

Правительство Российской Федерации
Федеральное государственное автономное образовательное
учреждение
высшего образования

«Национальный исследовательский университет
«Высшая школа экономики»
Московский институт электроники и математики

Департамент компьютерной инженерии

Лабораторная работа по курсу
“Математический компьютерный практикум”
Тема :«Машинное обучение»
Вариант 3

Выполнил:

Студент группы БИВ184

Каграманян Давид Геворгович

Задача классификации

Для решения этой задачи существует множество подходов. В данной работе разберем 3 из них.

- 1) Нейронная сеть многослойный персептрон с дополнительным нормальным шумом
- 2) Наивный классификатор Байеса
- 3) Рандомное дерево решений

Для начала работы необходимо считать данные из txtфайлов. Затем представить их в виде многомерных массивов и отнормализовать. (нужно вычесть среднее значение по столбцу и поделить на дисперсию по столбцу).

В результате получены данные:

- 1) Нейронная сеть обучается довольно долго (40 эпох минимум), однако ее точность (доля верных ответов) на тестовой выборке выше всех – 0.98. Нормальный шум добавлен для того, чтобы модель не переобучилась на обучающей выборке.
- 2) Наивный классификатор байеса с ядром гаусса и бернулли показали средний результат – 0,93 и 0,6 соответственно, однако у них есть сильная сторона – высокая скорость обучения и простота реализации
- 3) Дерево рандомных решений показало на тестовой выборке 0,95. Что в целом вполне оптимально по соотношению к точности и скорости обучения.

Выводы: рандомное дерево решений – оптимальный вариант для решения поставленной задачи.

```
In [1]: import pandas as pd
import numpy as np
import re
from keras.layers import Dense, GaussianNoise, LeakyReLU, Activation
from keras.models import Sequential
import matplotlib.pyplot as plt
from sklearn.naive_bayes import GaussianNB, BernoulliNB, MultinomialNB, CategoricalNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
```

Using TensorFlow backend.

Проведем предобработку данных

```
In [2]: train=open('train.txt','r')
train = train.read().splitlines()
test=open('test.txt','r')
test=test.read().splitlines()
```

Так как самая первая строка - индексы, то удаляем ее

Как видно, первые строки - числа

```
In [3]: train.pop(0)
test.pop(0)
print(train[0])
print(test[0])

1 -0.335467047065564 0.102235638179093 red
1 -2.24246004235479 0.47048871459103 green
```

Создаем из считанных файлов массивы

```
In [4]: train_data=np.array(train)
test_data=np.array(test)
```

Создаем пустую обучающую выборку, red - 0, green - 1

```
In [5]: width=2
train_length=train_data.shape[0]
test_length=test_data.shape[0]

x_train=np.ones((train_length,width))
y_train=np.ones((train_length,1))
x_test=np.ones((test_length,width))
y_test=np.ones((test_length,1))
```

Создаем на основе считанных файлов выборку

In [6]:

```
for i,line in enumerate(train_data):
    data=re.split(r' ',line)
    x_train[i][0]=float(data[1])
    x_train[i][1]=float(data[2])
    if data[3]=='red':
        y_train[i]=0
    else:
        y_train[i]=1

for i,line in enumerate(test_data):
    data=re.split(r' ',line)
    x_test[i][0]=float(data[1])
    x_test[i][1]=float(data[2])
    if data[3]=='red':
        y_test[i]=0
    else:
        y_test[i]=1
```

Нормализуем данные

Нормализуем обучающую выборку

In [7]:

```
means,stds=x_train.mean(axis=0),x_train.std(axis=0)

for i in range(x_train.shape[0]):
    for j in range(x_train.shape[1]):
        x_train[i][j]=(x_train[i][j]-means[j])/stds[j]

print('Первая строка после нормализации',x_train[0])
print('Среднее арифметическое по столбцам ',means)
print('Дисперсия по столбцам' ,stds)
```

Первая строка после нормализации [-0.2401377 0.11391791]
Среднее арифметическое по столбцам [-0.08505942 -0.00590062]
Дисперсия по столбцам [1.04276681 0.94924718]

Нормализуем тестовую выборку

In [8]:

```
for i in range(x_test.shape[0]):
    for j in range(x_test.shape[1]):
        x_test[i][j]=(x_test[i][j]-means[j])/stds[j]
```

Создадим модель нейронной сети

```
In [9]: model=Sequential()

#добавим нормального шума ,чтобы избежать переобучения
model.add(GaussianNoise(stddev=0.02,input_shape=(2,)))

model.add(Dense(256))
model.add(LeakyReLU(0.2))

model.add(Dense(128))
model.add(LeakyReLU(0.2))

model.add(Dense(1))
model.add(Activation('sigmoid'))
```

