МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ЛАБОРАТОРНАЯ РАБОТА №6

по дисциплине «Искусственные нейронные сети»

Тема: «Прогноз успеха фильмов по обзорам»

Студент гр. 7383 _____ Лосев М.Л.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

Цель.

Прогноз успеха фильмов по обзорам (Predict Sentiment From Movie Reviews).

Задачи.

- Ознакомиться с задачей регрессии
- Изучить способы представления текста для передачи в ИНС
- Достигнуть точность прогноза не менее 95%

Выполнение работы.

Предложенная архитектура позволяет достичь точности 0.975+:

```
model = models.Sequential()
# Input - Layer
model.add(layers.Dense(50, activation = "relu",
input_shape=(dimension, )))
# Hidden - Layers
model.add(layers.Dropout(0.3, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation = "relu"))
model.add(layers.Dropout(0.2, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation = "relu"))
# Output- Layer
model.add(layers.Dense(1, activation = "sigmoid"))
model.compile(optimizer = "adam", loss =
"binary_crossentropy", metrics = ["accuracy"])
```

Чтобы сследовать результаты при различном размере вектора представления текста, были рассмотрены следующие значения размера этого вектора: *10*, *50*, *100*, *250*, *500*, *1000*, *2500*, *5000*, *10000*. Была построена диаграма, показывающая зависимость достигаемой точности от длины вектора. Она представлена на рисунке 1.

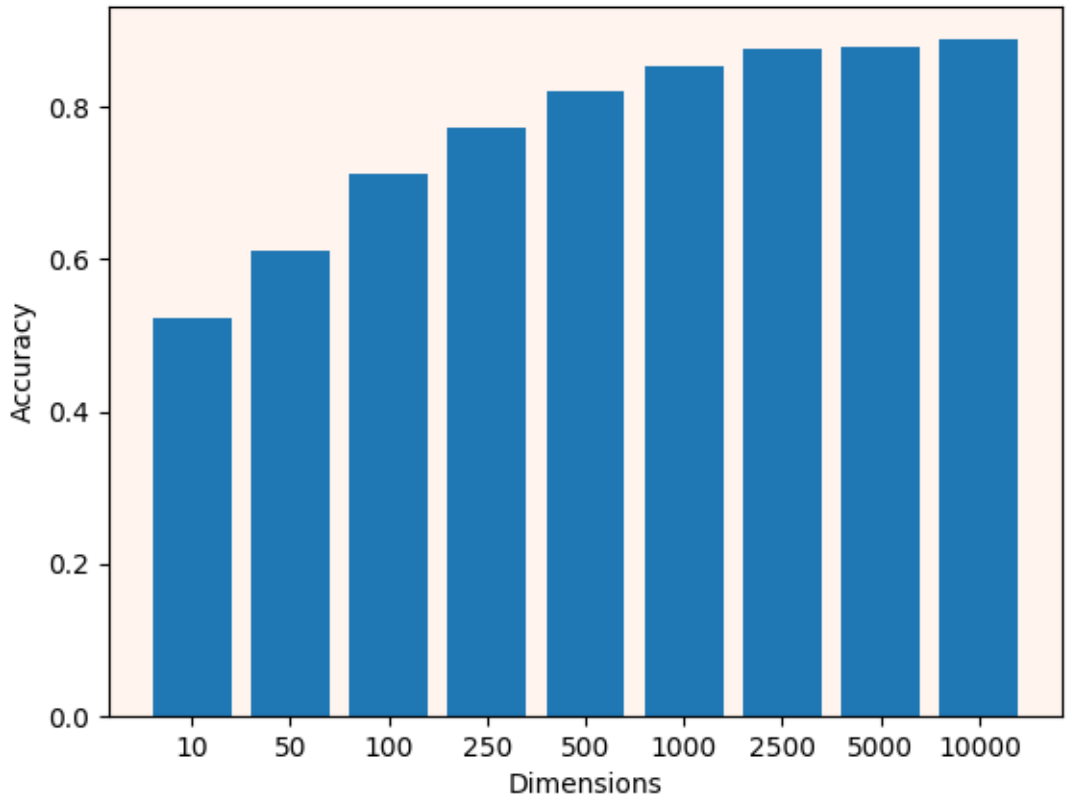

Рисунок 1 - диаграма точности при разных длинах вектора

По диаграмме на рис. 1 видно, что чем больше длина вектора представления текста, тем выше точность модели, причем для длины вектора 1000 и 10000 точность почти одинаково высока, а для длин вектора меньше 1000 точность быстро растет с увеличением его длины.

Была написана функция, которая позволяет ввести пользовательский текст (она преобразует его в вектор представления текста) и сделать для него предсказание с помошью модели. Код представлен в приложении А.

Возьмем рецензию, предложенную в методических указаниях к работе и сделаем для нее предскаание. Результат представлен на рисунке 2.

```
42500/45000 [===========================>..] - ETA: 0s - loss: 0.0114 - accuracy: 0.996
43500/45000 [============================>.] - ETA: 0s - loss: 0.0114 - accuracy: 0.996
44500/45000 [============================>.] - ETA: 0s - loss: 0.0113 - accuracy: 0.996
45000/45000 [==============================] - 4s 86us/step - loss: 0.0113 - accuracy:
It is 0.9992779687745497 percent good
```

Рисунок 2 - предсказание модели для хорошего отзыва

0.999 это ожидаемый результат, ведь сугубо положительный.

