Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №2**

По дисциплине «Модели решения задач в интеллектуальных системах»

Тема: «SLP. Прогнозирование»

**Выполнил:**

Студент 3 курса

Группы ИИ-23

Макаревич Н.Р.

**Проверил:**

Туз И. С.

Брест 2024

**Цель работы**: Изучить обучение и функционирование SLP для решения задачи регрессии.

Y = a\*sin(bx) +d

Написать нейронную сеть(single layer perceptron) для решения задачи прогнозирования

 Функции.

Обучить сеть с использованием константного и адаптивного шага обучения, online-

learning и batch-learning. Результаты для каждого варианта сети занести в таблицу(

test error, количество эпох, время обучения и тд)

**Ход работы:**

Вариант 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 3 | 7 | 0.3 | 5 |

Код нейронной сети:

import numpy as np

import matplotlib.pyplot as plt

class SimpleNN:

def \_\_init\_\_(self, x\_len, y\_len):

self.X = np.zeros(x\_len)

self.Y = np.zeros(y\_len)

self.W = np.random.rand(x\_len, y\_len)

self.T = np.zeros(y\_len)

self.alp: float = 0.01

self.adaptive\_flag: bool = False

self.errors\_for\_chart = np.array([])

def forward\_prop(self, image):

self.X = image

return np.dot(image, self.W) - self.T

def adjust(self, error):

for i in range(self.W.shape[0]):

for j in range(self.W.shape[1]):

self.W[i, j] -= self.alp \* error \* self.X[i]

self.T[j] += self.alp \* error

def adaptive\_online(self, image):

self.alp = 1 / (1 + np.sum(image \*\* 2))

def adaptive\_batch(self, batch\_data, batch\_e):

dividend: float = 0

divider: float = 0

a = self.helper\_a(batch\_data, batch\_e)

for ind, image in enumerate(batch\_data):

dividend += (self.forward\_prop(image) - batch\_e[ind]) \* a

divider += a \*\* 2

self.alp = dividend / divider

def helper\_a(self, batch\_data, batch\_e) -> float:

a: float = 0

for ind, image in enumerate(batch\_data):

a += (self.forward\_prop(image) - batch\_e[ind]) \* (1 + np.sum(image \*\* 2))

return a

def train\_online(self, train\_data, train\_e, test\_data, test\_e):

max\_epoch: int = 100

for epoch in range(max\_epoch):

ind: int = 0

for image in train\_data:

Y = self.forward\_prop(image)

error = Y - train\_e[ind]

if self.adaptive\_flag:

self.adaptive\_online(image)

self.adjust(error)

ind += 1

if self.test(test\_data, test\_e):

print(f"Network trained in {epoch} epoches")

self.drawErrorGraph()

break

def test(self, test\_data, test\_e) -> bool:

mse: float = 0.0

ind: int = 0

for image in test\_data:

Y = self.forward\_prop(image)

mse += 0.5 \* (Y - test\_e[ind]) \*\*2

ind += 1

print(f"MSE: {mse}")

self.errors\_for\_chart = np.append(self.errors\_for\_chart, mse)

return mse < 0.00000000000001

def train\_batch(self, train\_data, train\_e, test\_data, test\_e, batch\_size):

max\_epoch: int = 100

num\_of\_batches = len(train\_data) / batch\_size

for epoch in range(max\_epoch):

start: int = 0

while start < num\_of\_batches:

batch\_data = train\_data[start: start + batch\_size]

batch\_e = train\_e[start: start + batch\_size]

batch\_error: float = 0

for ind, image in enumerate(batch\_data):

Y = self.forward\_prop(image)

error = Y - batch\_e[ind]

batch\_error += error

if self.adaptive\_flag:

self.adaptive\_batch(batch\_data, batch\_e)

self.adjust(batch\_error)

start += batch\_size

if self.test(test\_data, test\_e):

print(f"Network trained in {epoch} epoches")

self.drawErrorGraph()

return

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Adaptive alp** | | **Const alp** | |
| **batch** | **online** | **batch** | **online** |
| **Epochs** | 3 | 2 | 10 | 3 |
| **Error** | 1.11e-15 | 3.99e-16 | 9.27e-15 | 2.83e-17 |
| **time** | 4.23 | 3.52 | 6.99 | 2.38,. |

**Вывод:** в ходе лабораторной работы я научился реализовывать однослойную нейронную сеть для решения задачи прогнозирования