Лабораторная работа 2

Линейная нейронная сеть. Правило обучения Уидроу-Хоффа

Воронов К.М., М8О-407Б-19

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

Вариант 19

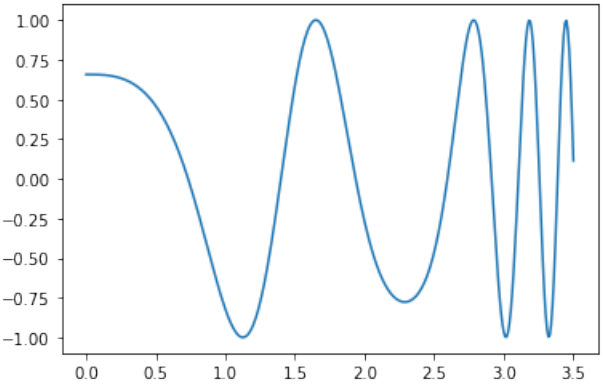
[ ]:
```
import keras
import tensorflow as tf
from keras.layers import *
import matplotlib.pyplot as plt
import numpy as np
import pylab
```

[ ]:
```
!pip install matplotlib --upgrade
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-
packages (3.5.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-
packages (from matplotlib) (0.11.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-
packages (from matplotlib) (21.3)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.7/dist-
packages (from matplotlib) (7.1.2)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-
packages (from matplotlib) (1.21.6)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.7/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.7/dist-packages (from matplotlib) (4.37.4)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib) (1.4.4)
Requirement already satisfied: pyparsing>=2.2.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib) (3.0.9)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.7/dist-packages (from kiwisolver>=1.0.1->matplotlib)
(4.1.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-
packages (from python-dateutil>=2.7->matplotlib) (1.15.0)
```

[ ]:
```
def f1(t):
    return np.sin(-2 * np.sin(t) * t * t + 7)
```

[ ]:
```
fig, ax = pylab.subplots(1, 1)
t =  np.arange(0, 3.501, 0.01)
plt.plot(t, f1(t))
plt.show()
```



[ ]:
```
def get_windows(window_size, t, f1, f2):
    ansx = []
    ansy = []
    for i in range(len(t) - window_size):
        ansx.append([])
        for j in range(window_size):
            ansx[i].append(f1(t[i + j]))
        ansy.append(f2(t[i + window_size]))

    return np.array(ansx), np.array(ansy)
```

[ ]:
```
winsize = 5
wins, labels = get_windows(winsize, t, f1, f1)
```

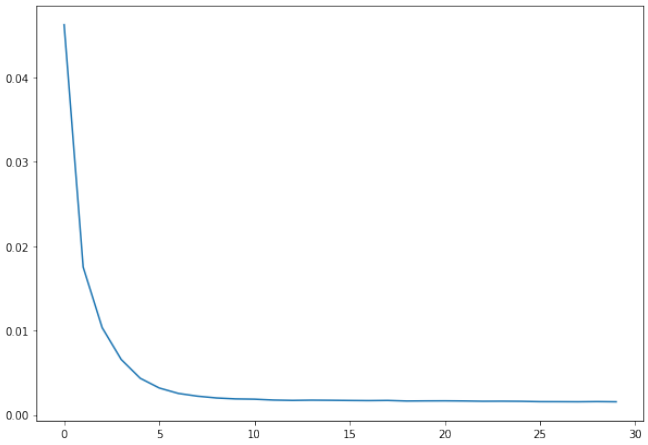
[ ]:
```
model = keras.models.Sequential()

model.add(Dense(1, activation = "linear",  kernel_initializer = keras.initializers.
 ↪RandomNormal(stddev=0.01), bias_initializer = keras.initializers.Zeros()))
model.compile(tf.keras.optimizers.SGD(0.01), 'mse')

hist = model.fit(wins, labels, batch_size = 1, epochs = 30, shuffle = True)
```

```
Epoch 1/30
346/346 [==============================] - 0s 771us/step - loss: 0.0462
Epoch 2/30
346/346 [==============================] - 0s 727us/step - loss: 0.0175
Epoch 3/30
346/346 [==============================] - 0s 1ms/step - loss: 0.0103
```

```
Epoch 4/30
346/346 [==============================] - 0s 1ms/step - loss: 0.0066
Epoch 5/30
346/346 [==============================] - 0s 711us/step - loss: 0.0043
Epoch 6/30
346/346 [==============================] - 0s 754us/step - loss: 0.0032
Epoch 7/30
346/346 [==============================] - 0s 783us/step - loss: 0.0025
Epoch 8/30
346/346 [==============================] - 0s 715us/step - loss: 0.0022
Epoch 9/30
346/346 [==============================] - 0s 744us/step - loss: 0.0020
Epoch 10/30
346/346 [==============================] - 0s 765us/step - loss: 0.0019
Epoch 11/30
346/346 [==============================] - 0s 1ms/step - loss: 0.0019
Epoch 12/30
346/346 [==============================] - 0s 719us/step - loss: 0.0018
Epoch 13/30
346/346 [==============================] - 0s 764us/step - loss: 0.0017
Epoch 14/30
346/346 [==============================] - 0s 785us/step - loss: 0.0017
Epoch 15/30
346/346 [==============================] - 0s 725us/step - loss: 0.0017
Epoch 16/30
346/346 [==============================] - 0s 754us/step - loss: 0.0017
Epoch 17/30
346/346 [==============================] - 0s 759us/step - loss: 0.0017
Epoch 18/30
346/346 [==============================] - 0s 795us/step - loss: 0.0017
Epoch 19/30
346/346 [==============================] - 0s 1ms/step - loss: 0.0016
Epoch 20/30
346/346 [==============================] - 0s 1ms/step - loss: 0.0017
Epoch 21/30
346/346 [==============================] - 0s 1ms/step - loss: 0.0017
Epoch 22/30
346/346 [==============================] - 0s 1ms/step - loss: 0.0016
Epoch 23/30
346/346 [==============================] - 0s 1ms/step - loss: 0.0016
Epoch 24/30
346/346 [==============================] - 0s 1ms/step - loss: 0.0016
Epoch 25/30
346/346 [==============================] - 0s 1ms/step - loss: 0.0016
Epoch 26/30
346/346 [==============================] - 0s 1ms/step - loss: 0.0016
Epoch 27/30
346/346 [==============================] - 1s 2ms/step - loss: 0.0016
Epoch 28/30
```

