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 _____ Левкович Д. В.

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 представления текста для передачи в ИНС
3. Достигнуть точность прогноза не менее 95%

Ход работы.

1. Была создана и обучена модель нейронной сети в (код представлен в приложении А).

Модель представлена на рис. 1.

```
model = models.Sequential()
model.add(layers.Dense(50, activation = "relu"))
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"])
```

Рисунок 1 - Модель сети

2. Для тестирования поведения сети в зависимости от размера вектора представления текста была написана функция `test_dimensions`.

Протестировано поведение при варьирующемся размере вектора представления текста. График точности показан на рис. 2, 3.

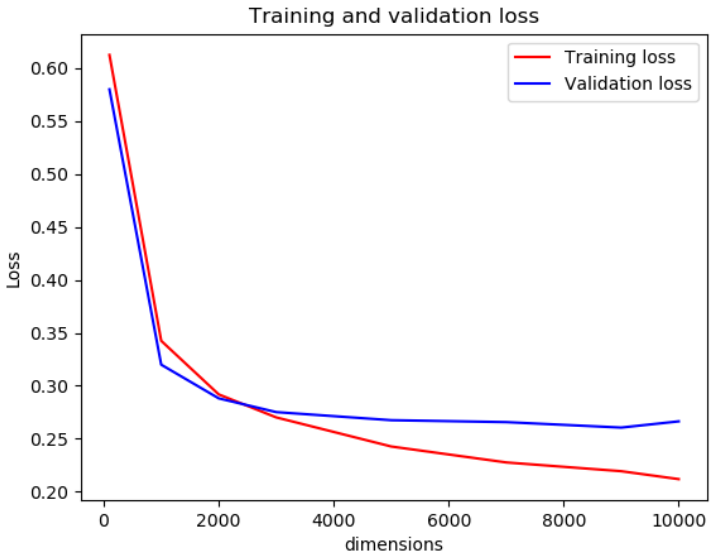

Рисунок 2 - График потерь сети

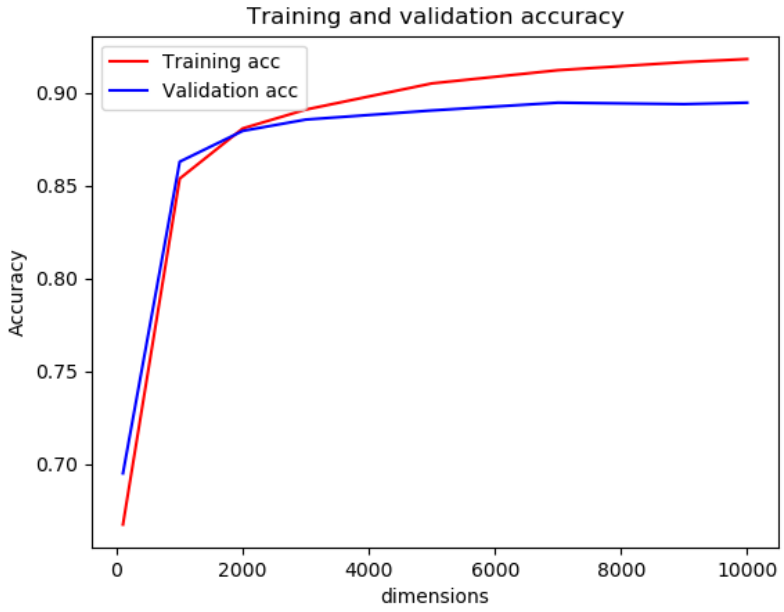

Рисунок 3 - График точности сети

Из графика видно, что наибольшая точность достигается при размере вектора 10000. И можно увидеть, что с увеличением размера вектора увеличивается точность.

3. Была написана функция `load_user_text` для загрузки пользовательского текста и прогнозирования успеха фильма по этому тексту.

Тексты обзоров: «worst movie i have ever seen bad scenario» и оценка сети

0.34097266, то есть это скорее плохая оценка. «great movie perfect scenario nice actors» и оценка сети 0.58716035 – скорее хороший фильм.

Выводы:

Была построена сеть, прогнозирующая оценку фильма по обзорам. Было рассмотрено преобразование текста в формат, с которым может работать нейросеть. Было исследовано влияние размера вектора представления текста и выявлено, что наибольшей точностью обладает сеть с равным 10000. Была написана функция прогнозирования оценки по пользовательскому тексту.

ПРИЛОЖЕНИЕ А

```
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

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


def load_dataset(dimension):
    (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 (train_x, train_y, test_x, test_y)


def build_model():
    model = models.Sequential()
    model.add(layers.Dense(50, activation = "relu"))
    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 test_model(dimension):
    train_x, train_y, test_x, test_y = load_dataset(dimension)
    model = build_model()
    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, dimensions):
    plt.plot(dimensions, res_traing_loss, 'r', label='Training loss')
    plt.plot(dimensions, res_test_loss, 'b', label='Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel('dimensions')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
    plt.clf()
    plt.plot(dimensions, res_train_acc, 'r', label='Training acc')
    plt.plot(dimensions, res_test_acc, 'b', label='Validation acc')
    plt.title('Training and validation accuracy')
    plt.xlabel('dimensions')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()


def test_dimensions(dimensions):
    res_train_acc = []
    res_test_acc = []
    res_train_loss = []
    res_test_loss = []
```

```
    for dimension in dimensions:
        train_acc, test_acc, train_loss, test_loss = test_model(dimension)
        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, dimensions)


train_x, train_y, test_x, test_y = load_dataset(10000)
model = build_model()
model.fit(train_x, train_y, epochs=2, batch_size=500,
                    validation_data=(test_x, test_y))


#text separated only with spaces
def load_user_text(review):
    text = review.split()
    print(text)
    dict = imdb.get_word_index()
    words_num = []
    for word in text:
        word = dict.get(word)
        if word is not None and word < 10000:
            words_num.append(word)
    text = vectorize([words_num])
    res = model.predict(text)
    print(res)


string = input()
load_user_text(string)
```