НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

Звіт

із лабораторної роботи №2

з дисципліни «Системи глибинного навчання»

на тему

“Розробка програмного забезпечення для реалізації ймовірнісної нейронної мережі PNN”

Виконав:
студент групи КМ-01
Іваник Ю. П.

Викладач:
Професор кафедри ПМА
Терейковський І. А.

Київ — 2023

Зміст

Теоретичні відомості

Нейронна мережа PNN є типом нейронних мереж, який використовує ядра для класифікації та прогнозування. Цей підхід виник з області статистики та обробки сигналів і визначається своєю здатністю працювати з неперервними та категоріальними даними.

Основна ідея PNN полягає в тому, щоб кожен екземпляр навчального набору даних представляти власне ядро, яке визначає вагу для прихованих нейронів. Коли потрібно здійснити класифікацію для нового прикладу, мережа використовує ядра для обчислення схожості між новим прикладом і навчальними екземплярами.

Основні компоненти PNN включають:

1. **Ядрові функції:** Кожен навчальний приклад служить ядром, що представляє собою функцію схожості між вхідними прикладами. Ці ядра визначають ваги для прихованих нейронів.
2. **Приховані нейрони:** Для кожного класу створюється прихований нейрон, який обчислює взважену суму схожості між новим прикладом і навчальними екземплярами.
3. **Комбінація класів:** Кількість прихованих нейронів дорівнює кількість класів у задачі класифікації. Вихідні значення цих нейронів об'єднуються для визначення кінцевого призначення класу для нового прикладу.

Перевагою PNN є здатність працювати як з числовими, так і з категоріальними даними, а також вміння працювати з невеликими обсягами даних. Однак вона може виявити схильність до перенавчання в разі обробки великої кількості шуму або надто складних завдань класифікації.

Основна частина

Завдання: розробити програмне забезпечення для реалізації мережі PNN, що призначена для апроксимації функції: $y = x_1 + x_2$

Передбачити режими навчання та розпізнавання.

Імпортуємо бібліотеки:

- *Numpy* для математичних розрахунків
- *Pandas* для зручного виводу інформації
- *Random* для генерації тестового набору даних

Створимо функцію *train_pnn*, яка прийматиме на вхід:

- X_train – вектор вхідних даних для тренування PNN
- Y_train – вектор міток для тренування PNN
- Значення параметра sigma для керування чутливістю до точок даних

Ця функція проходиться циклом по навчальним даним та їх міткам. Обчислює ймовірності класів за формулою ядра Гауса. Повертає словник, що містить ймовірності класів.

Створимо функцію *predict_pnn*, яка прийматиме на вхід:

- Class_probs – словник ймовірностей класів, отриманий у результаті навчання
- X_predict – точка даних для розпізнавання
- X_train – характеристики навчальних даних
- Значення параметра *sigma* для керування чутливістю до точок даних

Ця функція проходиться циклом по словнику ймовірностей класів та перевіряє до якого класу найбільше підходять дані для розпізнавання. Повертає мітку класу.

Далі визначаємо навчальні дані.

```
X_train = np.array([[0.00065, 0.00071], [0.0571, 0.0494], [0.9, 0.7],
                    [0.00034, 0.00045], [0.0454, 0.0662], [0.8, 0.9]])
Y_train = np.array(['A', 'B', 'C', 'A', 'B', 'C'])
```

Передаємо ці дані в функцію тренування, щоб PNN навчилась на навчальних даних.

Визначаємо дані для розпізнавання шляхом створення їх випадковим чином.

```
x_t = [[round(random.uniform(0, 1), 5), round(random.uniform(0, 1), 5)] for _ in
range(30)]
```

Та тестуємо PNN на тих даних яких вона ще не бачила.

