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

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Искусственные нейронные сети»

Тема: «Прогноз успеха фильмов по обзорам»

Студент гр. 7383 _____ Рудоман В. А.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

Цель работы:

Реализовать прогнозирование успеха фильмов по обзорам (Predict Sentiment From Movie Reviews)

Задачи.

1. Ознакомиться с задачей регрессии
2. Изучить способы преставления текста для передачи в ИНС
3. Достигнуть точности прогноза не менее 95%

Ход работы.

1. Была создана и обучена модель нейронной сети
2. Для исследования результатов при различном размере вектора представления текста была реализована функция `dimensions_test()`. Для тестов были выбраны размеры вектора равные 10, 50, 100, 500, 1000, 5000, 10000. Графики потерь и точности приведены ниже на рис. 1, 2.

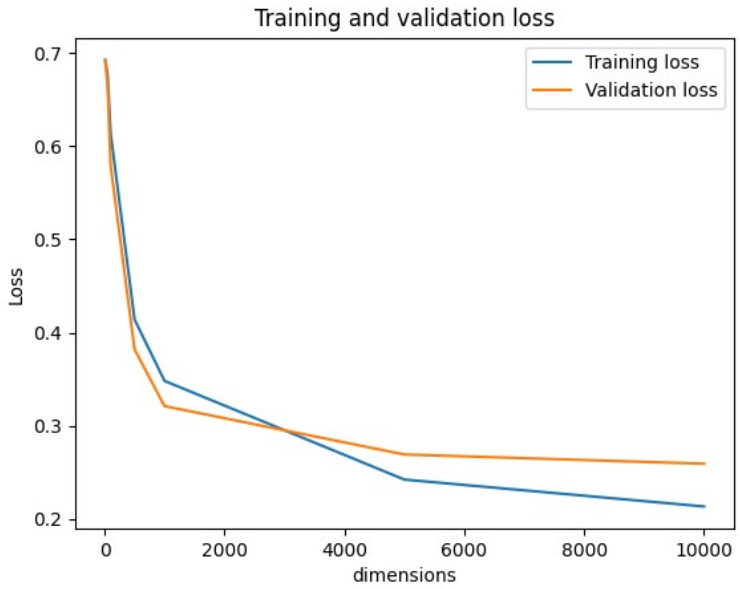

Рисунок 1 — График потерь при различной длине вектора

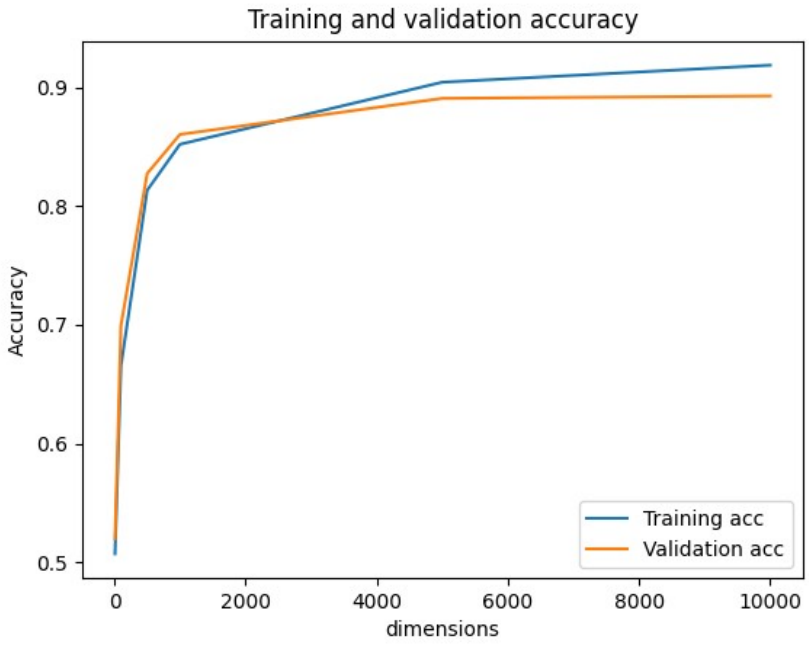

Рисунок 2 — График точности при различной длине вектора

Исходя из графиков видно, что наибольшая точность достигается при длине вектора 10000. Аналогичная ситуация и с графиком потерь.

3. Для оценки филма по пользовательскому отзыву была реализована функция `own_text(file)`.

