Лабораторная работа 2

Линейная нейронная сеть. Правило обучения Уидроу-Хоффа

Тимофеев А.В., М8О-407Б-19

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

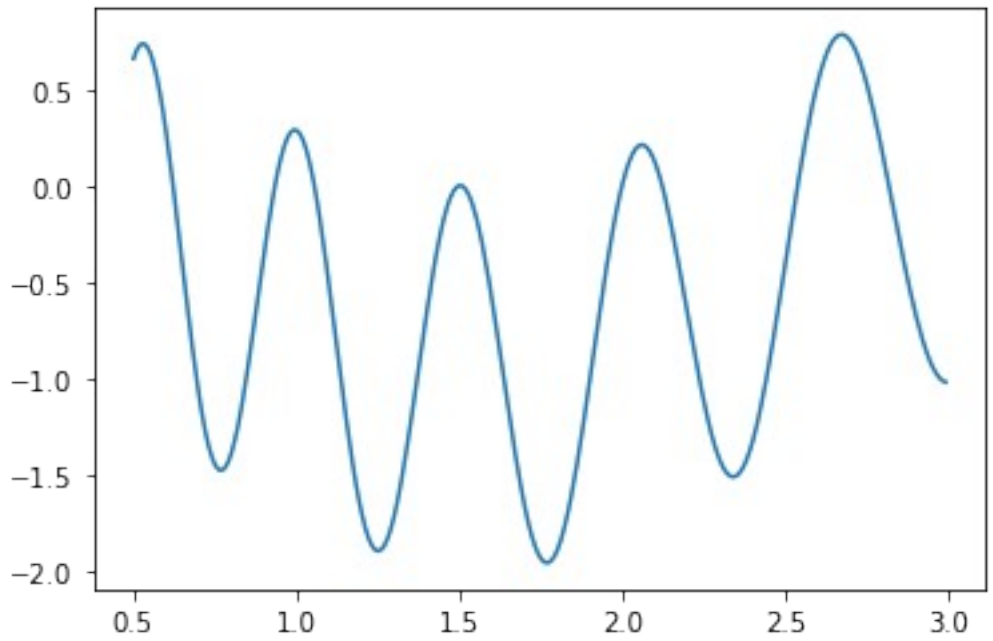
Вариант 12

```
import os
import keras
import tensorflow as tf
from keras.layers import *
import matplotlib.pyplot as plt
import numpy as np
import pylab

EPOCHES = 10
D = 5

def visualize(x, y, train_x, train_y, model, h):
    figure = plt.figure(figsize=(20, 10))
    pred = model.predict(train_x)
    axes = figure.add_subplot(221)
    plt.plot(x[D:], y[D:])
    plt.plot(x[D:], pred)
    plt.ylabel("y")
    plt.xlabel("x")
    axes = figure.add_subplot(222)
    epticks = [(i + 1) for i in range(len(h))]
    plt.plot(epticks, h, "g")
    plt.ylabel("mae")
    plt.xlabel("Эпохи")
    plt.xticks(epticks)
    axes = figure.add_subplot(223)
    plt.plot(x[D:], y[D:] - pred.flat, "r")
    plt.ylabel("Разность предсказания и сигнала, y")
    plt.xlabel("x")
    plt.show()
```

Задание №1

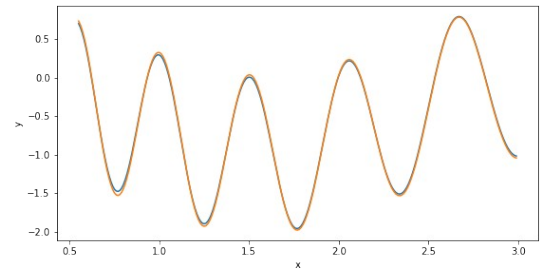
```
def signal(t):
    return (np.sin(t**2-15*t+3) - np.sin(t)**2)

h = 0.01
x = np.arange(0.5, 3, h)
y = signal(x)
```

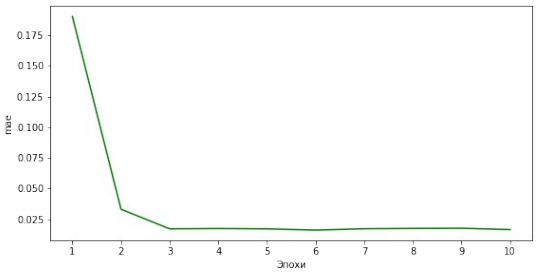
```
plt.plot(x, y)
plt.show()
```

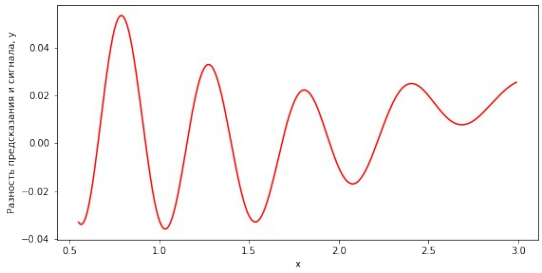



```
n = x.shape[0]

train_x, train_y = [], []

for i in range(n - D):
    xx = y[i:i + D]
    yy = y[i + D]
    train_x.append(xx)
    train_y.append(yy)

train_x = np.array(train_x)
train_y = np.array(train_y)

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(1, input_dim = D, activation = "linear")
])

model.compile(
    optimizer = tf.keras.optimizers.SGD(learning_rate = 0.05),
    loss = "mse",
    metrics = ["mae"]
)

h = model.fit(x = train_x, y = train_y, batch_size = 1, epochs =
EPOCHES, verbose = False, shuffle = True)

visualize(x, y, train_x, train_y, model, h.history["mae"])
```

```
8/8 [==============================] - 0s 714us/step
```

