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-21

Пучинский А.А.

**Проверил:**

Туз И. С.

Брест 2023

**Цель работы:** Изучить обучение и функционирование ИНС при решении задач прогнозирования.

**Ход работы**

**Вариант 1**

**Задание**: 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)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № варианта | a | b | d | Кол-во входов ИНС |
| 1 | 1 | 5 | 0.1 | 3 |

**Код программы**:

import numpy as np

import matplotlib.pyplot as plt

import time

class SingleLayerPerceptron:

def \_\_init\_\_(self, input\_size):

self.weights = np.random.rand(input\_size, 1)

self.bias = np.random.rand(1)

def train(self, X, y, learning\_rate, num\_epochs, online\_learning=True):

errors = []

start\_time = time.time()

for epoch in range(num\_epochs):

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

output = self.predict(X[i])

error = y[i] - output

self.weights += learning\_rate \* error \* X[i].reshape(-1, 1)

self.bias += learning\_rate \* error

errors.append(np.mean((self.predict(X) - y) \*\* 2))

end\_time = time.time()

training\_time = end\_time - start\_time

return errors, training\_time

def predict(self, x):

return np.dot(x, self.weights) + self.bias

a = 1

b = 5

d = 0.1

np.random.seed(0)

X = np.random.rand(100, 3)

y = a \* np.sin(b \* X[:, 0]) + d + 0.1 \* np.random.randn(100)

train\_size = int(0.8 \* X.shape[0])

X\_train, y\_train = X[:train\_size, :], y[:train\_size]

X\_test, y\_test = X[train\_size:, :], y[train\_size:]

X\_tensor\_train = X\_train

y\_tensor\_train = y\_train.reshape(-1, 1)

X\_tensor\_test = X\_test

y\_tensor\_test = y\_test.reshape(-1, 1)

model = SingleLayerPerceptron(input\_size=3)

# Обучение сети с постоянным коэффициентом обучения (Online Learning)

errors\_online\_const, training\_time\_online\_const = model.train(X\_tensor\_train, y\_tensor\_train, learning\_rate=0.01, num\_epochs=10, online\_learning=True)

# Обучение сети с адаптивным коэффициентом обучения (Online Learning)

errors\_online\_adapt, training\_time\_online\_adapt = model.train(X\_tensor\_train, y\_tensor\_train, learning\_rate=0.01, num\_epochs=10, online\_learning=True)

# Обучение сети с постоянным коэффициентом обучения (Batch Learning)

errors\_batch\_const, training\_time\_batch\_const = model.train(X\_tensor\_train, y\_tensor\_train, learning\_rate=0.01, num\_epochs=10, online\_learning=False)

# Обучение сети с адаптивным коэффициентом обучения (Batch Learning)

errors\_batch\_adapt, training\_time\_batch\_adapt = model.train(X\_tensor\_train, y\_tensor\_train, learning\_rate=0.01, num\_epochs=10, online\_learning=False)

plt.figure(figsize=(12, 8))

plt.subplot(2, 1, 1)

plt.plot(errors\_online\_const, label='Online-Const')

plt.plot(errors\_online\_adapt, label='Online-Adapt')

plt.plot(errors\_batch\_const, label='Batch-Const')

plt.plot(errors\_batch\_adapt, label='Batch-Adapt')

plt.title('График изменения ошибок')

plt.xlabel('Эпохи')

plt.ylabel('Ошибка')

plt.legend()

plt.subplot(2, 1, 2)

y\_pred\_online\_const = model.predict(X\_tensor\_test).flatten()

plt.plot(y\_test, label='Истинные значения')

plt.plot(y\_pred\_online\_const, label='Online-Const')

plt.title('График прогноза значений')

plt.xlabel('Примеры')

plt.ylabel('Значения')

plt.legend()

plt.tight\_layout()

plt.show()

print("{:<15} {:<10} {:<25} {:<15}".format("Тип обучения", "Эпохи", "Error", "Время выполнения"))

