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

Студентка гр. 7383 _____ Тян Е.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург
2020

Цель работы.

Реализовать предсказание прогноза успеха фильма по обзорам.

Задание.

1. Ознакомиться с задачей регрессии.
2. Изучить способы представления текста для передачи в ИНС.

Ход работы.

Импортируем необходимые для работы зависимости для предварительной обработки данных и построения модели:

```
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 import Sequential
```

Загрузим дотаяет IMDb, который уже встроен в Keras. Поскольку мы не хотим иметь данные обучения и тестирования в пропорции 50/50, мы сразу же объединим эти данные после загрузки для последующего разделения в пропорции 80/20:

```
from keras.datasets import imdb
(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)
```

Подготовим данные. Нужно векторизовать каждый обзор и заполнить его нулями, чтобы вектор содержал ровно 10 000 чисел. Это означает, что каждый обзор, который короче 10 000 слов, мы заполняем нулями. Это делается потому, что самый большой обзор имеет почти такой же размер, а каждый элемент входных данных нашей нейронной сети должен иметь одинаковый размер:

```
def vectorize(sequences, dimension = dmns):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1
    return results
data = vectorize(data)
targets = np.array(targets).astype("float32")
```

Разделим датасет на обучающий и тестировочный наборы. Обучающий набор будет состоять из 40 000 обзоров, тестировочный — из 10 000:

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

Создадим простую нейронную сеть. Начнем с определения типа модели, которую хотим создать. В Keras доступны два типа моделей: последовательные и с функциональным API. Затем нужно добавить входные, скрытые и выходные слои. Для предотвращения переобучения будем использовать между ними исключение («dropout»). На каждом слое используется функция «dense» для полного соединения слоев друг с другом. В скрытых слоях используем функцию активации «relu», потому это практически всегда приводит к удовлетворительным результатам. На выходном слое используем сигмоидную функцию, которая выполняет перенормировку значений в диапазоне от 0 до 1. Устанавливаем размер входных элементов датасета равным 10 000, потому что наши обзоры имеют размер до 10 000 целых чисел. Входной слой принимает элементы с размером 10 000, а выдает — с размером 50:

```
model = Sequential()
model.add(layers.Dense(50, activation = "relu", input_shape=(dmns, )))
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"))
```

Скомпилируем нашу модель, то есть, по существу, настроим ее для обучения. Будем использовать оптимизатор «adam». В качестве функции потерь используем бинарную кросс-энтропию, в качестве метрики оценки — точность:

```
model.compile( optimizer = "adam", loss = "binary_crossentropy", metrics = ["accuracy"])
```

Обучим модель. Будем делать это с размером партии 500 и только двумя эпохами, поскольку модель начинает переобучаться, если тренировать ее дольше:

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

Была построена нейронная сеть, графики точности и ошибки которой показаны на рис. 1 и 2. Точность составляет: 0.89. Ошибка – 0.26.

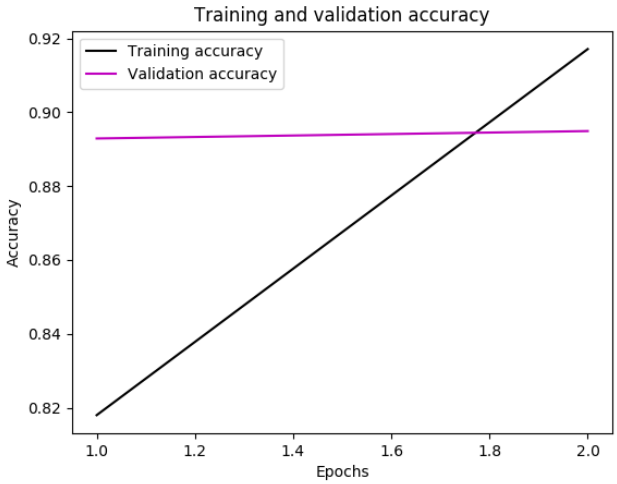

Рисунок 1 – График точности для вектора представления размером 10 тыс.

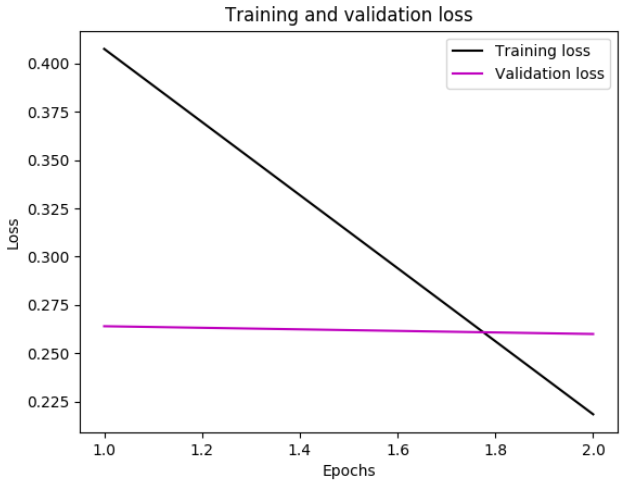

Рисунок 2 – График ошибки для вектора представления размером 10 тыс.