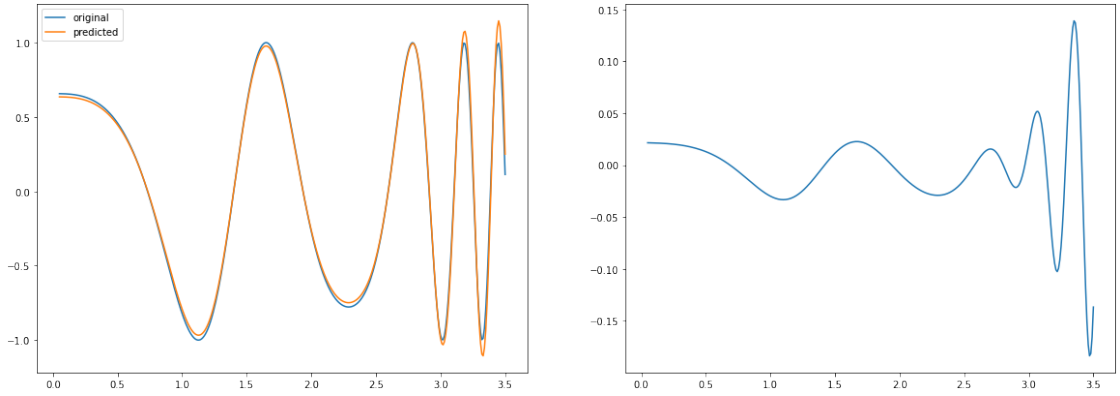
```
346/346 [==============================] - 0s 1ms/step - loss: 0.0016
Epoch 29/30
346/346 [==============================] - 1s 2ms/step - loss: 0.0016
Epoch 30/30
346/346 [==============================] - 1s 2ms/step - loss: 0.0015
```

[ ]:
```
fig, ax = pylab.subplots(1, 1, figsize = (10, 7))
histx = []
for i in range(len(hist.history['loss'])):
    histx.append(i)


plt.plot(histx, hist.history['loss'])
plt.show()
```



