Нейроинформатика. Лабораторная работа 3

Линейная нейронная сеть. Правило обучения Уидроу-Хоффа

Целью работы является исследование свойств многослойной нейронной сети прямого распространения и алгоритмов ее обучения, применение сети в задачах классификации и аппроксимации функции.

Выполнил Лисин Роман, М8О-406Б-20. Вариант 12.

```
import numpy as np

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

import matplotlib.pyplot as plt
```

Задание 1

Попробуем применить многослойный перцептрон для классификации линейно неразделимых множеств

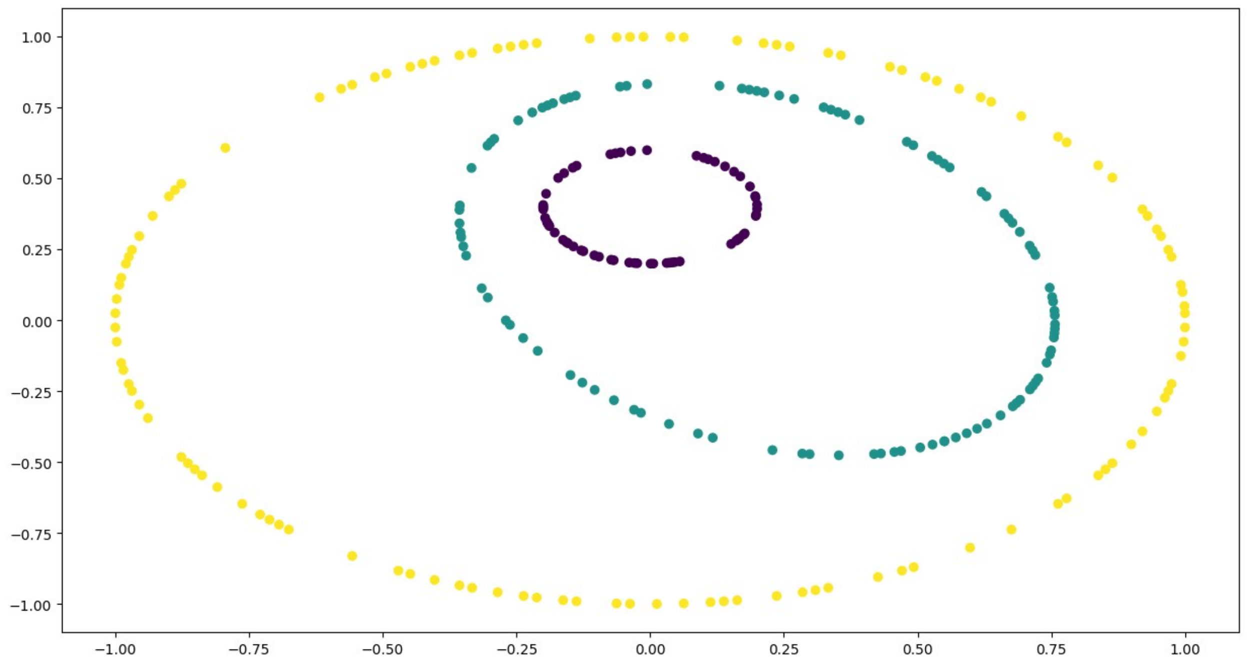
Сгенерируем датасет. Датасет будет состоять из точек, принадлежащим границам трех эллипсов

```
ellipse0 = dict(
    a = 0.2,
    b = 0.2,
    alpha = np.pi / 3,
    x0 = 0,
    y0 = 0.4,
    label = 0,
)

ellipse1 = dict(
    a = 0.7,
    b = 0.5,
    alpha = -np.pi / 3,
    x0 = 0.2,
    y0 = 0.18,
    label = 1,
)

ellipse2 = dict(
    a = 1,
    b = 1,
    alpha = 0,
    x0 = 0,
    y0 = 0,
    label = 2,
)

def gen_dataset(ellipses):
    t = np.linspace(0, 2 * np.pi, int(2 * np.pi / 0.025))

    points = np.array([
        [
            ellipses[i]['a'] * np.cos(t) * np.cos(ellipses[i]
['alpha']) \
            - ellipses[i]['b'] * np.sin(t) * np.sin(ellipses[i]
['alpha']) \
            + ellipses[i]['x0'],

            ellipses[i]['a'] * np.cos(t) * np.sin(ellipses[i]
['alpha']) \
            + ellipses[i]['b'] * np.sin(t) * np.cos(ellipses[i]
['alpha']) \
            + ellipses[i]['y0'],

            np.tile(ellipses[i]['label'], len(t)),

         ] for i in range(len(ellipses))])

    np.random.seed(0xDEAD)

    el_0 = points[0, :, np.random.choice(len(t), 60, replace=False)]
    el_1 = points[1, :, np.random.choice(len(t), 100, replace=False)]
    el_2 = points[2, :, np.random.choice(len(t), 120, replace=False)]

    data = np.vstack((el_0, el_1, el_2))
    np.random.shuffle(data)

    return data

data1 = gen_dataset([ellipse0, ellipse1, ellipse2])

data1.shape
```

```
(280, 3)
```

Посмотрим на получившийся датасет

```
plt.figure(figsize=(15, 8))

plt.scatter(data1[:, 0], data1[:, 1], c=data1[:, 2])

plt.show()
```