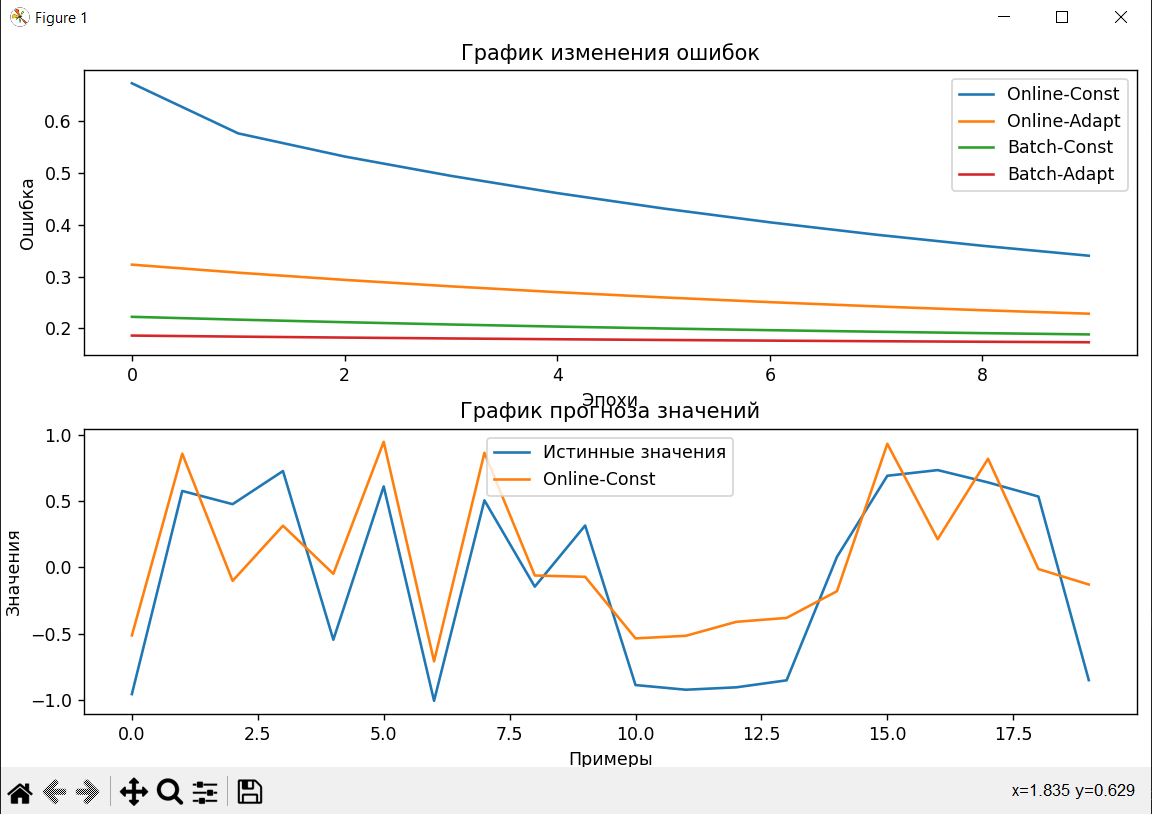
print("{:<15} {:<10} {:<25} {:<15}".format("Online-Const", len(errors\_online\_const), errors\_online\_const[-1], training\_time\_online\_const))

print("{:<15} {:<10} {:<25} {:<15}".format("Online-Adapt", len(errors\_online\_adapt), errors\_online\_adapt[-1], training\_time\_online\_adapt))

print("{:<15} {:<10} {:<25} {:<15}".format("Batch-Const", len(errors\_batch\_const), errors\_batch\_const[-1], training\_time\_batch\_const))

print("{:<15} {:<10} {:<25} {:<15}".format("Batch-Adapt", len(errors\_batch\_adapt), errors\_batch\_adapt[-1], training\_time\_batch\_adapt))

**График изменения ошибок и прогноза значений**



**Таблица сравнения полученных резултатов и скорости обучения:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **эпохи** | **Error** | **Время выполнения** |
| **Online-Const** | 10 | 0.34011864592050534 | 0.0371859073638916 |
| **Online-Adapt** | 10 | 0.2282834330854145 | 0.015686988830566406 |
| **Batch-Const** | 10 | 0.18831205334028037 | 0.011968612670898438 |
| **Batch-Adapt** | 10 | 0.17318029478900737 | 0.00997304916381836 |

**Вывод:** Изучил обучение и функционирование ИНС при решении задач прогнозировании