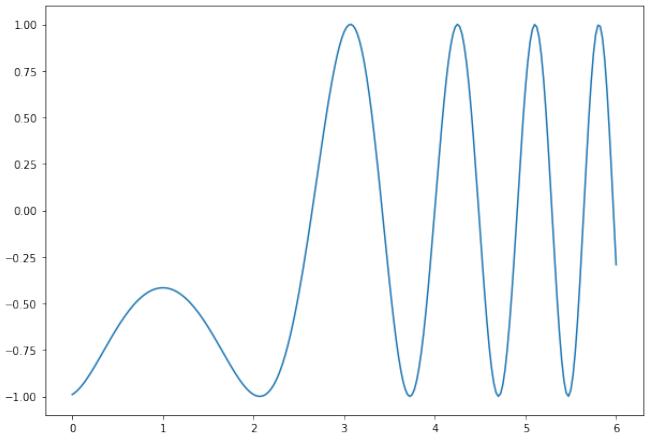
[ ]:
```
pred = model.predict(wins)
pred2 = pred.flat
orig = f1(t[winsize:])

err = orig - pred.flat


fig = plt.figure(figsize = (20, 7))

ax1 = fig.add_subplot(1, 2, 1)
ax2 = fig.add_subplot(1, 2, 2)


ax1.plot(t[winsize:], orig, label = 'original')
ax1.plot(t[winsize:], pred.flat, label = 'predicted')
```

```
ax2.plot(t[winsize:], err)

ax1.legend()

plt.show()
```

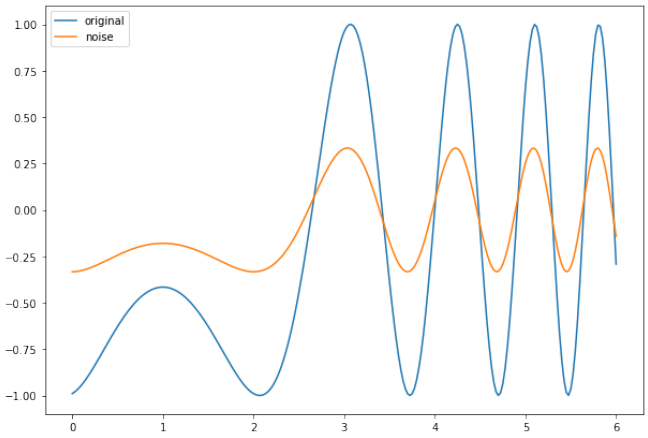


[ ]:
```
def f2(t):
    return np.cos(t * t - 2 * t + 3)

def f3(t):
    return 1/3 * np.cos(t * t - 2 * t - np.pi)
```

[ ]:
```
fig, ax = pylab.subplots(1, 1, figsize = (10, 7))
t =  np.arange(0, 6.025, 0.025)
plt.plot(t, f2(t))
plt.show()
```



[ ]:
```
fig, ax = pylab.subplots(1, 1, figsize = (10, 7))
plt.plot(t, f2(t), label = 'original')
plt.plot(t, f3(t), label = 'noise')
plt.legend()
plt.show()
```



[ ]:
```
winsize = 5
wins, labels = get_windows(winsize, t, f3, f3)
```

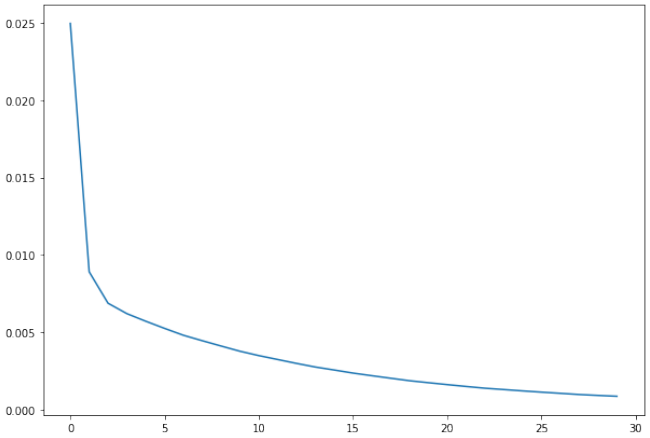
[ ]:
```
model = keras.models.Sequential()

model.add(Dense(1, activation = "linear",  kernel_initializer = keras.initializers.
 ↪RandomNormal(stddev=0.01), bias_initializer = keras.initializers.Zeros()))
model.compile(tf.keras.optimizers.SGD(0.01), 'mse')

hist = model.fit(wins, labels, batch_size = 1, epochs = 30, shuffle = True)
```

```
Epoch 1/30
236/236 [==============================] - 0s 738us/step - loss: 0.0250
Epoch 2/30
236/236 [==============================] - 0s 780us/step - loss: 0.0089
Epoch 3/30
236/236 [==============================] - 0s 735us/step - loss: 0.0069
Epoch 4/30
236/236 [==============================] - 0s 797us/step - loss: 0.0062
Epoch 5/30
236/236 [==============================] - 0s 745us/step - loss: 0.0057
Epoch 6/30
236/236 [==============================] - 0s 714us/step - loss: 0.0052
Epoch 7/30
236/236 [==============================] - 0s 730us/step - loss: 0.0048
Epoch 8/30
```