Поделим данные на трейн, тест и валидацию

```
train, val, test = np.split(data1, [int(.7*len(data1)),
int(.9*len(data1))])

train.shape, val.shape, test.shape

((196, 3), (56, 3), (28, 3))
```

Отделим фичи от таргетов. X - фичи, y - таргеты

```
X_train = train[:, :2]
y_train = train[:, 2]

X_test = test[:, :2]
y_test = test[:, 2]

X_val = val[:, :2]
y_val = val[:, 2]
```

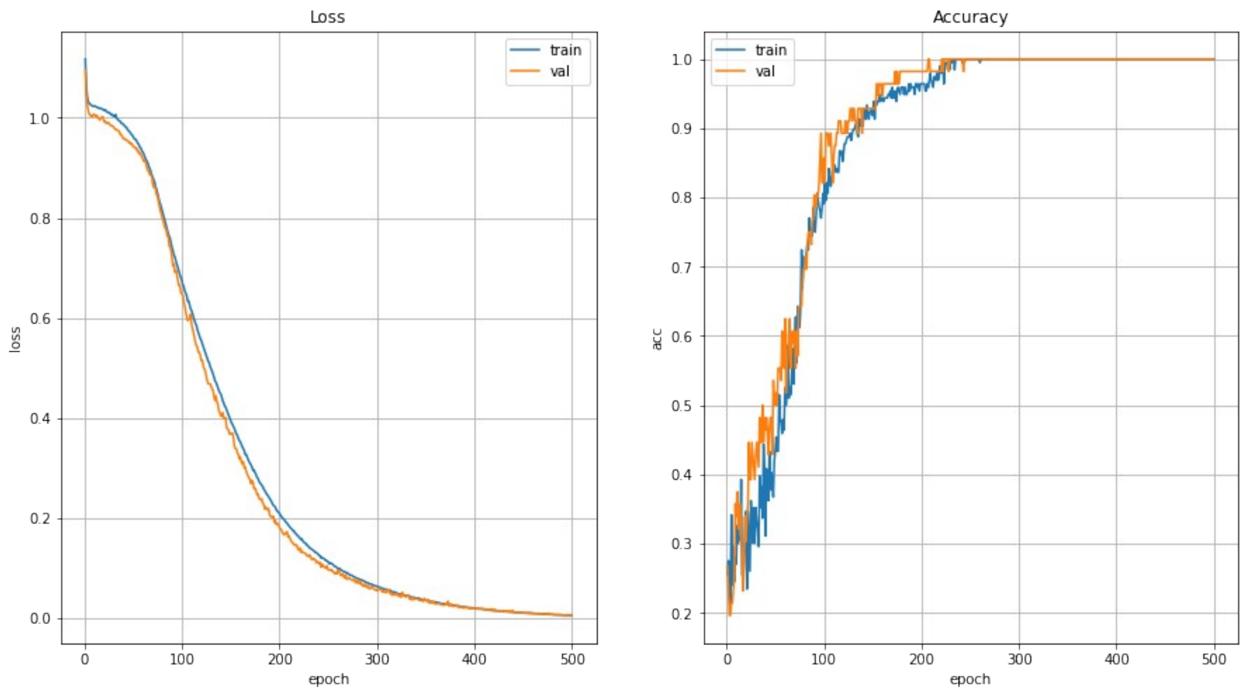
Будем использовать двухслойный перцептрон. В скрытом слое по заданию будет 20 нейронов. В качестве функции активации будем использовать танх, алгоритм обучения - RMSProp

```
model1 = keras.Sequential([
    keras.layers.Dense(20, activation='tanh'),
    keras.layers.Dense(3, activation='softmax'),
])
```

```
model1.compile(
    loss='sparse_categorical_crossentropy',
    optimizer='Adam',
    metrics='accuracy'
)

train_info1 = model1.fit(
    X_train,
    y_train,
    batch_size=1,
    epochs=500,
    validation_data=(X_val, y_val),
    shuffle=True,
    verbose=0
)
```

Посмотрим на графики

```
def plot_metrics(train_info):
    plt.figure(figsize=(15, 8))

    plt.subplot(1, 2, 1)
    loss_history = train_info.history['loss']
    val_loss_history = train_info.history['val_loss']
    plt.xlabel('epoch')
    plt.ylabel('loss')
    plt.plot(range(1, len(loss_history) + 1), loss_history,
label='train')
    plt.plot(range(1, len(loss_history) + 1), val_loss_history,
label='val')
    plt.grid()
    plt.legend()
    plt.title('Loss')

    plt.subplot(1, 2, 2)
    acc_history = train_info.history['accuracy']
    val_acc_history = train_info.history['val_accuracy']
    plt.xlabel('epoch')
    plt.ylabel('acc')
    plt.plot(range(1, len(acc_history) + 1), acc_history,
label='train')
    plt.plot(range(1, len(val_acc_history) + 1), val_acc_history,
label='val')
    plt.grid()
    plt.legend()
    plt.title('Accuracy')

    plt.show()

plot_metrics(train_info1)
```



```
print("Val accuracy =", train_info1.history['val_accuracy'][-1])
```

```
Val accuracy = 1.0
```

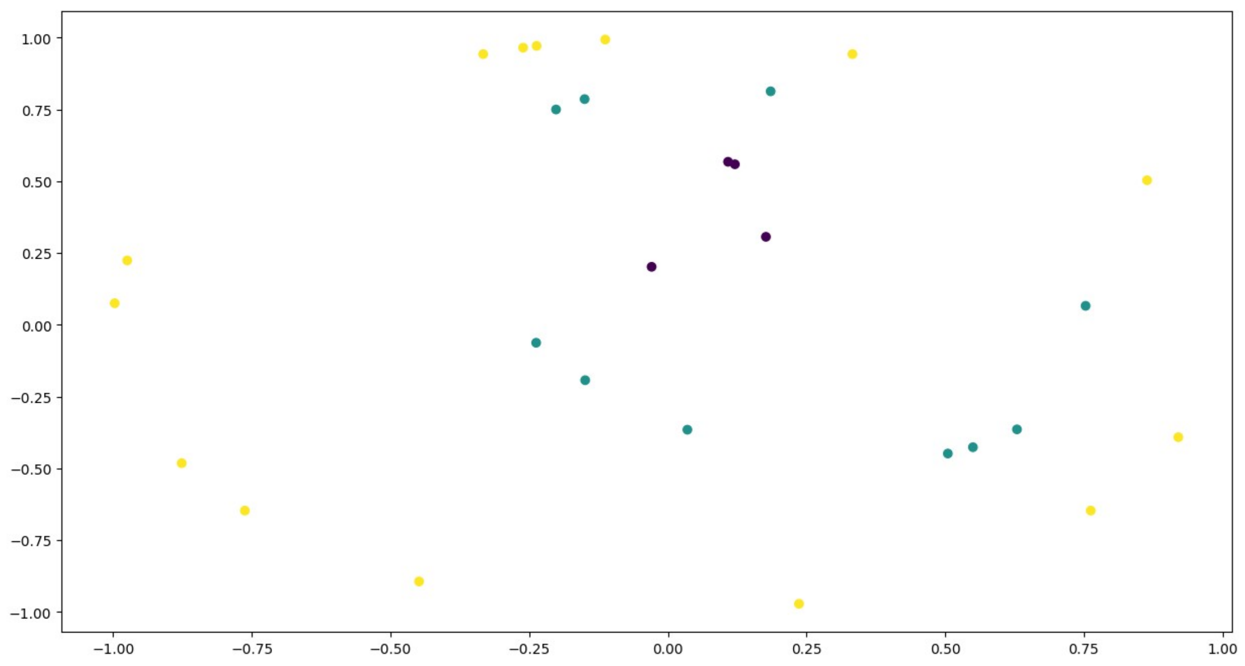
Получили очень идеальную точность на валидации.

Проверим, что происходит с тестовой выборкой.

```
np.argmax(model1.predict(X_test, verbose=0), axis=1).shape
```

```
(28,)
```

```
plt.figure(figsize=(15, 8))

plt.scatter(X_test[:, 0], X_test[:, 1],
c=np.argmax(model1.predict(X_test, verbose=0), axis=1))

plt.show()
```



```
print("Test accuracy =", (np.argmax(model1.predict(X_test, verbose=0),
axis=1) == y_test).mean())
```

```
Test accuracy = 1.0
```

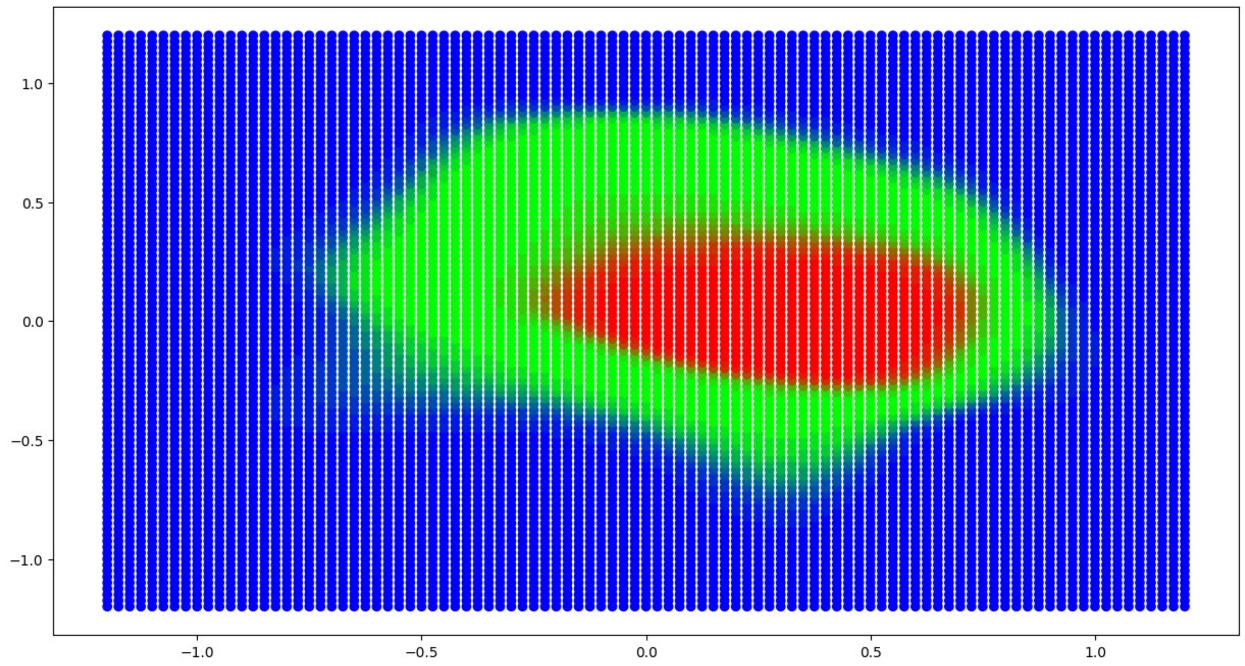
Все точки на эллипсах классифицированы верно.

Построим на классификацию точек во всей области

```
h = 0.025

grid = [model1.predict(np.array([[i, j]]), verbose=0).round(1)
        for i in np.arange(-1.2, 1.2 + h, h)
        for j in np.arange(-1.2, 1.2 + h, h)]

x_vals = np.arange(-1.2, 1.2 + h, h)
y_vals = np.arange(-1.2, 1.2 + h, h)

xx, yy = np.meshgrid(x_vals, y_vals)

colors = np.array(grid).reshape((len(grid), 3))
colors.shape
```

```
(9409, 3)
```

```
plt.figure(figsize=(15, 8))

plt.scatter(xx, yy, c=colors)

plt.show()
```



Вырисовывается картинка, похожая на изначальный датасет с эллипсами

Задание 2

Пробуем аппроксимировать функцию многослойной сетью с помощью методов первого порядка

```
def fun(t):
    return np.cos(-np.cos(t) * t * t + t)

range_t = (0.5, 4)
h = 0.01
```

Подготовим датасет:

```
t = np.linspace(range_t[0], range_t[1], int((range_t[1] - range_t[0])
/ h))
x = fun(t)
```

Поделим на трейн и вал

```
train_len = int(t.shape[0] * 0.9)

t_train = t[:train_len]
t_val = t[train_len:]

x_train = x[:train_len]
x_val = x[train_len:]
```

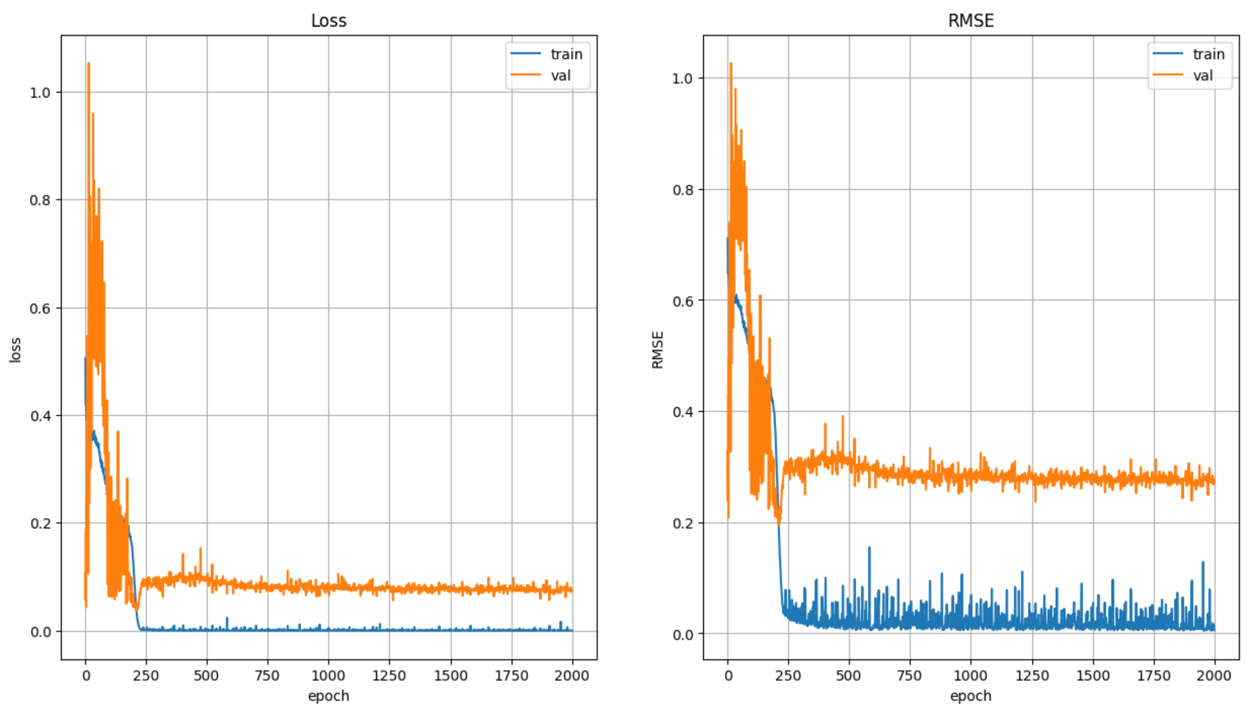
```
t_train = np.expand_dims(t_train, 1)
t_val = np.expand_dims(t_val, 1)

t_train.shape, t_val.shape

((315, 1), (35, 1))
```

Обучим модель. В качестве алгоритма обучения возьмем Adam (метод оптимизации 1 порядка)

```
model2 = keras.Sequential([
    keras.layers.Dense(64, activation='tanh'),
    keras.layers.Dense(32, activation='tanh'),
    keras.layers.Dense(1),
])

model2.compile(
    loss='mse',
    optimizer='Adam',
    metrics=tf.keras.metrics.RootMeanSquaredError(),
)

train_info2 = model2.fit(
    t_train,
    x_train,
    batch_size=4,
    epochs=2000,
    validation_data=(t_val, x_val),
    verbose=0
)
```

Посмотрим на лосс и метрики

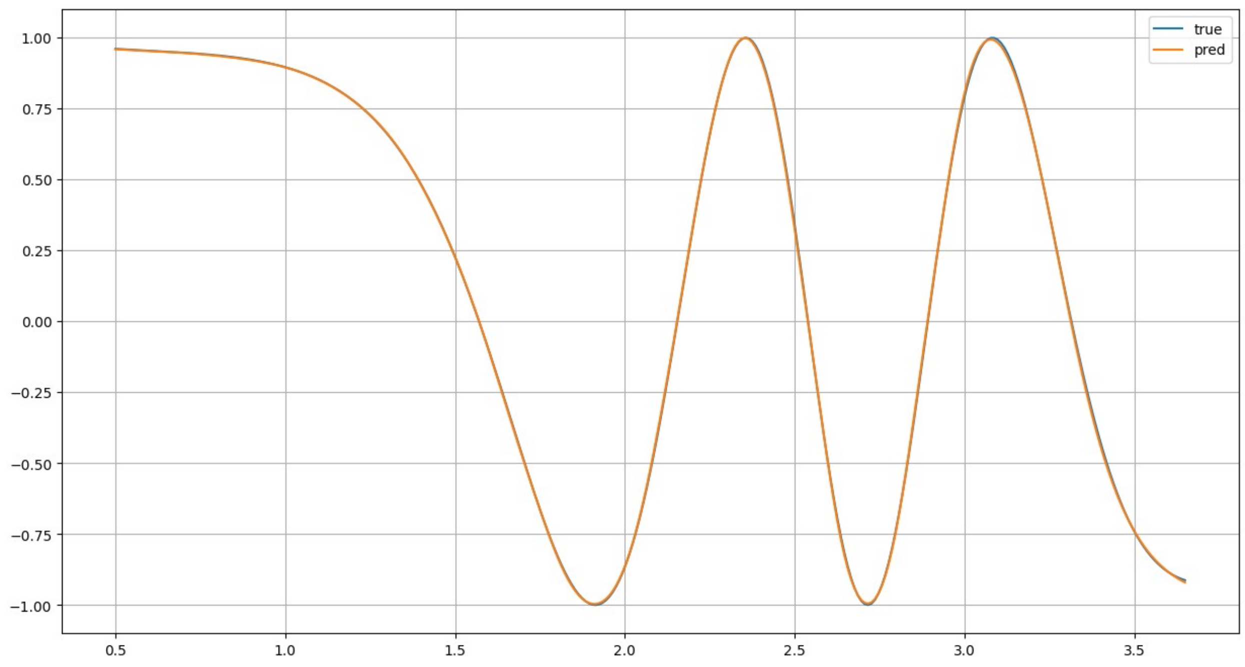
```
def plot_metrics2(train_info):
    plt.figure(figsize=(15, 8))

    plt.subplot(1, 2, 1)
    loss_history = train_info.history['loss']
    val_loss_history = train_info.history['val_loss']
    plt.xlabel('epoch')
    plt.ylabel('loss')
    plt.plot(range(1, len(loss_history) + 1), loss_history,
label='train')
    plt.plot(range(1, len(loss_history) + 1), val_loss_history,
label='val')
    plt.grid()
    plt.legend()
    plt.title('Loss')

    plt.subplot(1, 2, 2)
    acc_history = train_info.history['root_mean_squared_error']
    val_acc_history =
train_info.history['val_root_mean_squared_error']
    plt.xlabel('epoch')
    plt.ylabel('RMSE')
    plt.plot(range(1, len(acc_history) + 1), acc_history,
label='train')
    plt.plot(range(1, len(val_acc_history) + 1), val_acc_history,
label='val')
    plt.grid()
    plt.legend()
    plt.title('RMSE')

    plt.show()

plot_metrics2(train_info2)
```



```
print("Val RMSE =", train_info2.history['val_root_mean_squared_error']
[-1])

Val RMSE = 0.27449116110801697
```

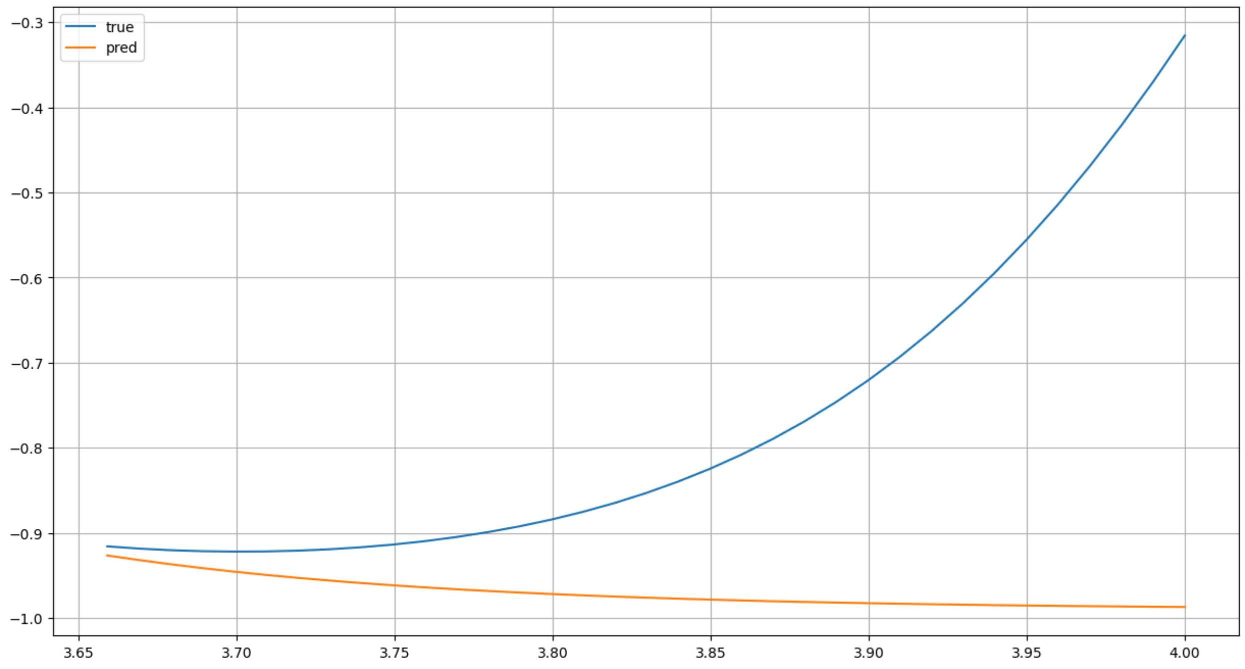
Посмотрим на результаты на трейне

```
def plot_results(model, t, fun):
    plt.figure(figsize=(15, 8))
    plt.plot(t, fun(t), label='true')
    plt.plot(t, model.predict(t), label='pred')

    plt.grid()
    plt.legend()
    plt.show()

plot_results(model2, t_train, fun)

10/10 [==============================] - 0s 2ms/step
```



