|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)

КАФЕДРА «Информационная безопасность» (ИУ8)

Отчёт

по лабораторной работе № 2

по дисциплине «Интеллектуальные технологии информационной безопасности»

**Тема: «Применение однослойной нейронной сети с линейной функцией активации»**

Выполнил: Евула А. С.,

студент группы ИУ8-63

Вариант: 4

Проверил: Волосова Н.К,

аспирант каф. ИУ8

г. Москва,

2021 г.

1. Цель работы

Изучить возможности однослойный НС в задачах прогнозирования временных рядов методом скользящего окна (авторегрессия).

1. Постановка задачи

На временном интервале [a, b] задан дискретный набор значений функции: *sin(x-1).* Количество точек N = 20, расположение – равномерное. Методом “скользящего окна” спрогнозировать поведение функции на N точках последующего интервала (b, 2b – a). Для решение использовать однослойную нейронную сеть с количеством нейронов p=4 и линейной функцией активации. Обучение проводить методом Видроу-Хоффа. Исследовать количество эпох обучения и коэффициента обучения на среднеквадратичной погрешность приближения. Исследовать процесс прогнозирование при постепенным изменении (уменьшение /увеличение) размера окна. Сделать выводы по результатам численного эксперимента.

1. Ход работы
2. Сначала были сформированы наборы иксов на интервале [a, b], [a, 2b – a] и вычислены значения функции на заданном интервале.
3. При старте обучения нейронной сети (НС) были сгенерированы наборы весов в количестве **длина окна + 1** и выбрана норма обучения равная rate = 0.55.
4. В процессе самого обучения мы начинали с элемента с индексом длины окна (необходимо для начала обучения).
5. Сравнивая полученное значение с реальным, получается ошибка прогноза.
6. Далее проводится корреляция весов по формуле:



, где:





xk – значение функции на k-ом шаге.

1. Корреляция происходит до достижения минимально допустимой погрешности или прохождения заданного количества эпох.

1. Выводы

В ходе выполнения данной лабораторной работы были получены результаты обучения нейронной сети методом “скользящего окна”. Исходя из них можно сделать вывод, что чем больше размер окна тем вычисления более неточны.

**Приложение 1 Исходный код программы.**

# Copyright 2021, Evula A. S., All rights reserved.

# IC8-63 BMSTU

from math import sin, cos, exp, sqrt

from matplotlib import pyplot as plt

class NN:

def \_\_init\_\_(self, p, rate, maxEpoch, minSigma):

self.a = -5

self.b = 3

self.N = 20

self.foo = lambda x: 0.5\*exp(0.5\*cos(0.5\*x)) + sin(0.5\*x)

self.calc()

self.W = [0] \* (p+1)

self.p = p

self.rate = rate

self.MAXEPOCH = maxEpoch

self.MINSIGMA = minSigma

self.train()

def calc(self):

self.c = 2\*self.b - self.a

self.d = (self.b-self.a)/(self.N-1) # dist between points

# correct foo

X = [self.a + self.d\*i for i in range(2\*self.N - 1)]

Y = [self.foo(x) for x in X]

self.graph(X, Y)

def train(self):

self.delta = 1

self.epoch = 0

while self.delta > self.MINSIGMA and self.epoch < self.MAXEPOCH:

self.tick()

self.epoch += 1

def tick(self):

self.delta = 0

for i in range(self.N - self.p + 1):

y = [self.foo(self.a + (i+j)\*self.d) for j in range(self.p)]

net = self.W[0]

for j in range(self.p):

net += self.W[j+1] \* y[j]

sigma = self.foo(self.a + self.d\*(i+self.p)) - net

self.correctWeights(sigma, y)

self.delta += sigma\*sigma

self.delta = sqrt(self.delta)

def correctWeights(self, sigma, y):

for i in range(self.p):

self.W[i+1] += self.rate \* sigma \* y[i]

def predict(self):

start = self.b - (self.d\*self.p - 1)

Y = []

X = [start+self.d\*(i+self.p) for i in range(self.N-1)]

for i in range(self.N-1):

y = [self.foo(start + (i+j)\*self.d) for j in range(self.p)]

net = self.W[0]

for j in range(self.p):

net += self.W[j+1] \* y[j]

Y.append(net)

self.graph(X, Y)

def graph(self, x, y):

plt.plot(x, y)

plt.grid()

plt.xlabel('x')

plt.ylabel('y')

def foo(\_p, \_rate):

nn = NN(

p = \_p,

rate = \_rate,

maxEpoch = 1000,

minSigma = 0.05

)

nn.train()

nn.predict()

plt.show()

print(f'window size: {round(nn.p, 2)}\nteaching rate: {round(nn.rate, 2)}')

print(f'epochs: {round(nn.MAXEPOCH,2)}\nweights:', end=' ')

for w in nn.W:

print(round(w,2), end=' ')

if \_\_name\_\_ == '\_\_main\_\_':