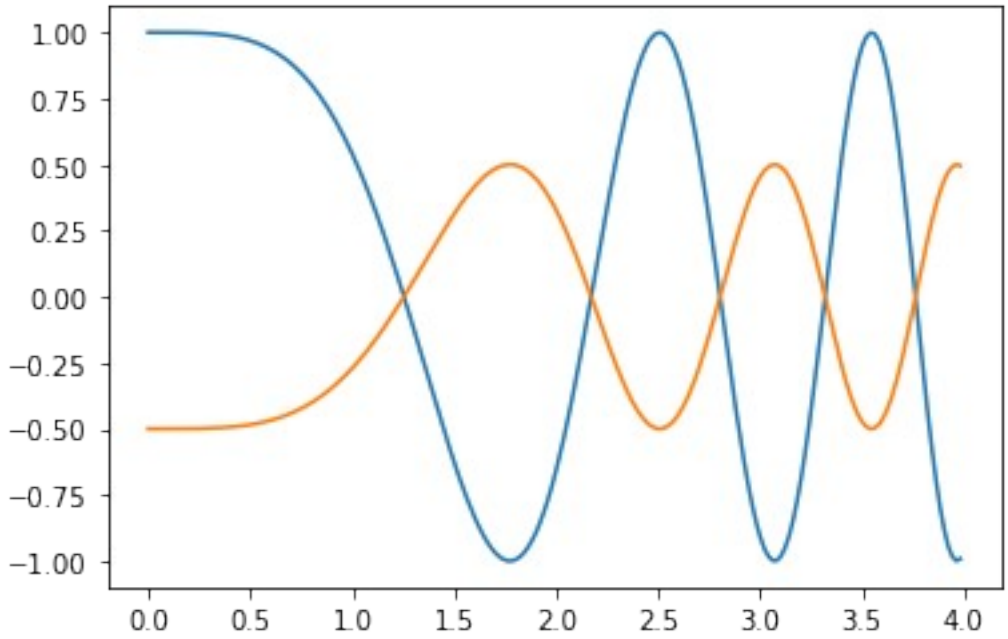




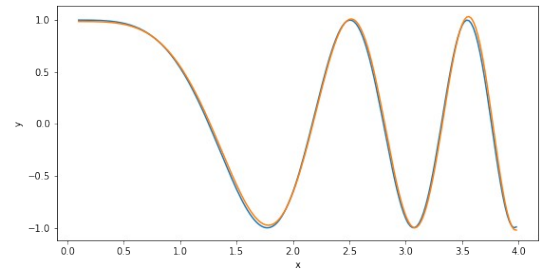


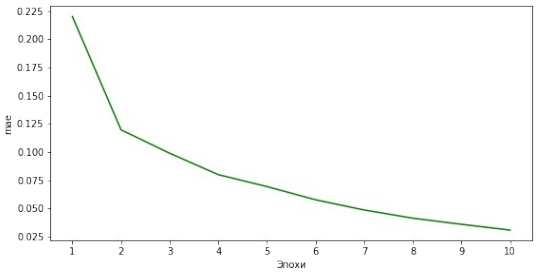
Задание №2

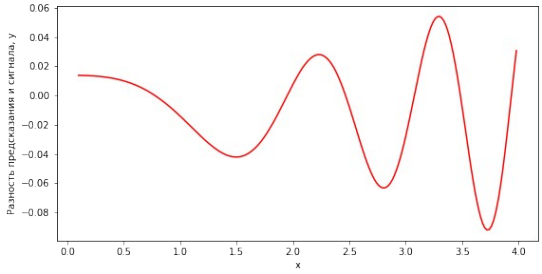
```python
def signal_true(t):
    return np.cos(t**2)

def signal_noise(t):
    return 0.5 * np.cos(t**2 + np.pi)

h = 0.02
x = np.arange(0, 4, h)
y_true = signal_true(x)
y_noise = signal_noise(x)

plt.plot(x, y_true)
plt.plot(x, y_noise)
plt.show()
```




```
n = x.shape[0]

train_x, train_y = [], []

for i in range(n - D):
    xx = y_noise[i:i + D]
    yy = y_true[i + D]
    train_x.append(xx)
    train_y.append(yy)

train_x = np.array(train_x)
train_y = np.array(train_y)

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(1, input_dim = D, activation = "linear")
])

model.compile(
    optimizer = tf.keras.optimizers.SGD(learning_rate = 0.05),
    loss = "mse",
    metrics = ["mae"]
)

h = model.fit(x = train_x, y = train_y, batch_size = 1, epochs =
EPOCHES, verbose = False, shuffle = True)

visualize(x, y_true, train_x, train_y, model, h.history["mae"])

7/7 [==============================] - 0s 833us/step
```







Вывод

В ходе выполнения лабораторной работы я ознакомился с задачами апроксимации и фильтрации, реализовал их решение с помощью однослойной нейросети.