И на вале

```
plot_results(model2, t_val, fun)

2/2 [==============================] - 0s 5ms/step
```



С правой частью кривой (и с валидацией в том числе) модель справляется не очень хорошо

Задание 3

Пробуем аппроксимировать функцию многослойной сетью с помощью методов второго порядка.

Так как в керасе нет методов оптимизации второго порядка, то перейдем на фреймворк neupy. В качестве алгоритма обучения будем использовать алгоритм Ньютона (метод второго порядка). Все остальное будет аналогично заданию 2

```
!pip install neupy

Collecting neupy
  Downloading neupy-0.8.2-py2.py3-none-any.whl (226 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 226.8/226.8 kB 2.8 MB/s eta
0:00:00
ent already satisfied: numpy>=1.13.3 in
/usr/local/lib/python3.10/dist-packages (from neupy) (1.23.5)
Requirement already satisfied: scipy>=0.19.0 in
/usr/local/lib/python3.10/dist-packages (from neupy) (1.11.3)
INFO: pip is looking at multiple versions of neupy to determine which
version is compatible with other requirements. This could take a
while.
  Downloading neupy-0.8.1-py2.py3-none-any.whl (225 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 226.0/226.0 kB 19.5 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 224.7/224.7 kB 21.7 MB/s eta
0:00:00
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 212.3/212.3 kB 18.7 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 208.5/208.5 kB 21.9 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 208.0/208.0 kB 19.5 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 198.6/198.6 kB 23.0 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 197.2/197.2 kB 22.2 MB/s eta
0:00:00
 neupy)
  Downloading Theano-1.0.0.tar.gz (2.9 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.9/2.9 MB 6.1 MB/s eta
0:00:00
etadata (setup.py) ... ent already satisfied: matplotlib>=1.5.1 in
/usr/local/lib/python3.10/dist-packages (from neupy) (3.7.1)
Collecting graphviz==0.5.1 (from neupy)
  Downloading graphviz-0.5.1-py2.py3-none-any.whl (14 kB)
Collecting tableprint==0.7.1 (from neupy)
  Downloading tableprint-0.7.1.tar.gz (6.7 kB)
  Preparing metadata (setup.py) ...  neupy)
  Downloading progressbar2-3.34.3-py2.py3-none-any.whl (25 kB)
Requirement already satisfied: python-utils>=2.1.0 in
/usr/local/lib/python3.10/dist-packages (from progressbar2==3.34.3-
>neupy) (3.8.1)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-
packages (from tableprint==0.7.1->neupy) (1.16.0)
Requirement already satisfied: future in
/usr/local/lib/python3.10/dist-packages (from tableprint==0.7.1-
>neupy) (0.18.3)
Requirement already satisfied: wcwidth in
/usr/local/lib/python3.10/dist-packages (from tableprint==0.7.1-
>neupy) (0.2.8)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=1.5.1-
>neupy) (1.1.1)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=1.5.1-
>neupy) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=1.5.1-
>neupy) (4.43.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=1.5.1-
>neupy) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=1.5.1-
>neupy) (23.2)
Requirement already satisfied: pillow>=6.2.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from matplotlib>=1.5.1-
>neupy) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=1.5.1-
>neupy) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=1.5.1-
>neupy) (2.8.2)
Requirement already satisfied: typing-extensions>3.10.0.2 in
/usr/local/lib/python3.10/dist-packages (from python-utils>=2.1.0-
>progressbar2==3.34.3->neupy) (4.5.0)
Building wheels for collected packages: tableprint, Theano
  Building wheel for tableprint (setup.py) ... e=tableprint-0.7.1-py3-
none-any.whl size=6163
sha256=d8ab11a2bc589bb6790eafa85675aa75176146737e0a356bd4e67a1251671b3
1
  Stored in directory:
/root/.cache/pip/wheels/89/f4/89/e6e577409bbc25fed6dedef9b60b60df263bd
4ce1cd5c222d5
  Building wheel for Theano (setup.py) ... e=Theano-1.0.0-py3-none-
any.whl size=2649592
sha256=9d45628a86196c0efe8b0b542d2a73f993cf5309d09a0f73596c2bb41c9494a
2
  Stored in directory:
/root/.cache/pip/wheels/d9/06/e7/b742d72dba1f1896f21519bcaf138ee5123f8
e8e0cf424b382
Successfully built tableprint Theano
Installing collected packages: graphviz, tableprint, Theano,
progressbar2, neupy
  Attempting uninstall: graphviz
    Found existing installation: graphviz 0.20.1
    Uninstalling graphviz-0.20.1:
      Successfully uninstalled graphviz-0.20.1
  Attempting uninstall: progressbar2
    Found existing installation: progressbar2 4.2.0
    Uninstalling progressbar2-4.2.0:
      Successfully uninstalled progressbar2-4.2.0
Successfully installed Theano-1.0.0 graphviz-0.5.1 neupy-0.6.5
progressbar2-3.34.3 tableprint-0.7.1
```