	Data	Predicted class
0	[0.63943, 0.02501]	A
1	[0.27503, 0.22321]	B
2	[0.73647, 0.6767]	C
3	[0.89218, 0.08694]	A
4	[0.42192, 0.0298]	B
5	[0.21864, 0.50536]	B
6	[0.02654, 0.19884]	B
7	[0.64988, 0.54494]	A
8	[0.22044, 0.58927]	A
9	[0.80943, 0.0065]	A
10	[0.80582, 0.69814]	C
11	[0.34025, 0.15548]	B
12	[0.95721, 0.33659]	A
13	[0.09275, 0.09672]	B
14	[0.84749, 0.60373]	C
15	[0.80713, 0.72973]	C
16	[0.53623, 0.97312]	A
17	[0.37853, 0.55204]	A
18	[0.8294, 0.61852]	C
19	[0.86171, 0.57735]	C
20	[0.70457, 0.04582]	A
21	[0.2279, 0.28939]	B
22	[0.07979, 0.23279]	B
23	[0.101, 0.27797]	B
24	[0.63568, 0.36483]	A
25	[0.37018, 0.20951]	B
26	[0.26698, 0.93665]	A
27	[0.64804, 0.60913]	A
28	[0.17114, 0.72913]	A
29	[0.1634, 0.37946]	B

В результаті отримали такі класи для тестового набору даних із 30 записів

Загалом розподіл по класах виглядає так:

```
Кількість класів тестового набору:
Predicted class
A    13
B    11
C     6
Name: count, dtype: int64
```

Перелік посилань

1. Навчальний посібник «Основні концепції нейроних мереж» Роберт Каллан – стор. 158-164.

Додаток А – Код програми

```
# КМ-01, Іваник Юрій, Лаб 2
import numpy as np
import pandas as pd
import random
def train_pnn(X_train, Y_train, sigma):
    """
    Обчислюємо ймовірності класів за формулою ядра Гауса.
    Ядро Гауса вимірює схожість між точками даних.
    Воно базується на евклідовій відстані між X_train та x, зваженій на сігму
    """

    class_probs = {}
    for x, class_label in zip(X_train, Y_train):
        if class_label not in class_probs:  #  Якщо мітка класу відсутня у словнику, ініціалізуйте її значенням 0.0
            class_probs[class_label] = 0.0
        class_probs[class_label] += np.exp(-np.sum((X_train - x) ** 2, axis=1) / (2 * sigma ** 2))

    return class_probs

def predict_pnn(class_probs, X_predict, X_train, sigma):
    max_prob = 0
    predicted_class = None

    for class_label, prob in class_probs.items():
```

```
        similarity = np.exp(-np.sum((X_train - X_predict) ** 2, axis=1) / (2 * sigma ** 2))
        class_probs[class_label] += similarity
        if np.max(prob) > max_prob:
            max_prob = np.max(prob)
            predicted_class = class_label

    return predicted_class
#### Визначаємо навчальні дані
X_train = np.array([[0.00065, 0.00071], [0.0571, 0.0494], [0.9, 0.7],
                    [0.00034, 0.00045], [0.0454, 0.0662], [0.8, 0.9]])
Y_train = np.array(['A', 'B', 'C', 'A', 'B', 'C'])
#### Навчаємо та тестуємо PNN
Параметр sigma для бажаної точності
sigma = 0.165
random.seed(42)
x_t = [[round(random.uniform(0, 1), 5), round(random.uniform(0, 1), 5)] for _ in range(30)]

results_df = pd.DataFrame(columns=["Data", "Predicted class"])

for x in x_t:
    x = np.array(x)
    class_probs = train_pnn(X_train, Y_train, sigma)
    pred_cl = predict_pnn(class_probs, x, X_train, sigma)
    results_df = pd.concat([results_df, pd.DataFrame({"Data": [x.tolist()], "Predicted class": [pred_cl]})],
                           ignore_index=True)
```

```
print(results_df)
class_counts = results_df['Predicted class'].value_counts()
print(f'Кількість класів тестового набору:\n{class_counts}')
```