9.986717063007193	Y7 = 9.416579944681535
Y3 = 9.652327789154077	Y8 = 9.982350654893459
Y4 = 9.992235464890271	Y9 = 9.2424302103226
Y5 = 9.62265985251544	Y10 = 9.982238169017565

Обчисливши ці значення, нам треба знайти значення похідної функції помилки формула якої зображена нижче:

$$Y' = \frac{10e^{-S}}{(1+e^{-S})^2}$$

отримані значення множимо на різницю прогнозованого значення та реального і отримаємо:

Δ1 = 2.776999094335373	Δ6 = 0.1271128087984916
Δ2 = 0.12040531450747106	Δ7 = 2.5307728366456677
Δ3 = 1.5578937802361132	Δ8 = 0.16653335863686414
Δ4 = 0.06844736951542348	Δ9 = 3.1455034550730567
Δ5 = 1.7983186612055735	Δ10 = 0.159080319353938

Враховуючи попередні розрахунки, обчислюємо, наскільки потрібно скоригувати кожен із вагових коефіцієнтів для зменшення помилки. При цьому використовуємо коефіцієнт навчання, який дорівнює 0.01 і визначає швидкість оновлення ваг.

$\Delta w_{1,1}$ = -0.016106594747145164	$\Delta w_{1,2}$ = -0.0938625693885356	$\Delta w_{1,3}$ = -0.025270691758451896
$\Delta w_{2,1}$ = -0.004069699630352522	$\Delta w_{2,2}$ = -0.0010956883620179868	$\Delta w_{2,3}$ = -0.006983508241433322
$\Delta w_{3,1}$ = -0.014176833400148631	$\Delta w_{3,2}$ = -0.09035783925369457	$\Delta w_{3,3}$ = -0.014176833400148631
$\Delta w_{4,1}$ = -0.0039699474318945615	$\Delta w_{4,2}$ = -0.0006228710625903537	$\Delta w_{4,3}$ = -0.0034292132127227157
$\Delta w_{5,1}$ = -0.01636469981697072	$\Delta w_{5,2}$ = -0.09009576492639923	$\Delta w_{5,3}$ = -0.021040328336105207
$\Delta w_{6,1}$ = -0.0063683517208044296	$\Delta w_{6,2}$ = -0.0014872198629423518	$\Delta w_{6,3}$ = -0.005936168170889558
$\Delta w_{7,1}$ = -0.029610042188754307	$\Delta w_{7,2}$ = -0.11818709147135267	$\Delta w_{7,3}$ = -0.015184637019874004

$\Delta w_{8,1}$ = - 0.00777107848341555	$\Delta w_{8,2}$ = - 0.0009992001518211847	$\Delta w_{8,3}$ = - 0.008010254550433164
$\Delta w_{9,1}$ = - 0.01887302073043834	$\Delta w_{9,2}$ = - 0.151298716189014	$\Delta w_{9,3}$ = - 0.0166711683118872
$\Delta w_{10,1}$ = - 0.0076517633609244256	$\Delta w_{10,2}$ = - 0.0008431256925758724	$\Delta w_{10,3}$ = - 0.007556315169312064

Усереднимо дані дельти для кожної синаптичної ваги($w_{нов} = w + \Delta w_{сер}$):

$\Delta w_{сер1}$ = -0.012496806087577467

$\Delta w_{сер2}$ = -0.05488500863609439

$\Delta w_{сер3}$ = -0.012425911817125776

Оновлюємо значення синаптичних ваг:

w_1 = 0.47128292044986914

w_2 = 0.3164907404258401

w_3 = 0.7891953828857577

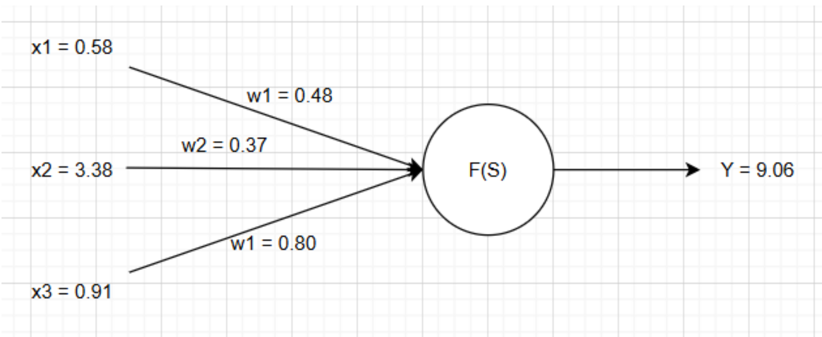
Обчислюємо сумарну помилку прогнозування, яка має формулу:

$$E = \sum_{i=1}^{N}(Y_i - y_i)^2$$

З попередніми ваговими коефіцієнтами сумарна помилка прогнозування становила **516.2796515138459**, тоді як після оновлення ваг вона зменшилася до **508.9101273337352**, що відповідає зниженню помилки на 1.43%.

