НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

Звіт з лабораторної роботи №2

З дисципліни «Методі штучного інтелекту»

Виконала:  
студентка групи КМ-81

Верзун Поліна

Керівник:  
Терейковська Л.О.

Київ-2021

Зміст

[ВСТУП 4](#_Toc53436196)

[1 ОСНОВНА ЧАСТИНА 5](#_Toc53436197)

[2 РЕЗУЛЬТАТ РОБОТИ 7](#_Toc53436198)

[ВИСНОВОКИ 8](#_Toc53436199)

[Список використаних джерел 9](#_Toc53436200)

[Додаток А 10](#_Toc53436201)

ВСТУП

Мета: розробити програмне забезпечення для реалізації ймовірнісної нейронної мережі PNN

Завдання: розробити програмне забезпечення для реалізації мережі PNN.   
Побудувати мережу PNN, призначену для класифікації об’єктів на А та В

1 ОСНОВНА ЧАСТИНА

Опис роботи

Імовірнісну мережу було реалізовано на мові програмування Python. Ймовірнісна НМ складається з вхідного шару, шару образів, шару додавання та вихідного шару - рис 1.

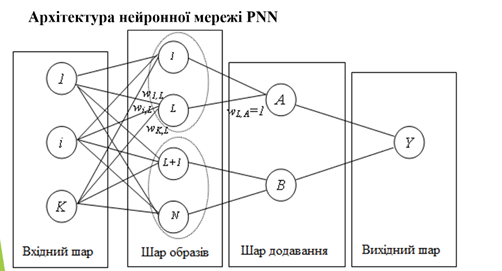


Рисунок 1 – Архітектура нейронної мережі PNN

Мережа PNN не потребує навчання, в тому сенсі, яке необхідно для мереж зі зворотним розповсюдженням помилки, так як всі параметри PNN (число елементів та ваг) визначаються безпосередньо учбовими даними.

Процедура для використання PNN є відносно нескладною. Архітектура визначається структурою учбових даних:

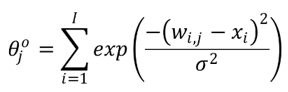
* Число вхідних елементів дорівнює числу ознак;
* Число елементів шару образів дорівнює числу учбових образів;
* Число елементів шару додавання дорівнює числу класів.

Перший шар вагових значень визначається учбовими образами. Для другого шару всі значення встановлюються рівними 1. Вагові значення кінцевого шару встановлюються так, щоб на виході був розпізнаний елемент шару додавання з найбільшим значення активності.

Після побудови мережі, невідомий екземпляр можна подати на вхід мережі і в результаті прямого проходу через мережу вихідний шар повертає клас, до якого імовірніше всього відноситься даних екземпляр.

Тобто, фактично, відбувається пошук евклідової відстані від невідомого прикладу до учбових.

Для знаходження активності реалізовано функції find\_activity, де обчислюється активність шару образів за допомогою формули 1.1.

(1.1)

Режим розпізнавання реалізовано в функції learn\_PNN. В цій функцію реалізовано:

1. Створення порожніх списки, що відповідають певному класу;
2. Обчислення активації елементу;
3. Обчислення середнього значення цих списків;
4. Визначення до якого класу належить вхідний вектор.

# 2 РЕЗУЛЬТАТ РОБОТИ

[1.0, 0] is A class

[3.0, 4] is A class

[12.0, 10] is B class

ВИСНОВОКИ

В ході виконання лабораторної роботи , було спроектовано ймовірнісну нейронну мережу. Навчання мережі було проведено на прикладах х1 = [1.,0], х2= [3.,4], х3 = [12., 10]. Хоча якість реалізації цієї мережі можна оцінити лише на великій тестовій вибірці, можна припустити, якщо а1 та а2 близькі близько 1 - 2, то приклад належить до класу «А»; якщо ж ці значення близькі до 10, то є велика імовірність, що даний приклад належить о класу «В»

Список використаних джерел

1. Лекція по МШІ №5.
2. Навчальний посібник «Основные концепции нейронных сетей» Роберт Каллан – стор. 158-164.

Додаток А

Лістинг коду

import math

X\_train = [[1., 1.],[1.,2],[2.,1.],[10.,10.]]

x = [[1.,0],[3.,4], [12., 10]]

radius = 0.3

def find\_activity(x, a):

Q = math.e\*\*(-((x-a)\*\*2)/(radius\*\*2))

return Q

def learn\_PNN(X\_train, x):

n = len(X\_train)

param\_num = len(x)

y\_example = [0]\*n

for i in range(n):

sum\_elem = 0

for j in range(param\_num):

sum\_elem += find\_activity(X\_train[i][j], x[j])

y\_example[i] = sum\_elem

y\_A = sum(y\_example[:3])/len(y\_example[:3])

y\_B = y\_example[-1]/1

if y\_A>= y\_B:

answer = "A"

else:

answer = "B"

return answer

for example in x:

class\_type = learn\_PNN(X\_train, example)

print('{} is {} class'.format(example, class\_type))