1Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №5**

По дисциплине «Модели решения задач в интеллектуальных системах»

Тема: «MLP. Прогнозирование»

**Выполнил:**

Студент 3 курса

Группы ИИ-21

Пучинский А.А.

**Проверил:**

Туз И. С.

Брест 2023

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

**Ход работы**

**Вариант 1**

F(x) = 0.1 \* cos(0.1 \* x) + 0.05 \*sin(0.1 \* x)

from matplotlib import pyplot as plt

import numpy as np

from scipy.special import expit

f = lambda x: 0.3 \* np.cos(0.1 \* x) + 0.06 \* np.sin(0.1 \* x)

def normalize\_data(data, min\_val, max\_val):

min\_data = np.min(data)

max\_data = np.max(data)

normalized\_data = (data - min\_data) / (max\_data - min\_data) \* (max\_val - min\_val) + min\_val

return normalized\_data

ERROR = []

class Network:

def \_\_init\_\_(self, input\_size, hidden\_size, output\_size, learning\_rate=0.03):

self.input\_size = input\_size

self.hidden\_size = hidden\_size

self.output\_size = output\_size

self.learning\_rate = learning\_rate

self.E = 0

self.weights\_input\_hidden = np.random.randn(self.input\_size, self.hidden\_size)

self.bias\_hidden = np.zeros((1, self.hidden\_size))

self.weights\_hidden\_output = np.random.randn(self.hidden\_size, self.output\_size)

self.bias\_output = np.zeros((1, self.output\_size))

def sigmoid(self, x):

return (1 / (1 + expit(-x)))

def sigmoid\_derivative(self, x):

return x \* (1 - x)

def forward(self, inputs):

self.hidden\_input = np.dot(inputs, self.weights\_input\_hidden) + self.bias\_hidden

self.hidden\_output = self.sigmoid(self.hidden\_input)

self.output = np.dot(self.hidden\_output, self.weights\_hidden\_output) + self.bias\_output

return self.output

def backward(self, inputs, target, output):

error = target - output

self.E += error[0]

delta\_hidden = error.dot(self.weights\_hidden\_output.T) \* self.sigmoid\_derivative(self.hidden\_output)

self.weights\_hidden\_output += self.hidden\_output.T.dot(error) \* self.learning\_rate

self.bias\_output += np.sum(error, axis=0, keepdims=True) \* self.learning\_rate

self.weights\_input\_hidden += inputs.T.dot(delta\_hidden) \* self.learning\_rate

self.bias\_hidden += np.sum(delta\_hidden, axis=0, keepdims=True) \* self.learning\_rate

def train(self, inputs, targets, epochs):

for epoch in range(epochs):

for i in range(len(inputs)):

input\_data = np.array([inputs[i]])

target\_data = np.array([targets[i]])

output = self.forward(input\_data)

self.backward(input\_data, target\_data, output)

self.E \*\*= 2

ERROR.append(self.E)

self.E = 0

def predict(self, inputs):

output = self.forward(inputs)

result = [el[0] for el in output]

return normalize\_data(result, -0.3, 0.3)

def get\_train\_data(all\_points, input\_size):

result\_X = [all\_points[i:i+input\_size] for i in range(len(all\_points) - input\_size)]

result\_Y = [all\_points[i] for i in range(input\_size, len(all\_points), 1)]

return np.array(result\_X), np.array(result\_Y)

input\_size = 6

hidden\_size = 2

output\_size = 1

all\_train\_points = f(np.arange(0, 200, 0.2))

all\_test\_points = f(np.arange(180, 380, 0.2))

X\_train, Y\_train = get\_train\_data(all\_train\_points, input\_size)

X\_test, Y\_test = get\_train\_data(all\_test\_points, input\_size)

NN = Network(input\_size, hidden\_size, output\_size)

NN.train(X\_train, Y\_train, 30)

plt.plot(range(len(ERROR)), ERROR)

plt.show()

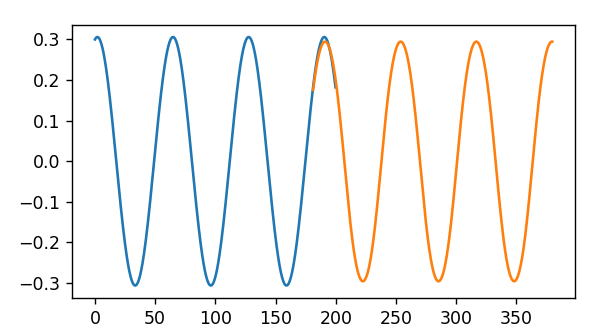
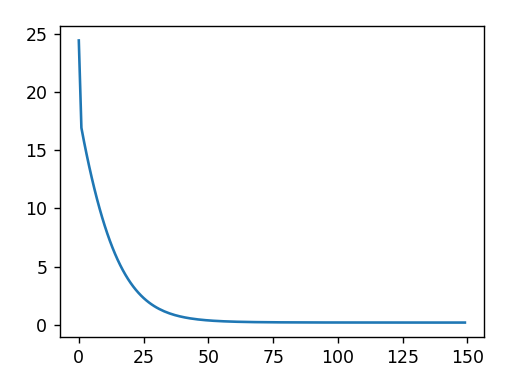
predicted = NN.predict(X\_test)

plt.plot(np.arange(0, 200, 0.2), all\_train\_points)

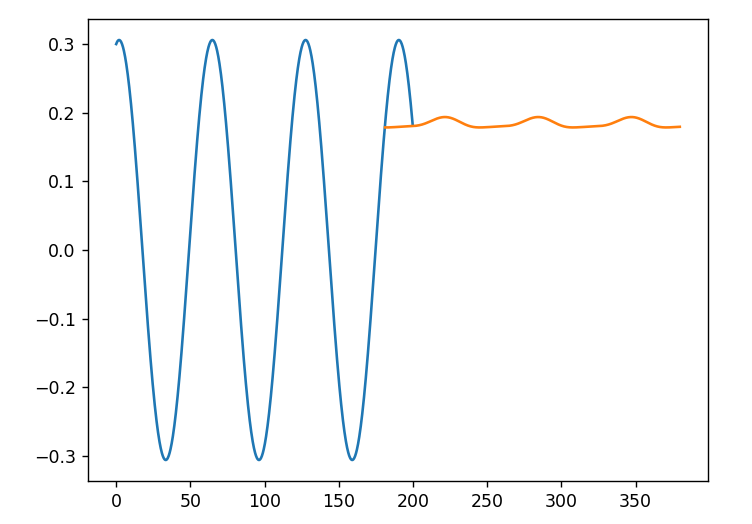
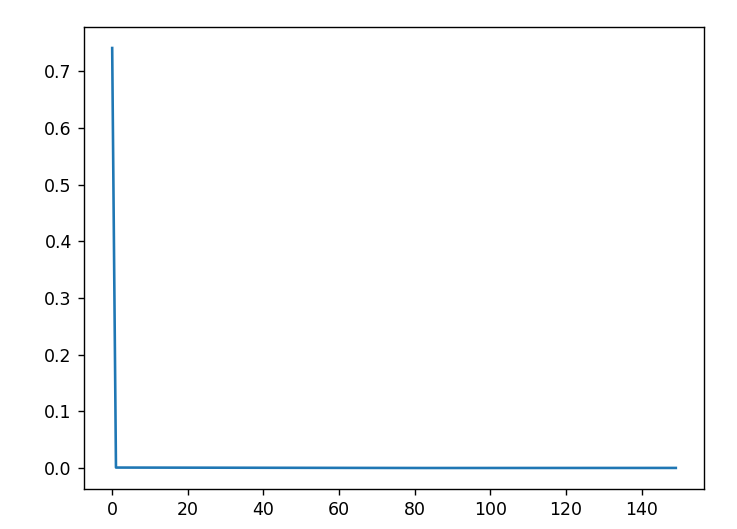
plt.plot(np.arange(181.2, 380, 0.2), predicted)

plt.show()