WARNING:tensorflow:From C:\Anaconda\envs\env_neuro\lib\site-packages\tensorflow_core\python\ops\resource_variable_ops.py:1630: calling BaseResourceVariable.__init__ (from tensorflow.python.ops.resource_variable_ops) with constraint is deprecated and will be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.

```
In [10]: model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
=====		
gaussian_noise_1 (GaussianNo	(None, 2)	0
dense_1 (Dense)	(None, 256)	768
leaky_re_lu_1 (LeakyReLU)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32896
leaky_re_lu_2 (LeakyReLU)	(None, 128)	0
dense_3 (Dense)	(None, 1)	129
activation_1 (Activation)	(None, 1)	0
=====		
Total params: 33,793		
Trainable params: 33,793		
Non-trainable params: 0		

```
In [11]: batch=2
epochs=80

validation_split=0.2

model.compile(loss='binary_crossentropy',optimizer='RMSprop',metrics=['accuracy'])

history=model.fit(x_train,y_train,batch_size=batch,epochs=epochs,verbose=True,validation_split=validation_split)

80/80 [=====] - 0s 2ms/step - loss: 0.0653 - accuracy: 0.9750 - val_loss: 0.2084 - val_accuracy: 0.9
500
Epoch 75/80
80/80 [=====] - 0s 2ms/step - loss: 0.0219 - accuracy: 0.9875 - val_loss: 0.2746 - val_accuracy: 0.9
000
Epoch 76/80
80/80 [=====] - 0s 2ms/step - loss: 0.0326 - accuracy: 0.9875 - val_loss: 0.2389 - val_accuracy: 0.9
500
Epoch 77/80
80/80 [=====] - 0s 2ms/step - loss: 0.0465 - accuracy: 0.9875 - val_loss: 0.3187 - val_accuracy: 0.9
000
Epoch 78/80
80/80 [=====] - 0s 2ms/step - loss: 0.0510 - accuracy: 0.9750 - val_loss: 0.3315 - val_accuracy: 0.9
000
Epoch 79/80
80/80 [=====] - 0s 2ms/step - loss: 0.0141 - accuracy: 1.0000 - val_loss: 0.4322 - val_accuracy: 0.9
000
Epoch 80/80
80/80 [=====] - 0s 2ms/step - loss: 0.0510 - accuracy: 0.9875 - val_loss: 0.3610 - val_accuracy: 0.9
000
```

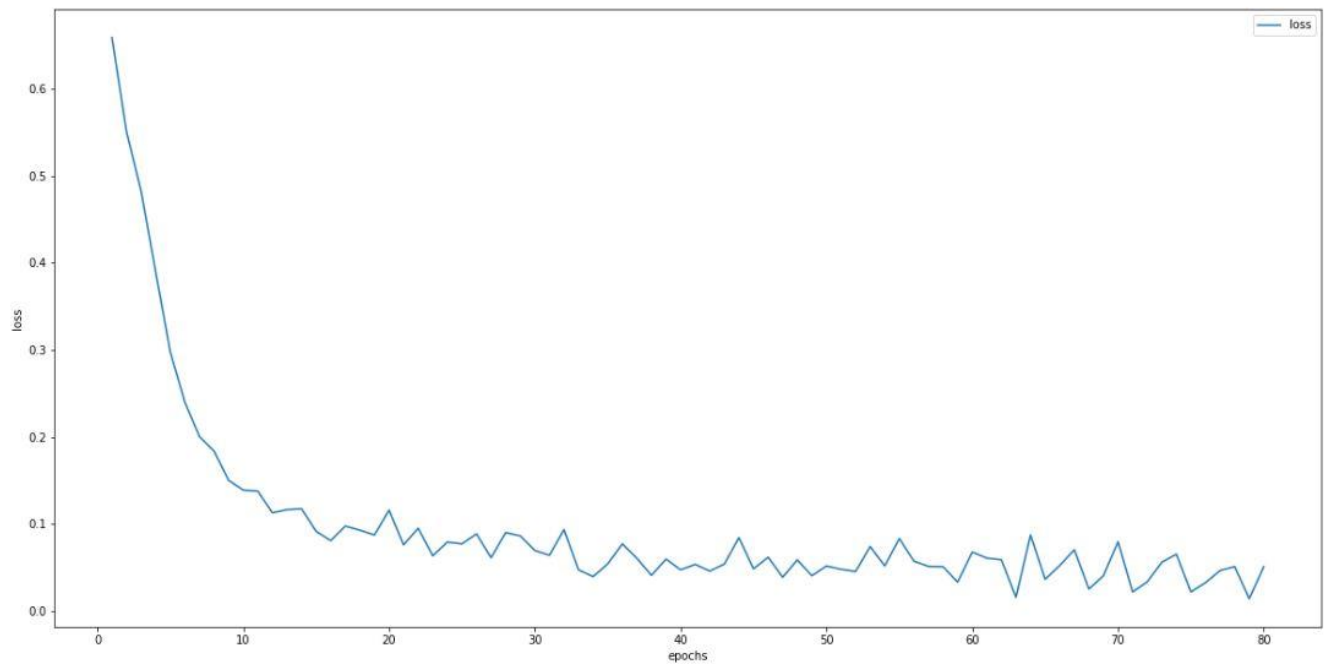
Выведем график метрики качества и метрики ошибки от эпохи

```
In [27]: x=np.arange(1,epochs+1)

fig, ax = plt.subplots()
ax.plot(x,history.history['loss'],label='loss')
plt.xlabel('epochs')
plt.ylabel('loss')

plt.legend()

fig.set_figheight(10)
fig.set_figwidth(20)
plt.show()
```



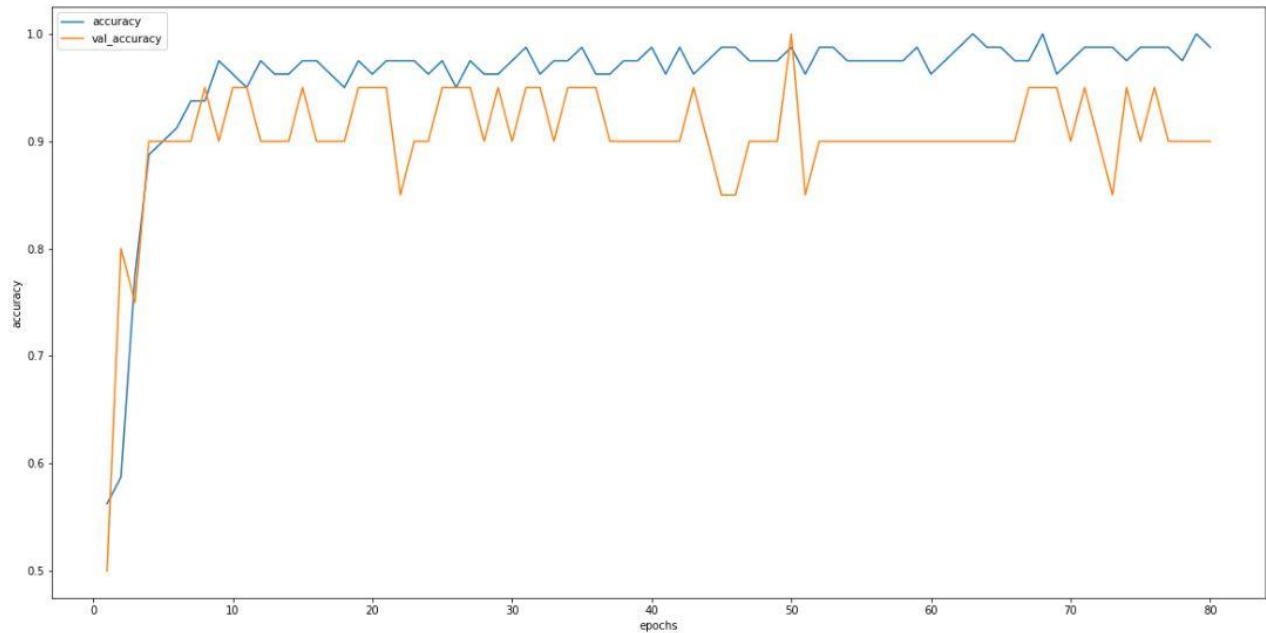
```
In [28]: x=np.arange(1,epochs+1)

fig, ax = plt.subplots()

ax.plot(x,history.history['accuracy'],label='accuracy')
ax.plot(x,history.history['val_accuracy'],label='val_accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')

plt.legend()

fig.set_figheight(10)
fig.set_figwidth(20)
plt.show()
```



Проверим модель на тестовой выборке

```
In [18]: score=model.evaluate(x_test,y_test)
print("Test loss:", score[0])
print('Test accuracy:', score[1])
```

```
100/100 [=====] - 0s 85us/step
Test loss: 0.04029238638089737
Test accuracy: 0.9800000190734863
```

Создадим модели байесовского классификатора бернулли и гаусса

```
In [19]: NB = GaussianNB()
NB.fit(x_train, y_train)
BNB=BernoulliNB()
BNB.fit(x_train, y_train)
```

```
C:\Anaconda\envs\env_neuro\lib\site-packages\sklearn\utils\validation.py:73: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
    return f(**kwargs)
C:\Anaconda\envs\env_neuro\lib\site-packages\sklearn\utils\validation.py:73: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
    return f(**kwargs)
```

```
Out[19]: BernoulliNB()
```

Проверим модель на тестовой выборке

```
In [20]: y_predict = NB.predict(x_test)
y2_predict=BNB.predict(x_test)
print("Accuracy Normal NB: {:.2f}".format(NB.score(x_test, y_test)))
print("Accuracy Bernulli NB: {:.2f}".format(BNB.score(x_test, y_test)))
```

```
Accuracy Normal NB: 0.93
Accuracy Bernulli NB: 0.60
```

Создадим случайное дерево решений

```
In [21]: rf = RandomForestClassifier()

rf.fit(x_train, y_train)
score=rf.score(x_test,y_test)
print('Random forest accuracy',score)
```

```
Random forest accuracy 0.95
```

```
C:\Users\Администратор\AppData\Roaming\Python\Python36\site-packages\ipykernel_launcher.py:3: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
This is separate from the ipykernel package so we can avoid doing imports until
```