МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

**Лабораторная работа №2**

**по курсу «Нейроинформатика»**

**Линейная нейронная сеть. Правило обучения Уидроу-Хоффа**

Выполнил: Гамов Павел Антонович

Группа: 8О-407Б-18

Преподаватель: Аносова Н. П.

Москва, 2022

**Условие**

Цель работы – исследование свойств линейной нейронной сети и алгоритмов ее обучения применение сети в задачах аппроксимации и фильтрации. Использование нейронных сетей с задержками.

**Метод решения и описание программы**

Нам требуется аппроксимировать функцию. Будем подавать в качестве обучения некоторый диапазон значений функции, а правильность проверять со следующими, мы постараемся обучить сеть не только аппроксимировать функцию, но и по возможности предсказывать ее последовательность в надежде достроить график.

class Linear:

def \_\_init\_\_(self, D):

self.w = [uniform(-5,5) for i in range(D)]

self.b = uniform(-5,5)

self.D = D

def predict(self, data, D=None):

assert len(data) >= self.D, 'len of data incorrect'

return sum([self.w[i] \* data[i] for i in range(self.D)]) + self.b

def fitp(self, data):

assert len(data) >= self.D, 'len of data incorrect'

for i in range(self.D, len(data)-1):

return [self.predict(data[i - self.D:i]) for i in range(self.D, len(data)-1)]

def fit(self, data, epoc=50, lr=0.01):

hist = []

for ep in range(epoc):

h = []

for i in range(self.D, len(data)-1):

y = self.predict(data[i - self.D:i])

t = data[i+1]

h.append(abs(y - t))

for j in range(self.D):

self.w[j] = self.w[j] - lr \* (y - t) \* data[i - self.D + j]

hist.append(sum(h)\*\*0.5)

return hist

def linearappend(self, data, N):

data = self.fitp(data)

for i in range(N):

data.append(self.predict(data[-1 - self.D:-1]))

return data

class LinearN:

def \_\_init\_\_(self, D, outp):

self.w = [[uniform(-5,5) for i in range(D)] for j in range(outp)]

self.b = [uniform(-5,5) for j in range(outp)]

self.D = D

self.outp = outp

def predict(self, data, D=None):

assert len(data) >= self.D, 'len of data incorrect'

return [sum([self.w[j][i] \* data[i] for i in range(self.D)]) + self.b[j] for j in range(self.outp)]

def fitp(self, data):

assert len(data) >= self.D, 'len of data incorrect'

for i in range(self.D, len(data)-1):

return [self.predict(data[i - self.D:i])[0] for i in range(self.D, len(data)-1)]

def fit(self, data, epoc=600, lr=0.01):

hist = []

for ep in range(epoc):

h = []

for i in range(self.D, len(data)-1-self.outp):

y = self.predict(data[i - self.D:i])

t = data[i + 1:i + 1 + self.outp]

assert len(y) == len(t)

h.append(sum([y[k] - t[k] for k in range(self.outp)]))

for n in range(self.outp):

for j in range(self.D):

self.w[n][j] = self.w[n][j] - lr \* (y[n] - t[n]) \* data[i - self.D + j]

hist.append(sum(h)\*\*0.5)

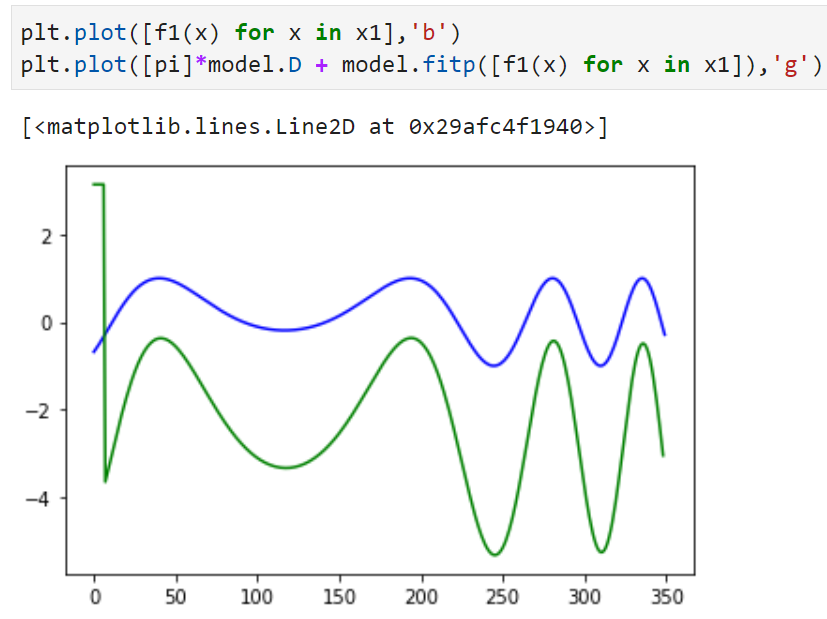
if abs(hist[-1]) < 10\*\*-6:

print(f"epoc = {ep}, exit by err")

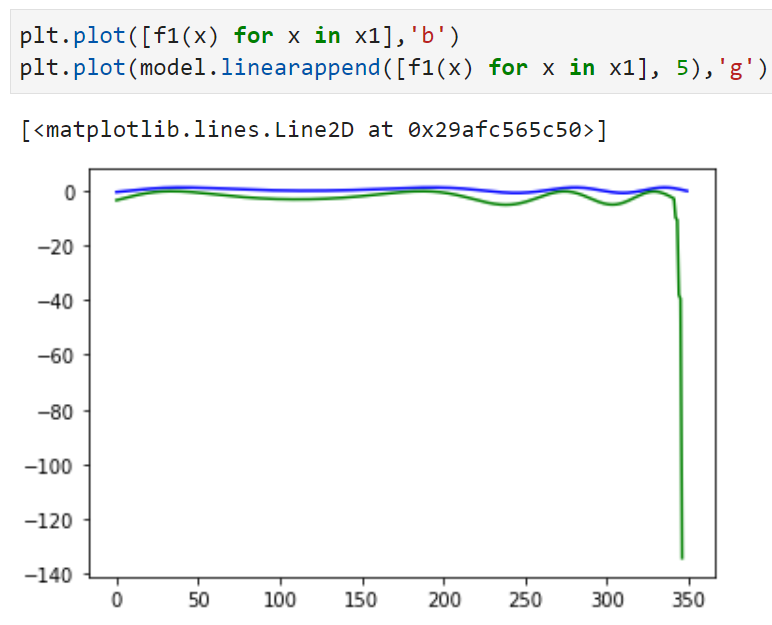
return hist

return hist

**Результаты**



Функция повторяет очертания графика. Зеленый – то, что выдала сеть после обучения на данном интервале из 350 точек. В качестве начальных данных просто передаем значения числа Пи.



Правда в попытке продолжить график мы потерпели провал. Сеть сломалась и стала выдавать неправдоподобное.

**Выводы**

Такого плана нейронные сети позволяют строить аппроксимации для функций, а также имеют потенциал к продолжению или предсказыванию следующих последовательностей.