Содержание отзыва 1 (Кошки):

«The cats were very creepy looking and it didn't say the name of most of the cats. Idris Elba had a six pack which is not very cat like and all the characters are very forgettable. Wouldn't recommend».

ИНС выдала оценку в 0.29227227. Указывает на то, что фильм скорее всего не стоит смотреть.

Содержание отзыва 2 (Достать Ножи):

«I loved it. An amazing cast of characters led flawlessly by Daniel Craig, Chris Evans and Ana de Armas. The moment I thought I knew what was happening, a twist threw me. One of my year's favorite films».

ИНС выдала оценку в 0.47866812. Скорее всего фильм можно порекомендовать к просмотру.

Вывод:

Была построена сеть, прогнозирующая рейтинги фильмов по обзорам. Рассмотрено преобразование текста в формат, пригодный для работы с нейросетью. Было протестировано влияние размера вектора текста на точность результата. В ходе чего была выявлена длина вектора, которая позволяет добиться наибольшей точности — 10000. Также была реализована функция прогнозирования рейтинга фильма по пользовательскому отзыву.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

```
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


TEST_DIMENSIONS = [10, 50, 100, 500, 1000, 5000, 10000]

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

def data_load(dimension = 10000):
    (training_data, training_targets), (testing_data,
testing_targets) = imdb.load_data(num_words=dimension)
    data = np.concatenate((training_data, testing_data), axis=0)
    targets = np.concatenate((training_targets, testing_targets),
axis=0)
    data = vectorize(data, dimension)
    targets = np.array(targets).astype("float32")

    test_x = data[:10000]
    test_y = targets[:10000]
    train_x = data[10000:]
    train_y = targets[10000:]

    return test_x, test_y, train_x, train_y

def model_build(dimensions):
    model = models.Sequential()
    model.add(layers.Dense(50, activation = "relu",
input_shape=(dimensions, )))
    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"))
    model.add(layers.Dense(1, activation = "sigmoid"))
    model.compile(optimizer = "adam", loss =
"binary_crossentropy", metrics = ["accuracy"])
    return model
```

```
def model_run(dimension):
    test_x, test_y, train_x, train_y = data_load(dimension)
    model = model_build(dimension)
    results = model.fit(train_x, train_y,
                        epochs= 2,
                        batch_size = 500,
                        validation_data = (test_x, test_y)
                        )
    return (results.history['accuracy'][-1],
        results.history['val_accuracy'][-1],
        results.history['loss'][-1],
        results.history['val_loss'][-1])

def plot(res_train_acc, res_test_acc, res_traing_loss,
res_test_loss, DIM):
    plt.plot(TEST_DIMENSIONS, res_traing_loss, label='Training
loss')
    plt.plot(TEST_DIMENSIONS, res_test_loss, label='Validation
loss')
    plt.title('Training and validation loss')
    plt.xlabel('dimensions')
    plt.ylabel('Loss')
    plt.legend()
    plt.savefig("loss.png")
    plt.clf()

    plt.plot(TEST_DIMENSIONS, res_train_acc, label='Training acc')
    plt.plot(TEST_DIMENSIONS, res_test_acc, label='Validation
acc')
    plt.title('Training and validation accuracy')
    plt.xlabel('dimensions')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.savefig("accuracy.png")


def dimensions_test():
    res_train_acc = []
    res_test_acc = []
    res_train_loss = []
    res_test_loss = []
    for DIM in TEST_DIMENSIONS:
        train_acc, test_acc, train_loss, test_loss =
model_run(DIM)
        res_train_acc.append(train_acc)
        res_test_acc.append(test_acc)
        res_train_loss.append(train_loss)
        res_test_loss.append(test_loss)
```

```
    plot(res_train_acc, res_test_acc, res_train_loss,
res_test_loss, DIM)



def own_text(file):
    test_x, test_y, train_x, train_y = data_load(10000)
    model = model_build(10000)
    model.fit(train_x, train_y, epochs=2, batch_size=500,
                        validation_data=(test_x, test_y))
    txt = []
    with open(file, 'r') as text:
        f_text = text.read()
        index = imdb.get_word_index()
        for word in f_text:
            if word in index and index[word] < 10000:
                txt.append(index[word])

    txt = vectorize([txt])
    result = model.predict(txt)
    print(result)

own_text("text.txt")

#dimensions_test()
```