Так как фреймворк neupy не поддерживается на версии Python 3.10 (в Google Colab только такой Python, попытки поменять версию питона в Colab не увенчались успехом), то будем использовать Python 3.6 на компьютере.

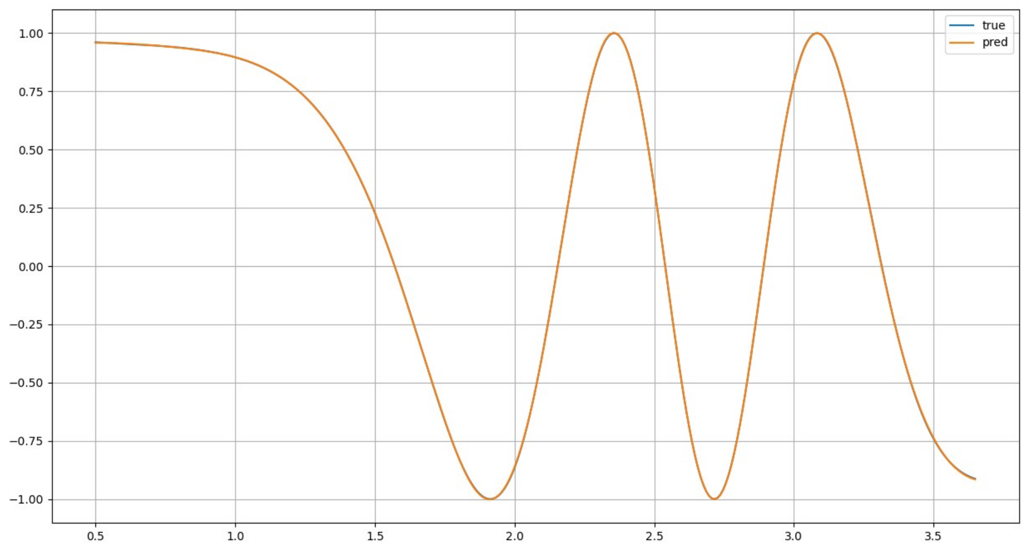
```
Python 3.6.12 (default, Oct 19 2023, 18:45:21)
[GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from neupy import algorithms, layers
/home/roma/.local/lib/python3.6/site-packages/tensorflow/python/framew
```

```
ork/dtypes.py:526: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/home/roma/.local/lib/python3.6/site-packages/tensorflow/python/framew
ork/dtypes.py:527: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/home/roma/.local/lib/python3.6/site-packages/tensorflow/python/framew
ork/dtypes.py:528: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/home/roma/.local/lib/python3.6/site-packages/tensorflow/python/framew
ork/dtypes.py:529: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/home/roma/.local/lib/python3.6/site-packages/tensorflow/python/framew
ork/dtypes.py:530: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/home/roma/.local/lib/python3.6/site-packages/tensorflow/python/framew
ork/dtypes.py:535: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
>>> model3 = algorithms.Hessian(
    (
        layers.Input(1),
        layers.Tanh(10),
        layers.Tanh(5),
        layers.Linear(1),
    ),
    verbose=True,
    show_epoch=1000,
)

Main information

[ALGORITHM] Hessian

[OPTION] loss = mse
[OPTION] penalty_const = 1
[OPTION] regularizer = None
[OPTION] show_epoch = 1000
[OPTION] shuffle_data = False
[OPTION] signals = None
```

```
[OPTION] target = Tensor("placeholder/target/linear-1:0", shape=(?,
1), dtype=float32)
[OPTION] verbose = True

[TENSORFLOW] Initializing Tensorflow variables and functions.
WARNING:tensorflow:From /home/roma/.local/lib/python3.6/site-
packages/tensorflow/python/framework/op_def_library.py:263:
colocate_with (from tensorflow.python.framework.ops) is deprecated and
will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
2023-10-22 15:55:14.322147: I
tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports
instructions that this TensorFlow binary was not compiled to use: AVX2
FMA
2023-10-22 15:55:14.332645: I
tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU Frequency:
1796640000 Hz
2023-10-22 15:55:14.334491: I
tensorflow/compiler/xla/service/service.cc:150] XLA service
0x56067bc34c40 executing computations on platform Host. Devices:
2023-10-22 15:55:14.334560: I
tensorflow/compiler/xla/service/service.cc:158]   StreamExecutor
device (0): <undefined>, <undefined>
WARNING:tensorflow:From /home/roma/.local/lib/python3.6/site-
packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from
tensorflow.python.ops.math_ops) is deprecated and will be removed in a
future version.
Instructions for updating:
Use tf.cast instead.
[TENSORFLOW] Initialization finished successfully. It took 0.33
seconds
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> def fun(t):
    return np.cos(-np.cos(t) * t * t + t)
...
>>> range_t = (0.5, 4)
>>> h = 0.01
>>> t = np.linspace(range_t[0], range_t[1], int((range_t[1] -
range_t[0]) / h))
>>> x = fun(t)
>>> train_len = int(t.shape[0] * 0.9)
>>> t_train = t[:train_len]
>>> t_val = t[train_len:]
>>> x_train = x[:train_len]
>>> x_val = x[train_len:]
>>> t_train = np.expand_dims(t_train, 1)
>>> t_val = np.expand_dims(t_val, 1)
>>> t_train.shape, t_val.shape
((315, 1), (35, 1))
>>> model3.train(t_train, x_train, t_val, x_val, epochs=15000)
#1 : [187 ms] train: 0.564537, valid: 0.637994
#1000 : [13 ms] train: 0.171533, valid: 3.230381
#2000 : [14 ms] train: 0.000514, valid: 0.118814
#3000 : [15 ms] train: 0.000220, valid: 0.124973
#4000 : [14 ms] train: 0.000151, valid: 0.113195
#5000 : [15 ms] train: 0.000103, valid: 0.089933
#6000 : [15 ms] train: 0.000064, valid: 0.066262
#7000 : [14 ms] train: 0.000039, valid: 0.048536
#8000 : [15 ms] train: 0.000024, valid: 0.036590
#9000 : [14 ms] train: 0.000015, valid: 0.028699
#10000 : [15 ms] train: 0.000010, valid: 0.023464
#11000 : [14 ms] train: 0.000008, valid: 0.019952
#12000 : [14 ms] train: 0.000006, valid: 0.017553
#13000 : [14 ms] train: 0.000005, valid: 0.015880
#14000 : [14 ms] train: 0.000005, valid: 0.014681
#15000 : [18 ms] train: 0.000004, valid: 0.013798
```