Схема нейронної мережі

Нейронна мережа складається з одного нейрона з сигмоїдальною функцією активацією представленого на рисунку нижче:



де S = w1 * x1 + w2 * x2 + w3 * x3,

F(S) = 10 / (1 + e^(-S))

Лістинг програми на python

```
import random
import math
import matplotlib.pyplot as plt


# Сигмоїдальна функція активації
def sigmoid(x):
    return 10 / (1 + math.exp(-x))


# Похідна сигмоїдальної функції
def der_sigmoid(x):
    return (math.exp(-x) / (1 + math.exp(-x)) ** 2) * 10


# Тренування
def train_neuron(data, learning_rate=0.01, max_epochs=10_000, error_threshold=0.0001):
    # Випадкова генерація синаптичних ваг
    weights = [random.uniform(-1, 1) for _ in range(3)]

    # Кількість навчальних прикладів
    n = len(data) - 3
    previous_total_error = float('inf')
    errors = []
    # цикл тренування
    for epoch in range(max_epochs):

        total_error = 0
        deltas = [0, 0, 0]

        for i in range(n):
```

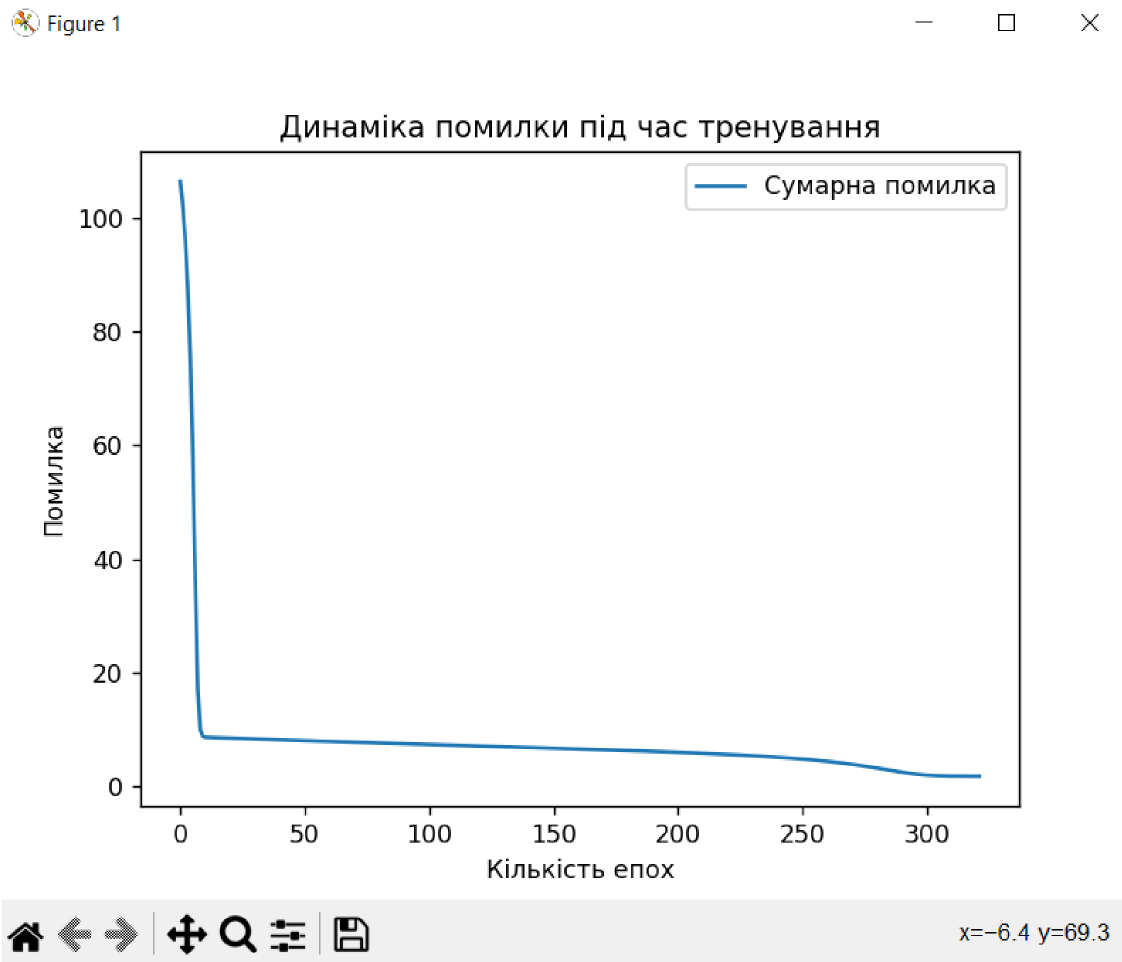
```
    x = [data[i], data[i+1], data[i+2]]

    y = data[i+3]
    # Зважувальна сума(вхід нейрона)
    w_sum = sum(x[j] * weights[j] for j in range(3))

    # Прогнозоване значенння(сигмоїдальна функція активації)
    y_pred = sigmoid(w_sum)

    # Квадратична помилка
    error = (y_pred - y) ** 2
    total_error += error

    # Похідна помилки для зворотного поширення (градієнт)
    delta = (y_pred - y) * der_sigmoid(w_sum)

    # Дельти для кожної ваги
    for j in range(3):
        deltas[j] += -learning_rate * delta * x[j]

# Оновлені ваги
for k in range(3):
    weights[k] += deltas[k] / n

errors.append(total_error)

# Якщо зміна помилки менше 0.01 * кікість навчальних наборів
if abs(previous_total_error - total_error) < error_threshold * n:
    break
```

```
        previous_total_error = total_error

    plt.plot(range(0, len(errors)), errors, label='Сумарна помилка')
    plt.xlabel('Кількість епох')
    plt.ylabel('Помилка')
    plt.title('Динаміка помилки під час тренування')
    plt.legend()
    plt.show()
    # Навчені вагові коефіцієнти
    return weights

# Функція прогнозу
def predict_next(data, weights):
    x1, x2, x3 = data[-3], data[-2], data[-1]
    s = x1 * weights[0] + x2 * weights[1] + x3 * weights[2]
    return sigmoid(s)

# 1 - 13 дані для тренування, 14 і 15 - тестові
data = [0.58, 3.38, 0.91, 5.80, 0.91, 5.01, 1.17, 4.67, 0.60, 4.81, 0.53, 4.75, 1.01, 5.04,
1.07]

# Тренування нейрону, поверне ваги
weights = train_neuron(data[:13])

# Прогнозуєм
predicted_x14 = predict_next(data[:13], weights)
predicted_x15 = predict_next(data[1:14], weights)

print(f"Прогнозуємо x14: {predicted_x14}")
```

```
print(f"Дійсне x14: {data[13]}")
print(f"Прогнозуємое x15: {predicted_x15}")
print(f"Дійсне x15: {data[14]}")


# Обчислюємо похибку
print(f"Похибка на тестових даних: {round((abs(predicted_x14 - data[13])/data[13] + abs(predicted_x15 - data[14])/data[14])*50, 2)}%")
```