Вывод.

В ходе выполнения лабораторной работы была создана сеть для оценки успеха фильма по отзыву. Было исследовано влияние размера вектора представления текста на результат работы сети. Адекватная точность достигается при размере вектора представления текста 1000 и больше. Чем выше размер вектора, тем выше точность сети. Юыла написана функция, позволяющая получить оценку фильма по введенному обзору.

Приложение А

```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
from keras.utils import to_categorical
from keras import models
from keras import layers
from keras.datasets import imdb
import os
import string


picdir = './pics/'

def vectorize(sequences, dimension = 10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1
    return results

def getModel(dimension = 10000):
    model = models.Sequential()
    # Input - Layer
    model.add(layers.Dense(50, activation = "relu", input_shape=(dimension, )))
    # Hidden - Layers
    model.add(layers.Dropout(0.3, noise_shape=None, seed=None))
    model.add(layers.Dense(50, activation = "relu"))
    model.add(layers.Dropout(0.2, noise_shape=None, seed=None))
    model.add(layers.Dense(50, activation = "relu"))
    # Output- Layer
    model.add(layers.Dense(1, activation = "sigmoid"))
    model.compile(optimizer = "adam", loss = "binary_crossentropy", metrics =
["accuracy"])
    return model

def loadData(num_words=10000):
    (training_data, training_targets), (testing_data, testing_targets) =
imdb.load_data(num_words=num_words)
    data = np.concatenate((training_data, testing_data), axis=0)
    targets = np.concatenate((training_targets, testing_targets), axis=0)
    return data, targets

def savePlot(label, history):
    plt.clf()
    arg = range(1, len(history['accuracy']) + 1)
    plt.plot(arg, history['accuracy'], 'r', label='Training acc')
    plt.plot(arg, history['val_accuracy'], 'b', label='Validation acc')
    plt.title(label)
    plt.xlabel('Epoch')
    plt.ylabel(label)
    plt.legend()
    plt.savefig(label + '.png')
    plt.savefig(picdir + label + '.png')

def exploreDimensionValue(dim):
    model = getModel(dim)
    data, targets = loadData(dim)
    data = vectorize(data, dim)
    targets = np.array(targets).astype("float32")
    results = model.fit(data, targets, epochs= 50, batch_size = 500,
validation_split=0.1)
    savePlot(str(dim) + " dimensions", results.history)
    return results.history['val_accuracy'][-1]
```

```
def encode(text):
    text = text.translate(str.maketrans('', '', string.punctuation)) # remove
punctuation
    text = text.lower() # to lowercase cuz there's no indexes for non-lowercase
words
    words = text.split(" ")
    index = imdb.get_word_index()
    encoded = [index.get(word, 0) for word in words]
    return encoded

def makePrediction(review, model, dim = 10000):
    vectorized = vectorize([encode("Hello, little dear!")], dim)
    return model.predict(vectorized)[0][0]


# Collect data
dimAcc = dict()
for dim in [10, 50, 100, 250, 500, 1000, 2500, 5000, 10000]:
    acc = exploreDimensionValue(dim)
    dimAcc[dim] = acc
# Draw a diagram
fig, ax = plt.subplots()
plt.ylabel('Accuracy')
plt.xlabel('Dimensions')
ax.bar([str(key) for key in dimAcc.keys()], dimAcc.values())
ax.set_facecolor('seashell')
fig.set_facecolor('floralwhite')
plt.savefig(picdir + 'diagram.png')


# Make a prediction for user review
review = """
        # this film was just brilliant casting location scenery story direction
everyone's really suited the part they played
        and you could just imagine being there robert # is an amazing actor and
now the same being director # father came from
        the same scottish island as myself so i loved the fact there was a real
connection with this film the witty remarks
        throughout the film were great it was just brilliant so much that i
bought the film as soon as it was released for #
        and would recommend it to everyone to watch and the fly fishing was
amazing really cried at the end it was so sad and
        you know what they say if you cry at a film it must have been good and
this definitely was also # to the two little
        boy's that played the # of norman and paul they were just brilliant
children are often left out of the # list i think
        because the stars that play them all grown up are such a big profile for
the whole film but these children are amazing
        and should be praised for what they have done don't you think the whole
story was so lovely because it was true and was
        someone's life after all that was shared with us all
    """
# Get a model
dim = 10000
model = getModel(dim)
data, targets = loadData(dim)
data = vectorize(data, dim)
targets = np.array(targets).astype("float32")
model.fit(data, targets, epochs= 50, batch_size = 500, validation_split=0.1)
# '1 - prediction' cuz good reviews label is 1, bad ones label is 0
print('It is ' + str(1 - makePrediction(review, model, dim)) + ' percent good')

# Wake up Neo
os.system('play --no-show-progress --null --channels 1 synth 1 sine 440')
```