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

ОТЧЕТ

по лабораторной работе 6

по дисциплине «Искусственные нейронные сети»

Тема: «Прогноз успеха фильмов по обзорам»

Студентка гр. 7383 Прокопенко Н.

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 точности прогноза не менее 95%

Требования.

1. Построить и обучить нейронную сеть для обработки;
2. Исследовать результаты при различном размере вектора представления текста;
3. Написать функцию, которая позволяет ввести пользовательский текст (в отчете привести пример работы сети на пользовательском тексте).

Ход работы.

Была построена нейронная сеть, код которой представлен в приложении А.

Исследуем влияние размера вектора входных данных. При размере 10000 образов точность составила 89%, при 1000 – 86%, при 100 – 69%, а при 10 – 50%. Графики точности и ошибок представлены на рис. 1 – 2, 3 – 4, 5 – 6, 7 – 8 соответственно.

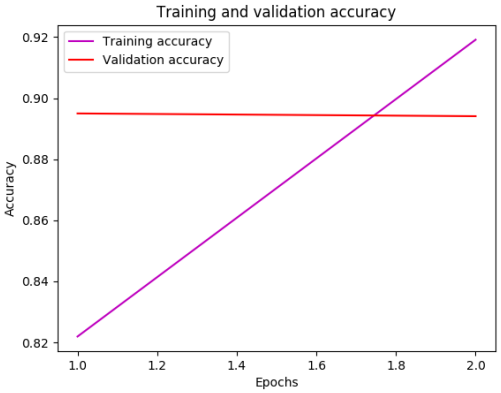

Рисунок 1 – График точности

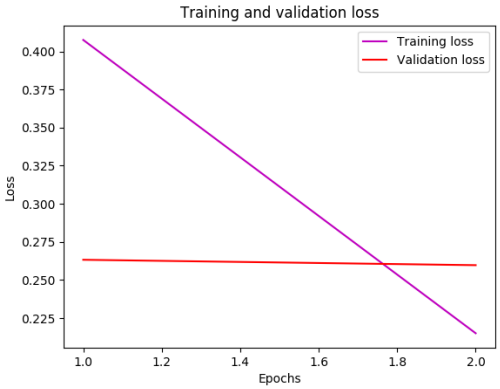

Рисунок 2 – График ошибки

Рисунок 3 – График точности

Рисунок 4 – График ошибки

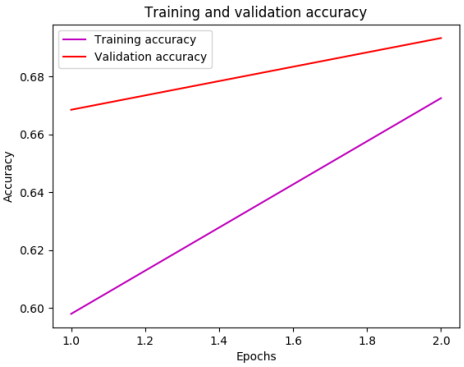

Рисунок 5 – График точности

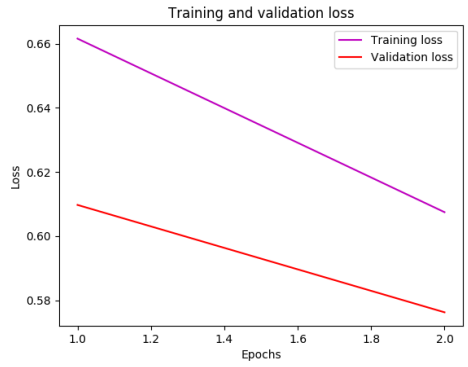

Рисунок 6 – График ошибки

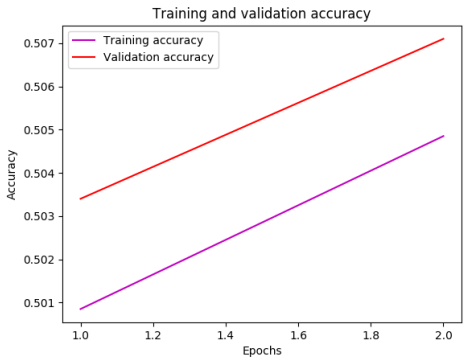

Рисунок 7 – График точности

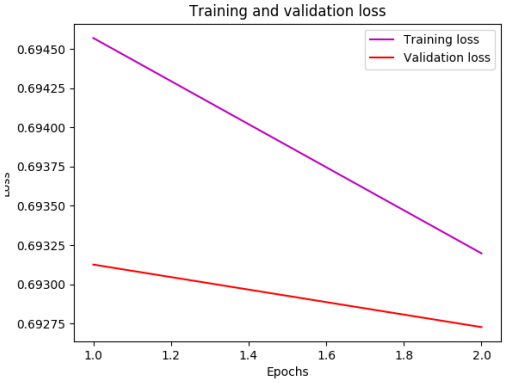

Рисунок 8 – График ошибки

Наибольшая точность достигается при длине вектора 10000 и равна 89%. По результатам видно, что с уменьшением размера вектора представления текста уменьшается точность и увеличивается ошибка. Это может быть связано с уменьшением в обзоре количества слов, характеризующих положительное или отрицательное отношение. Была протестирована работа программы на примере отзыва: «The cartoon characters were amazing! Movie graphics are always on top. Disney still holds a high level.». Результат – 0,60, что соответствует положительному настроению отзыва.

Выводы.

В ходе выполнения лабораторной работы была изучена задача регрессии. Была построена и обучена нейронная сеть. Была написана функция, позволяющая пользователю вводить свой текст. Была исследована зависимость результата от размера входного вектора.

ПРИЛОЖЕНИЕ А

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

dmns = 10000

review = ["The cartoon characters were amazing! Movie graphics are
always on top. Disney still holds a high level."]

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

def user_func(review):
    index = imdb.get_word_index()
    test_x = []
    words = []
    for line in review:
        lines     =     line.translate(str.maketrans('',     '',
',.?!:;()')).lower()
        for chars in lines:
            chars = index.get(chars)
            if chars is not None and chars < 10000:
                words.append(chars)
        test_x.append(words)
    test_x = vectorize(test_x)
    model = no_user_func()
    prediction = model.predict(test_x)
    print(prediction)

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

    data = vectorize(data)
```

```
        targets = np.array(targets).astype("float32")

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

        model = Sequential()
        # Input - Layer
        model.add(layers.Dense(50,      activation      =      "relu",
input_shape=(dmns, )))
        # 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"])

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

        loss = history.history['loss']
        val_loss = history.history['val_loss']
        epochs = range(1, len(loss) + 1)
        plt.plot(epochs, loss, 'm', label='Training loss')
        plt.plot(epochs, val_loss, 'r', label='Validation loss')
        plt.title('Training and validation loss')
        plt.xlabel('Epochs')
        plt.ylabel('Loss')
        plt.legend()
        plt.show()
        plt.clf()

        acc = history.history['accuracy']
        val_acc = history.history['val_accuracy']
        plt.plot(epochs, acc, 'm', label='Training accuracy')
        plt.plot(epochs, val_acc, 'r', label='Validation accuracy')
        plt.title('Training and validation accuracy')
        plt.xlabel('Epochs')
        plt.ylabel('Accuracy')
        plt.legend()
```

```
    plt.show()

    results = model.evaluate(test_x, test_y)
    print(results)
    return model

user_func(review)
```