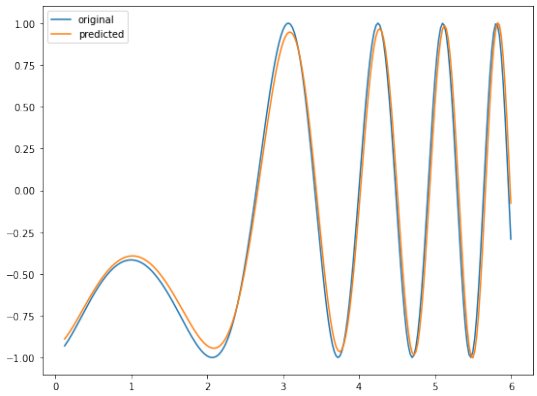
```
236/236 [==============================] - 0s 724us/step - loss: 0.0044
Epoch 9/30
236/236 [==============================] - 0s 720us/step - loss: 0.0041
Epoch 10/30
236/236 [==============================] - 0s 733us/step - loss: 0.0038
Epoch 11/30
236/236 [==============================] - 0s 797us/step - loss: 0.0035
Epoch 12/30
236/236 [==============================] - 0s 725us/step - loss: 0.0032
Epoch 13/30
236/236 [==============================] - 0s 936us/step - loss: 0.0030
Epoch 14/30
236/236 [==============================] - 0s 965us/step - loss: 0.0027
Epoch 15/30
236/236 [==============================] - 0s 725us/step - loss: 0.0026
Epoch 16/30
236/236 [==============================] - 0s 855us/step - loss: 0.0024
Epoch 17/30
236/236 [==============================] - 0s 735us/step - loss: 0.0022
Epoch 18/30
236/236 [==============================] - 0s 759us/step - loss: 0.0020
Epoch 19/30
236/236 [==============================] - 0s 729us/step - loss: 0.0019
Epoch 20/30
236/236 [==============================] - 0s 770us/step - loss: 0.0017
Epoch 21/30
236/236 [==============================] - 0s 717us/step - loss: 0.0016
Epoch 22/30
236/236 [==============================] - 0s 801us/step - loss: 0.0015
Epoch 23/30
236/236 [==============================] - 0s 763us/step - loss: 0.0014
Epoch 24/30
236/236 [==============================] - 0s 777us/step - loss: 0.0013
Epoch 25/30
236/236 [==============================] - 0s 718us/step - loss: 0.0012
Epoch 26/30
236/236 [==============================] - 0s 807us/step - loss: 0.0011
Epoch 27/30
236/236 [==============================] - 0s 734us/step - loss: 0.0010
Epoch 28/30
236/236 [==============================] - 0s 1ms/step - loss: 9.7197e-04
Epoch 29/30
236/236 [==============================] - 0s 1ms/step - loss: 9.1299e-04
Epoch 30/30
236/236 [==============================] - 0s 1ms/step - loss: 8.5889e-04
```

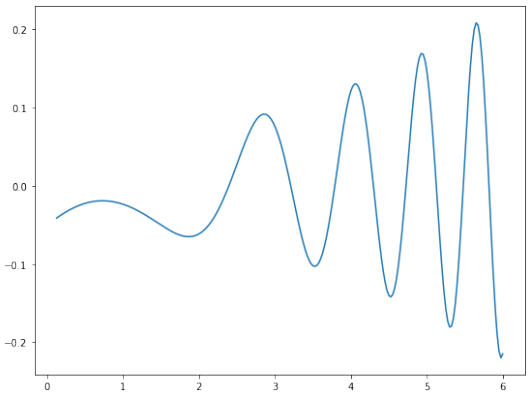
[ ]:
```
fig, ax = pylab.subplots(1, 1, figsize = (10, 7))
histx = []
for i in range(len(hist.history['loss'])):
```

```
    histx.append(i)


plt.plot(histx, hist.history['loss'])
plt.show()
```



[ ]:
```
wins, labels = get_windows(winsize, t, f2, f3)

pred = model.predict(wins)
orig = f2(t[winsize:])

err = orig - pred.flat


fig = plt.figure(figsize = (20, 7))

ax1 = fig.add_subplot(1, 2, 1)
ax2 = fig.add_subplot(1, 2, 2)


ax1.plot(t[winsize:], orig, label = 'original')
ax1.plot(t[winsize:], pred.flat, label = 'predicted')


ax2.plot(t[winsize:], err)

ax1.legend()

plt.show()
```





Выводы

В ходе выполнения лабораторной работы я познакомился с задачами аппроксимации и фильтрации функции, а также обучил и применил однослойную нейронную сеть для решения этих задач.

[ ]: