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 _____ Ласковенко Е.А.

Преподаватель _____ Жукова Н. А.

Санкт-Петербург

2020

Цель работы.

Реализовать ИНС для прогноза 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%

Требования.

1. Построить и обучить нейронную сеть для обработки текста.
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.

1. Была создана модель искуственной нейронной сети для прогноза успеха фильмов оп обзорам. Код программы представлен в приложении А.

2. Для исследования влияния размера вектора представления текста на точность результата сети выберем следующие размерности: 1000, 3000, 5000, 7000, 9000, 11000, 13000, 15000. Обучим модель для заданных размеров вектора представления текста и оценим точность нейронной сети. Графики точностей для модели представлены на рис. 1.

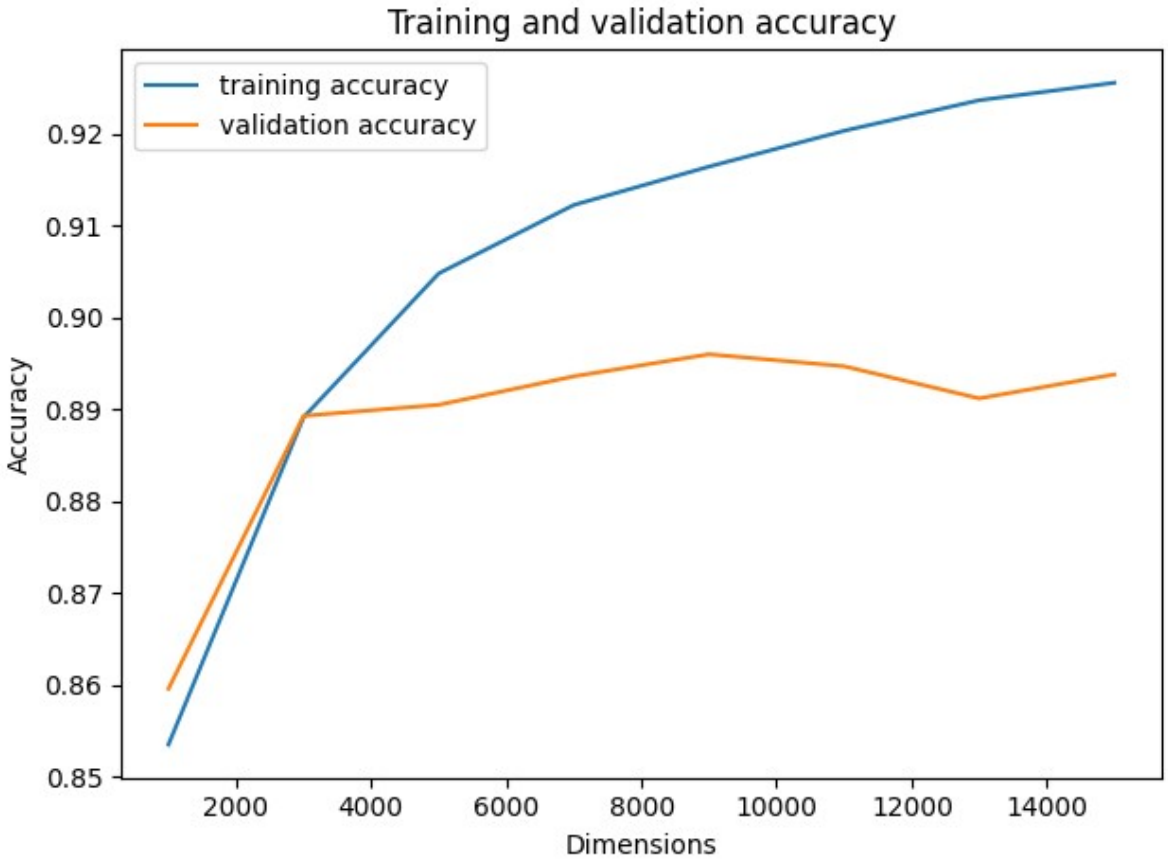

Рисунок 1 – График точности модели

Как видно из графика в обучении модели достигается пиковая точность на проверочных данных, равная около 0.89 при размерности, равной 3000. При дальнейшей увеличении размерности изменения в точности на проверочных данных незначительны.

3. Была написана функция, которая загружала пользовательский текст из файла и позволяла оценить отзыв. Для проверки был оценен следующий отзыв:

«Truly a masterpiece, The Best Hollywood film of 2019, one of the Best films of the decade.»

Результат оценки ИНС — 0.6191652, что говорит о том, что данный отзыв — положительный.

Вывод.

В ходе выполнения данной работы была создана ИНС для прогноза успеха фильмов по отзывам. Также были исследованы результаты при различном размере вектора представления текста. По результатам можно сделать вывод о том, что оптимальная точность достигается при длине вектора, равной 3000 и выше, однако, при увеличении длины точность на проверочных данных меняется незначительно и не улучшается. Также была написана функция, загружающая пользовательский отзыв из файла. Результаты оценивания отзыва показали, что данная функция работает корректно.

Приложения

Приложение А

```
import matplotlib.pyplot as plt
import numpy as np
from keras import layers, models
from keras.datasets import imdb


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


def load_data(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(input_dim):
    model = models.Sequential()
    model.add(layers.Dense(50, activation="relu",
input_shape=(input_dim, )))
    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


dimensions = [1000, 3000, 5000, 7000, 9000, 11000, 13000,
15000]
```

```
acc = []
val_acc = []
for dim in dimensions:
    # Loading data:
    (train_x, train_y), (test_x, test_y) = load_data(dim)
    # Building model:
    model = build_model(dim)
    # Fitting model:
    history = model.fit(train_x, train_y, epochs=2,
batch_size=500, validation_data=(test_x, test_y))
    acc.append(history.history['accuracy'][1])
    val_acc.append(history.history['val_accuracy'][1])

# Plotting results:
plt.plot(dimensions, acc, label='training accuracy')
plt.plot(dimensions, val_acc, label='validation accuracy')
plt.title('Training and validation accuracy')
plt.xlabel('Dimensions')
plt.ylabel('Accuracy')
plt.legend()
plt.show()


def load_text(filename):
    # File reading:
    text = []
    file = open(filename, 'rt')
    for line in file.readlines():
        text += [s.strip(''.join(['.', ',', ':', ';', '!',
'?', '(', ')'])).lower() for s in line.strip().split()]
    file.close()
    # Encode words:
    indexes = imdb.get_word_index()
    encoded = []
    for w in text:
        if w in indexes and indexes[w] < 10000:
            encoded.append(indexes[w])
    return np.array(encoded)


text = load_text('text.txt')
(train_x, train_y), (test_x, test_y) = load_data(10000)
model = build_model(10000)
results = model.fit(train_x, train_y, epochs=2,
batch_size=500, validation_data=(test_x, test_y))
text = vectorize([text], 10000)
res = model.predict(text)
print(res)
```