Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №2**

По дисциплине: «Аппаратное обеспечение интеллектуальных систем»

Тема: «Линейная искусственная нейронная сеть. Правило обучения Видроу-Хоффа»

**Выполнил:**

Студент 2 курса

Группы ИИ-21

Кирилович А. А.

**Проверил:**

Михно Е.В.

Брест 2023

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

**Задание:** написать на любом ЯВУ программу моделирования прогнозирующей линейной ИНС. Для тестирования использовать функцию ** .

Варианты заданий приведены в следующей таблице:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № варианта | a | b | d | Кол-во входов ИНС |
| 4 | 4 | 8 | 0.4 | 3 |

Обучение и прогнозирование производить на 30 и 15 значениях соответственно табулируя функцию с шагом 0.1. Скорость обучения выбирается студентом самостоятельно, для чего моделирование проводится несколько раз для разных α. Результаты оцениваются по двум критериям - скорости обучения и минимальной достигнутой ошибке. Необходимо заметить, что эти критерии в общем случае являются взаимоисключающими, и оптимальные значения для каждого критерия достигаются при разных α.

import numpy as np

import matplotlib.pyplot as plt

class ADALINE:

def \_\_init\_\_(self, alpha=0.01, n\_in=3):

self.alpha = alpha

self.n\_in = n\_in

self.W = np.random.rand(n\_in)

self.T = np.zeros(1)

self.errors = []

def prep\_data(self, y):

return np.array([y[i - self.n\_in:i] for i in range(self.n\_in, len(y))])

def calc\_error(self, x, e):

return (1 / 2) \* ((self.predict(x) - e) \*\* 2)

def predict(self, x):

return self.W @ x - self.T

def training(self, X, E):

for x, e in zip(X[:31 - self.n\_in], E[self.n\_in:31]):

y = self.predict(x)

self.W += -self.alpha \* (y - e) \* x

self.T += self.alpha \* (y - e)

def testing(self, X, E):

for x, e in zip(X[31 - self.n\_in:], E[31:]):

print(f"x: {x}; y: {self.predict(x)}; e: {e}")

def set\_mean\_error(self, X, E):

self.errors.append(np.mean([self.calc\_error(x, e) for x, e in zip(X[31 - self.n\_in:], E[31:])]))

def main(epochs=100, min\_error=1e-3):

model = ADALINE()

f = lambda a, b, d, x: a \* np.sin(b \* x) + d

y = f(4, 8, 0.4, np.arange(0, 4.5, 0.1))

x = model.prep\_data(y)

for epoch in range(epochs):

model.training(x, y)

model.set\_mean\_error(x, y)

if model.errors[-1] <= min\_error:

break

print('Training results')

print(f'Epochs: {epoch}; Error: {model.errors[-1]}')

print('Testing results')

model.testing(x, y)

plt.plot(range(1, len(model.errors) + 1), model.errors)

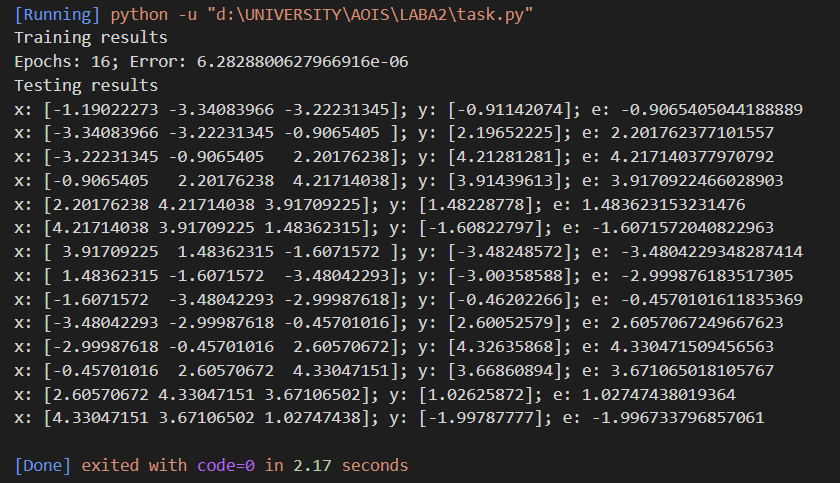
plt.xlabel('Epoch')

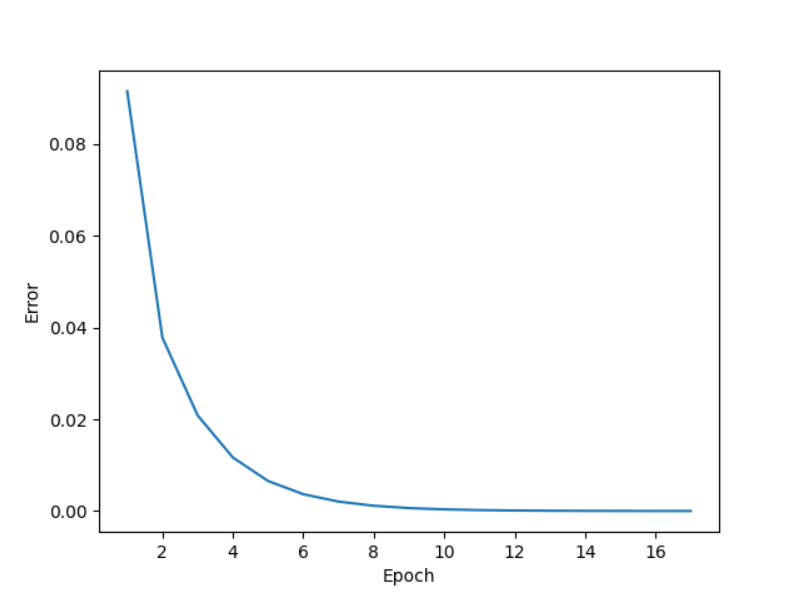
plt.ylabel('Error')

plt.savefig('ADALINE.png')

if \_\_name\_\_ == "\_\_main\_\_":

main()





**Вывод:** изучил обучение и функционирование линейной ИНС при решении задач прогнозирования.