Результат навчання

На діаграмі зображеній нижче наведено графік зменшення сумарної помилки прогнозування на нашому наборі:



Як бачимо на графіку для тренування нашої моделі потребувалось більше 300 епох

Результат тестування

В даному випадку значення похибки становить 14.65%

```
[Running] set PYTHONIOENCODING=utf8 && py -u "d:\Unik\ІКП\Pw1\main.py"
Прогнозуємо x14: 5.209064156495838
Дійсне x14: 5.04
Прогнозуємое x15: 0.7923111637945244
Дійсне x15: 1.07
Похибка на тестових даних: 14.65%

[Done] exited with code=0 in 226.038 seconds
```

В наступному випадку 13.87%

```
[Running] set PYTHONIOENCODING=utf8 && py -u "d:\Unik\ІКП\Pw1\main.py"
Прогнозуємо x14: 5.223837595255318
Дійсне x14: 5.04
Прогнозуємо x15: 0.8121187803960765
Дійсне x15: 1.07
Похибка на тестових даних: 13.87%

[Done] exited with code=0 in 4.293 seconds
```

Висновок

Під час виконання даної практичної роботи було отримано початкові знання та навички для створення штучних нейронних мереж на прикладі одношарової мережі з одного нейрона який прогнозує часовий ряд. Наша нейронна мережа використовує сигмоїдальну функцію активації а також алгоритм зворотного поширення помилки. Детальніше алгоритм було розписано на прикладі однієї ітераці,ї в результаті якої сумарну помилку було зменшено на 1.43%. Окрім того, під час тестування нашої нейронної мережі було отримано похибку 14.65% при навчанні за більше ніж 300 епох.