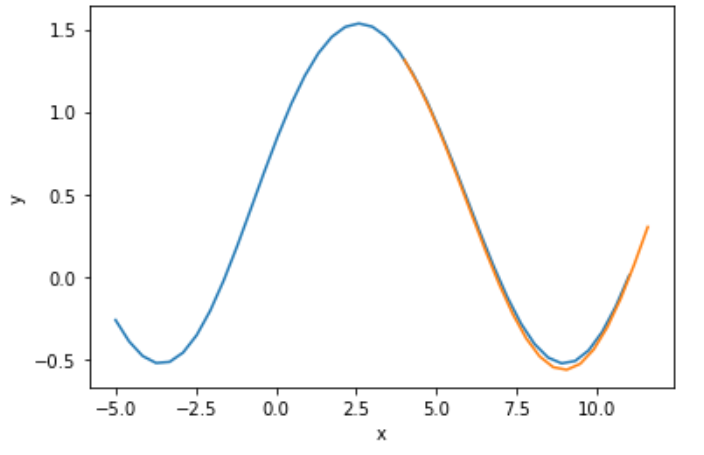
foo(4, 0.3)

foo(4, 0.6)

foo(6, 0.3)

foo(6, 0.6)

**Приложение 2 Вывод.**

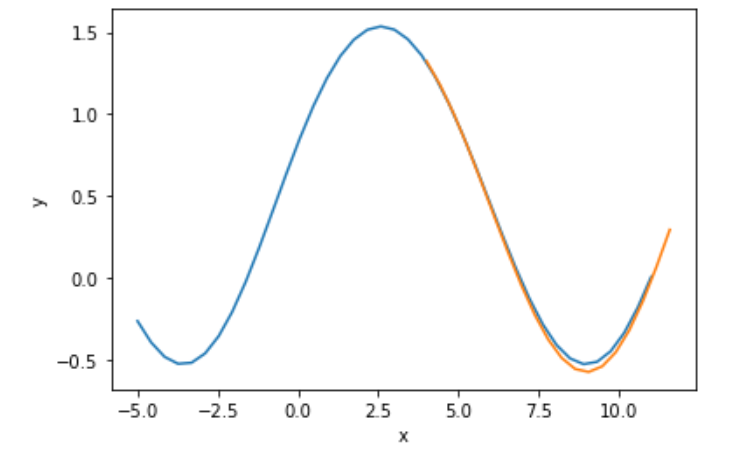


window size: 4

teaching rate: 0.3

epochs: 500

weights: 0 0.33 -0.84 -0.36 1.85

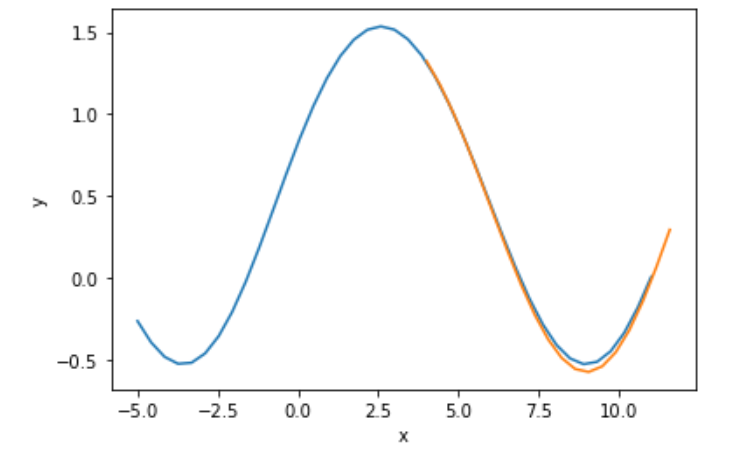


window size: 4

teaching rate: 0.3

epochs: 1000

weights: 0 0.73 -0.27 -0.72 -0.55 0.23 1.62

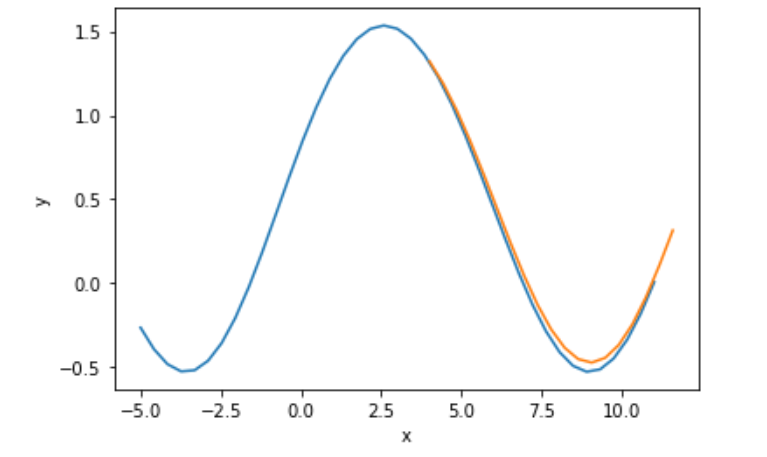


window size: 6

teaching rate: 0.6

epochs: 500

weights: 0 0.33 -0.25 -0.46 -0.28 0.32 1.32



window size: 6

teaching rate: 0.6

epochs: 1000

weights: 0 0.73 -0.27 -0.72 -0.55 0.23 1.62