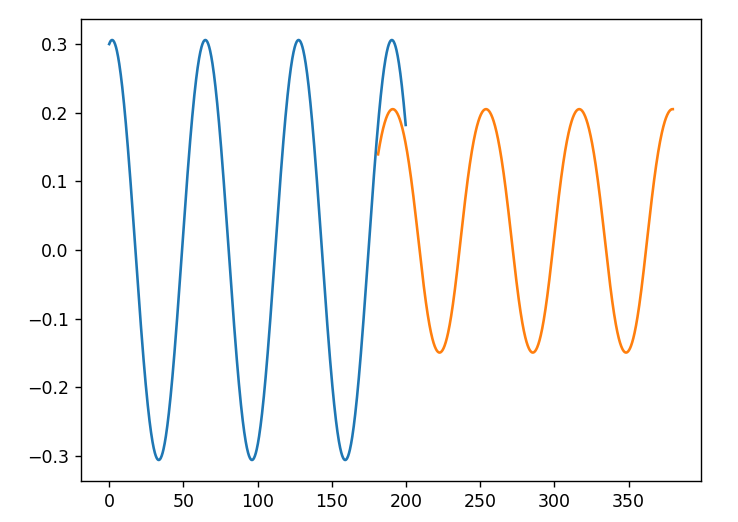
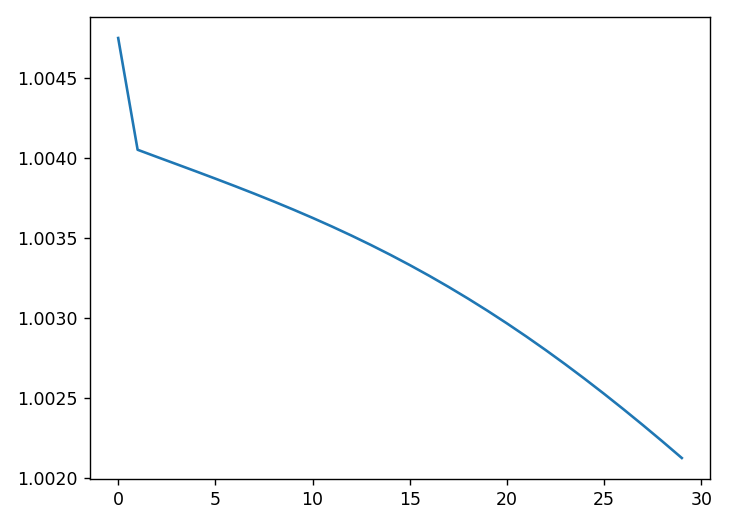
Online const:



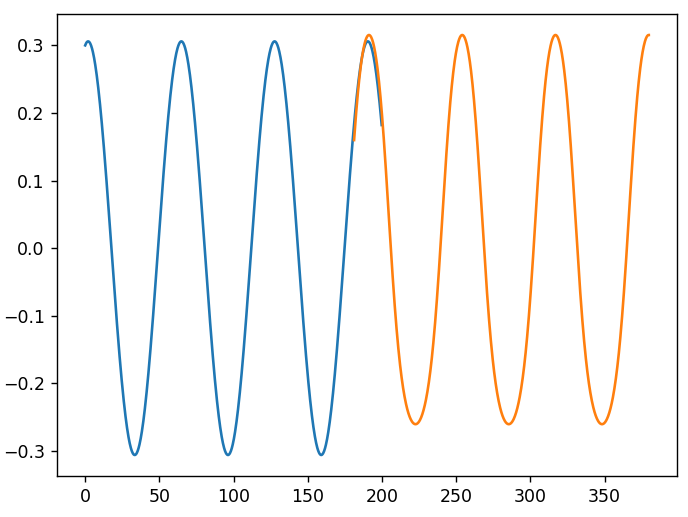
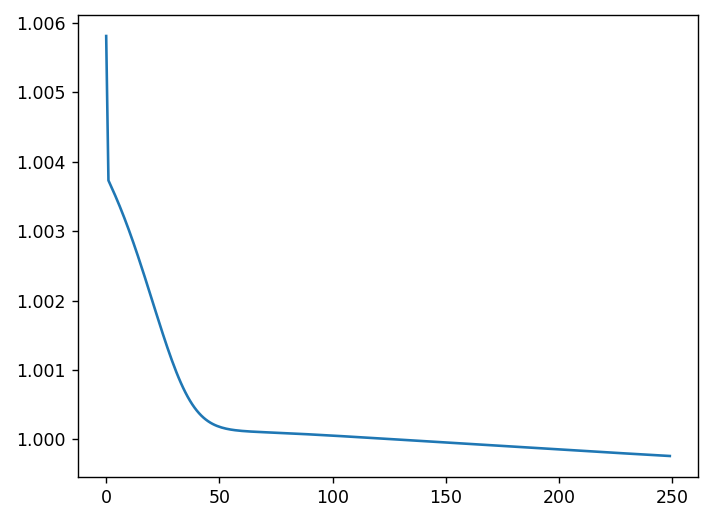
Online adaptive:



Batch const:



Batch adaptive:



В моем случае batch learning отрабатывает за большее количество эпох, по сравнению с online-learning.

**Вывод:** в ходе лабораторной работы я научился создавать ИНС для прогнозирования значений функции.