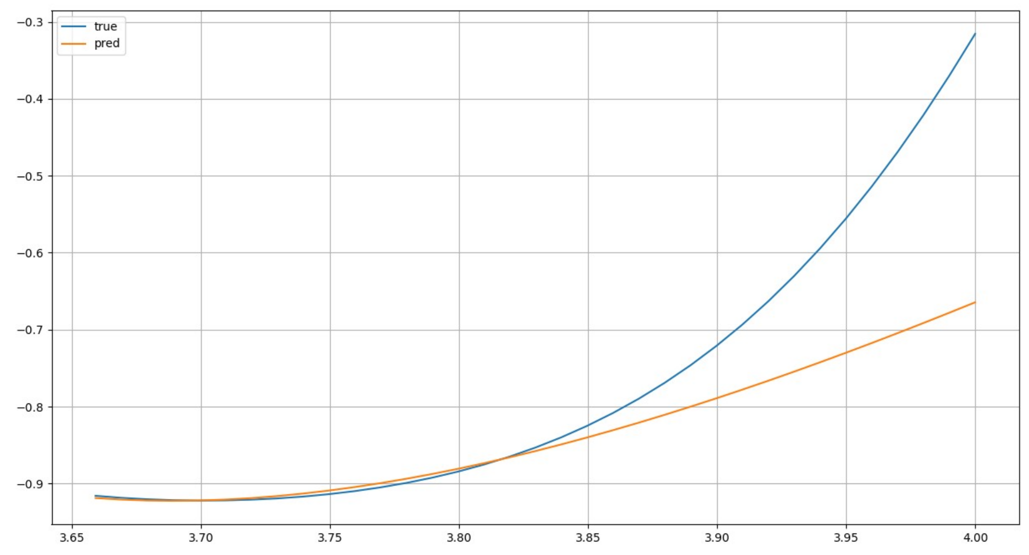
Проверим качество модели на трейне

```
>>> def plot_results(model, t, fun):
...     plt.figure(figsize=(15, 8))
...     plt.plot(t, fun(t), label='true')
...     plt.plot(t, model.predict(t), label='pred')
...     plt.grid()
...     plt.legend()
...     plt.show()
...
>>> plot_results(model3, t_train, fun)
```



И на вале

```
>>> plot_results(model3, t_val, fun)
```



```
>>> t_val_fl = t_val.flatten()
>>> rmse_val = np.sqrt(np.mean((model3.predict(t_val).flatten() -
fun(t_val_fl))**2))
>>> print("Val RMSE =", rmse_val)
Val RMSE = 0.11746344485153609
```

Модель, обученная с помощью алгоритма второго порядка справилась с задачей лучше.

Вывод

В этой работе я потренировался в работе с многослойными нейросетями. С их помощью я смог хорошо решить задачу классификации на 3 линейно неразделимых класса, а также попробовал аппроксимировать нелинейную функцию.

Для аппроксимации функции я воспользовался двумя алгоритмами обучения - Адам (метод первого порядка) и метод Ньютона (второго порядка). Метод второго порядка отработал лучше.

Но нельзя сказать, что мы получили хороший результат. Ни одна модель не смогла начать закругление на валидационной выборке - все они продолжали строить функцию по прямой