Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет: “Информационные технологии и прикладная математика”
Кафедра: 806 “Вычислительная математика и программирование”

Отчет по лабораторной работе №2

по курсу «Нейроинформатика»

Студент: Полей-Добронравова Амелия Вадимовна

Группа: М8О-407Б, № по списку 20.

Преподаватель: Аносова Н.П.

Дата: 20.09.2021

Итоговая оценка:

Подпись преподавателя:

Москва, 2021

Тема лабораторной: “Линейная нейронная сеть. Правило обучения Уидроу-Хоффа”

Целью работы является исследование свойств линейной нейронной сети и алгоритмов ее обучения, применение сети в задачах аппроксимации и фильтрации.

Основные этапы работы:

1. Использовать линейную нейронную сеть с задержками для аппроксимации функции. В качестве метода обучения использовать адаптацию.

2. Использовать линейную нейронную сеть с задержками для аппроксимации функции и выполнения многошагового прогноза.

3. Использовать линейную нейронную сеть в качестве адаптивного фильтра для подавления помех. Для настройки весовых коэффициентов использовать метод наименьших квадратов.

Вариант 20

20.	$x = \sin\left(-2\sin(t)t^2 + 7t\right), \quad t \in [0.5, 3.2],\ h = 0.01$ $x = \cos\left(-\cos(t)t^2 + t\right), \quad t \in [0.5, 4],\ h = 0.01$	$y = \frac{1}{4}\cos\left(-\cos(t)t^2 + t + 2\pi\right)$

Ход работы

Поставлена задача регрессии - предсказания конкретного значения некой функции.

Функция f нужна для формирования массива входных сигналов x, функция y для y соответственно.

Класс LinearNet - нейронная сеть для одномерного входа, LinearNetNd - сеть для многомерного входа (для 3 части задания). Их можно объединить в один класс, если задать размерность весов от размерности входа, но для удобства и демонстрации я разнесла это на отдельные классы.

Многошаговый прогноз у меня выполнен для 10 точек.

p - зашумленное множество для 3 части задания.

Исходный код

```
import math
import numpy as np
import random
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
```

```
h = 0.01
t = 0.5
x1 = []
y1 = []

def f(t):
  return math.sin(-2 * math.sin(t) * t * t + 7 * t)

def y(t):
  return math.cos(-1 * math.cos(t) * t * t + t + 2 * math.pi) / 4

while t <= 3.2:
  x1.append(f(t))
  y1.append(y(t))
  t += h
x1 = np.array(x1)
y1 = np.array(y1)
```

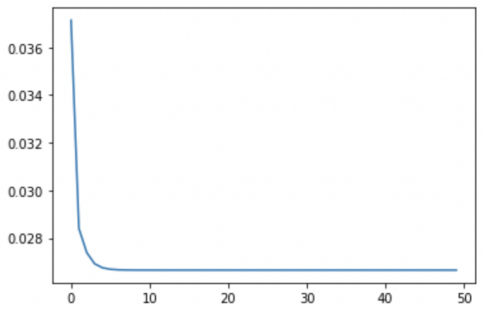
```
class LinearNet(object):
  def __init__(self, speed = 0.01, iter = 50, delays = 1, epsilon = 1e-6):
    self.speed = speed
    self.iterations = iter
    self.th = random.random()
    self.delays = delays
    self.last_x = np.zeros(delays)
    self.epsilon = epsilon

  def predict_one(self):
    sum = 0
    for i in range(self.delays):
      sum += self.last_x[self.delays - i - 1] * self.w[i] + self.th
    return sum

  def predict(self, X):
    res = []
    new_last = self.last_x
    for x in X:
        sum = 0
        for i in range(self.delays):
          sum += new_last[self.delays - i - 1] * self.w[i] + self.th
        #аналог move_last_x
        for i in range(self.delays - 1):
          new_last[i] = new_last[i+1]
          new_last[self.delays - 1] = x
        res.append(sum)
    return res

  def move_last_x(self, x):
    for i in range(self.delays - 1):
      self.last_x[i] = self.last_x[i+1]
    self.last_x[self.delays - 1] = x

  def fit(self, X, y):
      np.random.seed(42)
      self.w = np.zeros(self.delays)
      loss = []
      XX = X[self.delays:len(X) - 1] #без delays первичных точек
```

```
    yy = y[self.delays:len(y) - 1]
    for i in range(self.iterations):
      for j in range(self.delays):
        self.last_x[j] = X[j]
      l = []
      for xi, yi in zip(XX, yy):
        pred = self.predict_one()
        l.append(pred)
        if abs(pred - yi) > self.epsilon:
          for j in range(self.delays):
            self.w[j] -= self.speed * self.last_x[j] * (pred - yi)
          self.th -= self.speed * (pred - yi)
        self.move_last_x(xi)
      loss.append(mean_squared_error(yy, l))
    return loss, l
```

```
linear_net = LinearNet(speed=0.01)
```

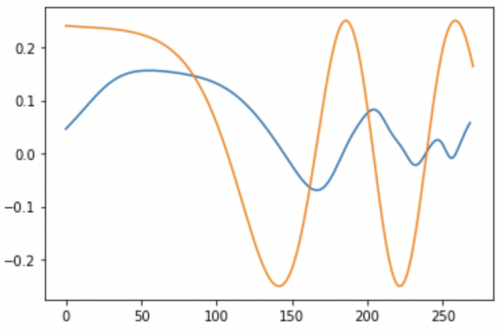
```
l, pred = linear_net.fit(x1, y1)
plt.plot(l)
```

```
[<matplotlib.lines.Line2D at 0x7f9761c76510>]
```



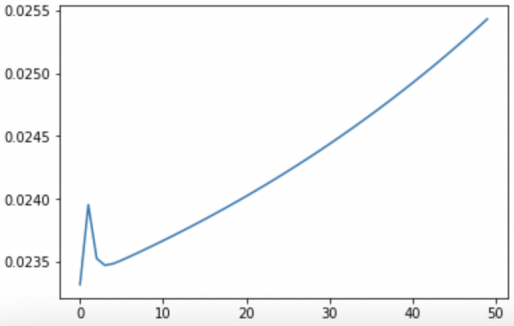

```
plt.plot(pred)
plt.plot(y1)
```

```
[<matplotlib.lines.Line2D at 0x7f9761e6bf90>]
```



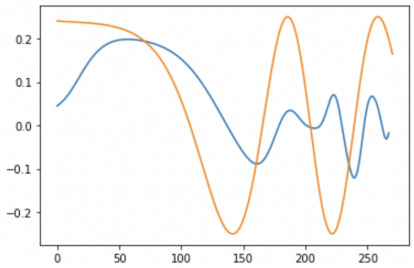

```
l2, pred = LinearNet(delays = 2,speed=0.01).fit(x1, y1)
plt.plot(l2)
```

```
[<matplotlib.lines.Line2D at 0x7f9761487b10>]
```



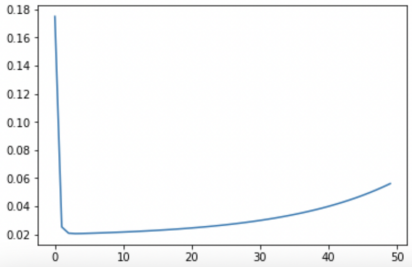
```
plt.plot(pred)
plt.plot(y1)
```

```
[<matplotlib.lines.Line2D at 0x7f97612d8d50>]
```



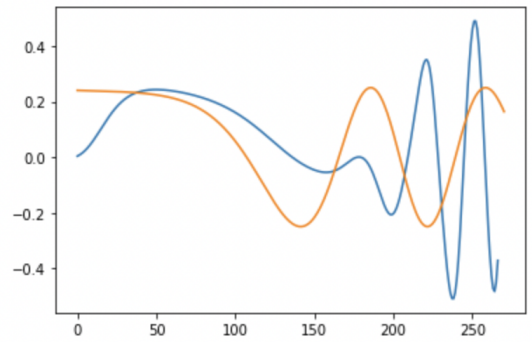

```
l3, pred = LinearNet(delays = 3, speed=0.01).fit(x1, y1)
plt.plot(l3)
```

```
[<matplotlib.lines.Line2D at 0x7f97612a6910>]
```



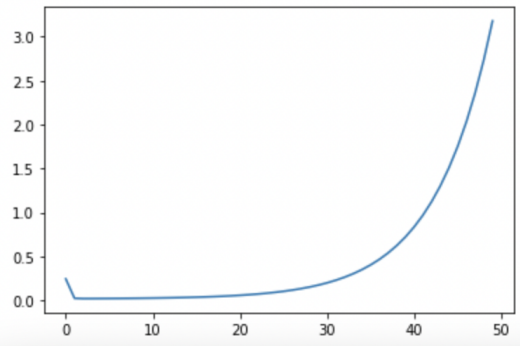

```
plt.plot(pred)
plt.plot(y1)
```

```
[<matplotlib.lines.Line2D at 0x7f976122e450>]
```



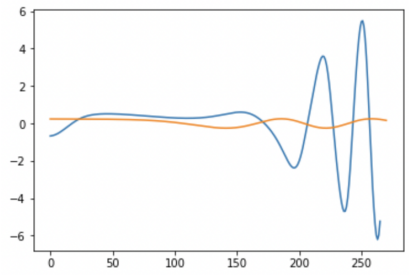

```
l4, pred = LinearNet(delays = 4, speed=0.01).fit(x1, y1)
plt.plot(l4)
```

```
[<matplotlib.lines.Line2D at 0x7f976128aa50>]
```



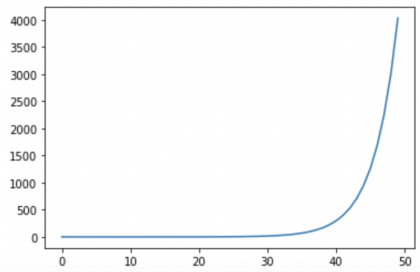
```
plt.plot(pred)
plt.plot(y1)
```

```
[<matplotlib.lines.Line2D at 0x7f976117ebd0>]
```



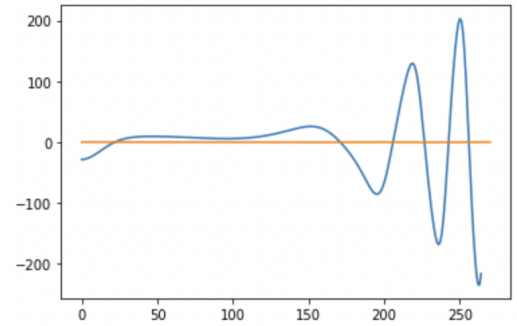

```
l5, pred = LinearNet(delays = 5, speed=0.01).fit(x1, y1)
plt.plot(l5)
```

```
[<matplotlib.lines.Line2D at 0x7f97610de450>]
```



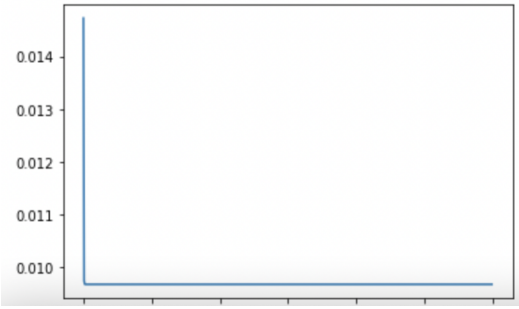

```
plt.plot(pred)
plt.plot(y1)
```

```
[<matplotlib.lines.Line2D at 0x7f9761061e50>]
```



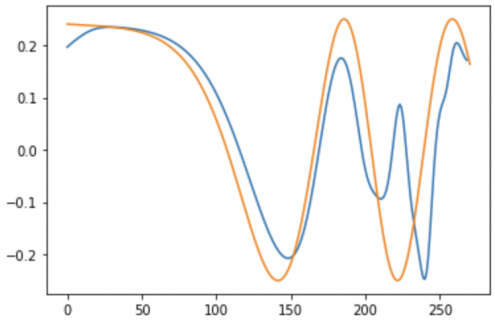

```
#ВТОРАЯ ЧАСТЬ ЗАДАНИЯ
net2 = LinearNet(iter = 600, speed=0.1)
g, pred = net2.fit(x1, y1)
plt.plot(g)
print('threhold = ', net2.th, 'w', net2.w)
```

```
threhold =  0.20021566374614044 w [0.03262227]
```



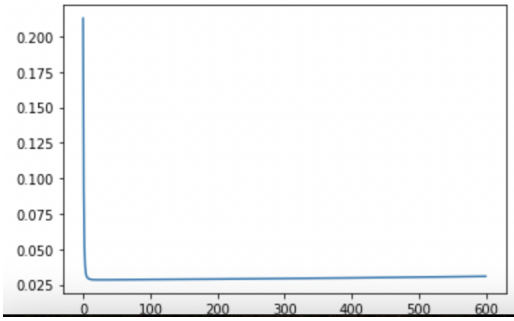
```
plt.plot(pred)
plt.plot(y1)
```

```
[<matplotlib.lines.Line2D at 0x7f9760e8f190>]
```



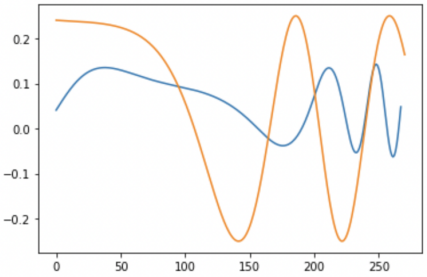

```
net2 = LinearNet(iter = 600, delays=2, speed = 1e-3)
g2, pred = net2.fit(x1, y1)
plt.plot(g2)
print('threhold = ', net2.th, 'w', net2.w)
```

```
threhold =  0.024155271102146027 w [ 0.28491372 -0.3545
```



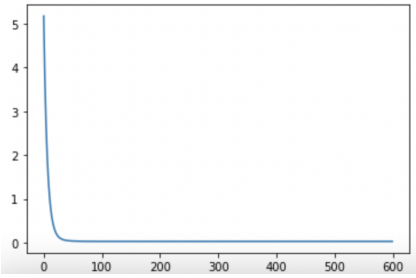

```
plt.plot(pred)
plt.plot(y1)
```

```
[<matplotlib.lines.Line2D at 0x7fa7b8a90ed0>]
```



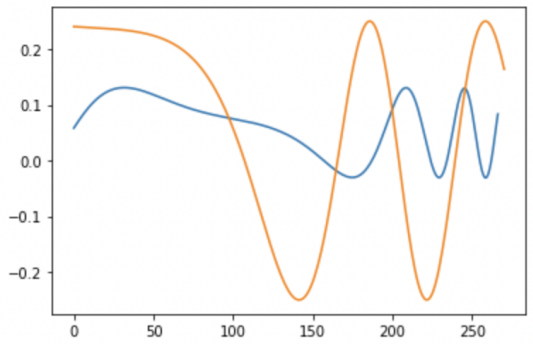

```
net2 = LinearNet(iter = 600, delays=3, speed = 1e-4)
g3, pred = net2.fit(x1, y1)
plt.plot(g3)
print('threhold = ', net2.th, 'w', net2.w)
```

```
threhold =  0.016773449416606134 w [ 0.02457122 -0.02
```



```
plt.plot(pred)
plt.plot(y1)
```

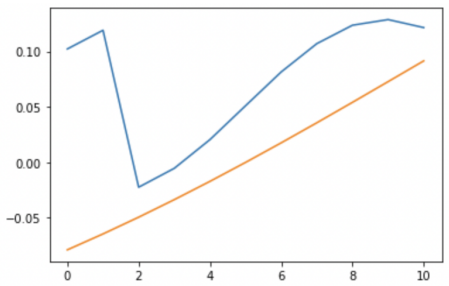
```
[<matplotlib.lines.Line2D at 0x7f9761a4c7d0>]
```




```
#многошаговый прогноз
t = 4
y_test = [y(t)]
x_test = [f(t)]
for i in range(10):
  t += h
  y_test.append(y(t))
  x_test .append(f(t))

x_test = np.array(x_test)
y_test = np.array(y_test)
```

```
pred = net2.predict(x_test)
plt.plot(pred)
plt.plot(y_test)
print('LOSS', mean_squared_error(pred, y_test))
```

```
LOSS 0.008188236151282013
```




```
#ТРЕТЬЯ ЧАСТЬ ЗАДАНИЯ
def j(t):
  return math.cos(math.cos(t) * t * t + t)

x2 = []
t = 0.5
while t <= 4:
  x2.append(j(t))
  t += h

x2 = np.array(x2)
```

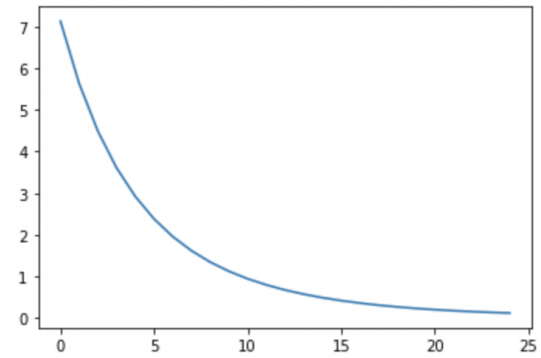
```
p = np.zeros((4, len(x2))) #?????????
for i in range(4):
  k = 0
  for j in range(i, len(x2)):
    p[i][j] = x2[k]
    k += 1
print(p)
```

```
[[0.7522041  0.74048753 0.72850566 ... 0.94653848 0.96820391 0.98470692]
 [0.         0.7522041  0.74048753 ... 0.92029054 0.94653848 0.96820391]
 [0.         0.         0.7522041  ... 0.89003161 0.92029054 0.94653848]
 [0.         0.         0.         ... 0.8563182  0.89003161 0.92029054]]
```

```
net3 = LinearNet_Nd(iter = 25, delays=4, speed = 1e-4)
r3, pred = net3.fit(p, y1)
```

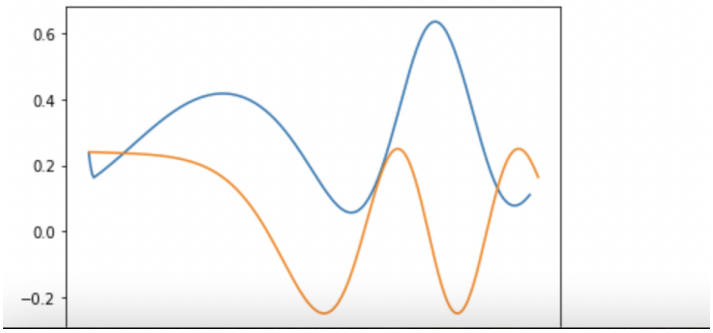
```
plt.plot(r3)
```

```
[<matplotlib.lines.Line2D at 0x7f97614cf350>]
```




```
plt.plot(pred)
plt.plot(y1)
```

```
[<matplotlib.lines.Line2D at 0x7f9761467050>]
```



Выводы

Объединение задержки с нейронной сетью считается линейным фильтром.

Большая глубина погружения не гарантирует лучший результат, её нужно подбирать индивидуально под задачу.

Задержки позволяют предсказывать значения функций с многими экстремумами, потому что подобрать значения весов для предсказания только по последней точке, способные разрешать функции и возрастать и уменьшаться, невозможно, а линейная комбинация весов для нескольких последних точек может справиться.