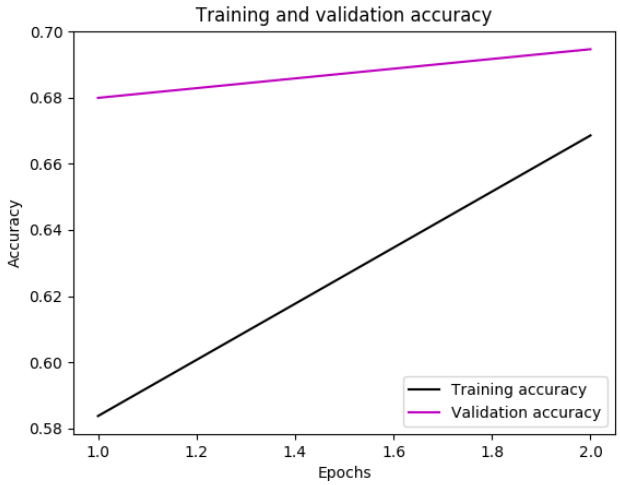

Рисунок 3 – График точности для вектора представления размером 100

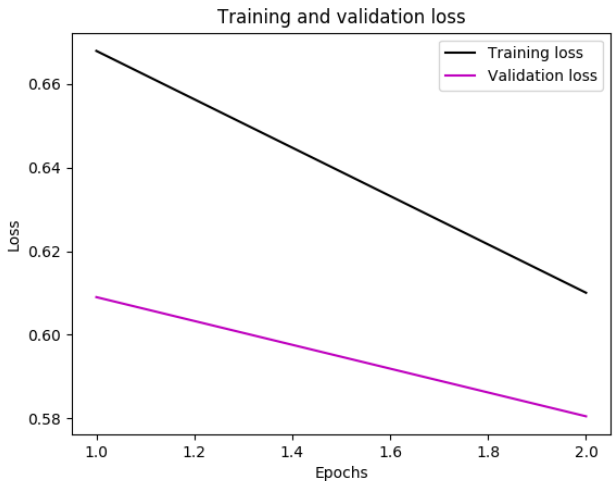

Рисунок 4 – График ошибки для вектора представления размером 100

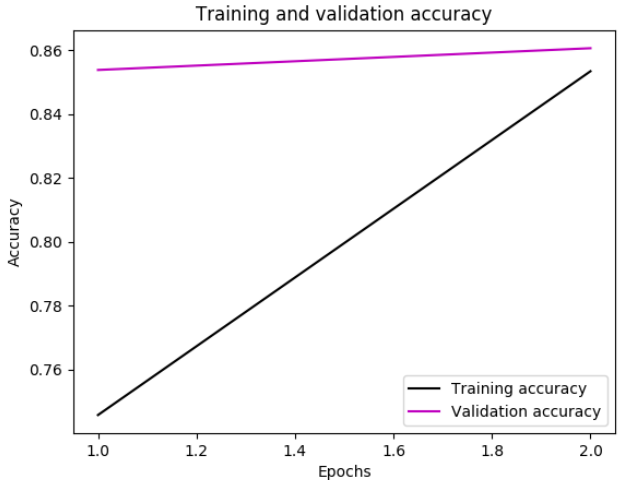

Рисунок 5 – График точности для вектора представления размером 1000

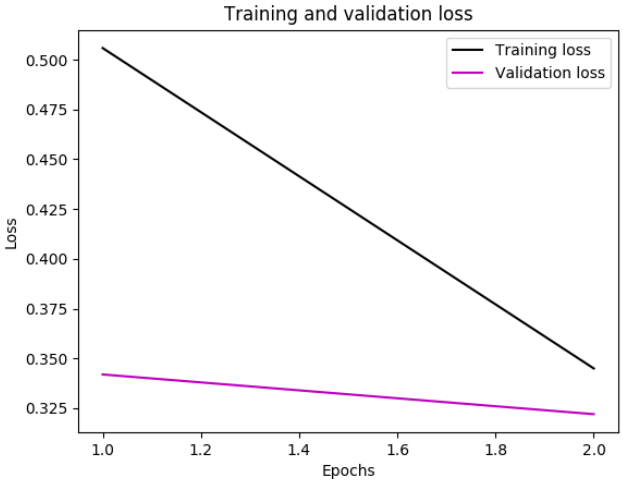

Рисунок 6 – График ошибки для вектора представления размером 1000

Изучим результаты при различном размере вектора представления текста. Для вектора представления размером 100 была получена точность 0.69, ошибка – 0.58, результаты представлены на рис. 3 и 4. Для вектора размером 1000 была получена точность 0.87, ошибка – 0.32, результаты показаны на рис. 5 и 6.

Напишем функцию, которая позволяет ввести пользовательский текст:

```
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)
```

Результат работы программы на примере: "My God People, Really?? Get Your heads out of the sand. Movie is over rated. Its an average movie. Don't know why it created this much buzz." – 0.29, что на самом деле подтверждает негативное настроение отзыва.

Выводы.

В ходе выполнения лабораторной работы была построена и обучена нейронная сеть для обработки текста. Точность – 0.89, ошибка – 0.26 для вектора представления текста размером 10 тыс. Были исследованы результаты для различных размеров вектора представления текста. С уменьшением размера вектора, ухудшается точность, что подтверждает тот факт, что вероятность точного определения настроения уменьшается из-за уменьшения количества найденных слов в тексте. Написана функция, позволяющая ввести пользовательский текст.