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 Чемова К.А.

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 89,3%, при 5000 – 89,07%, а при 1000 – 86,06%. С уменьшением размера вектора точность падает. Графики точности и ошибок представлены на рис. 1-2, 3-4, 5-6 соответственно.

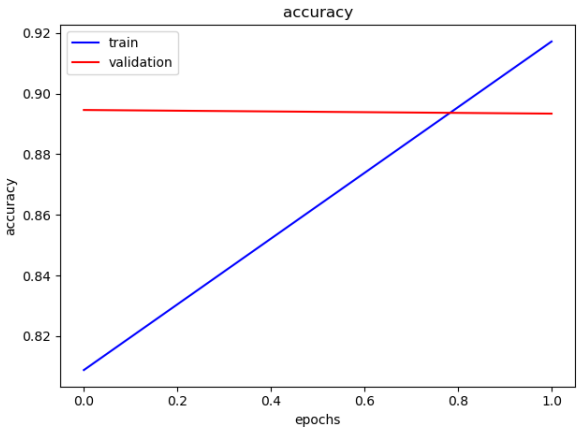

Рисунок 1 – График точности

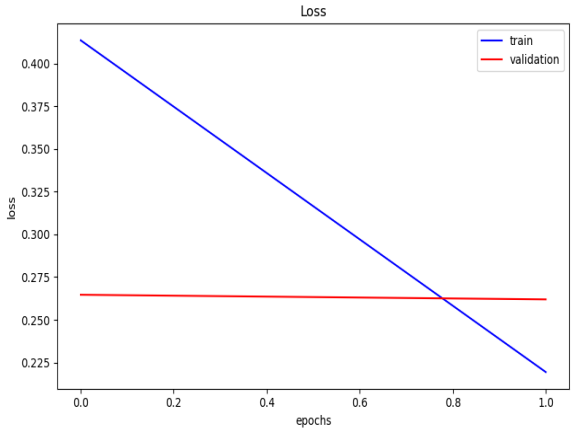

Рисунок 2 – График ошибки

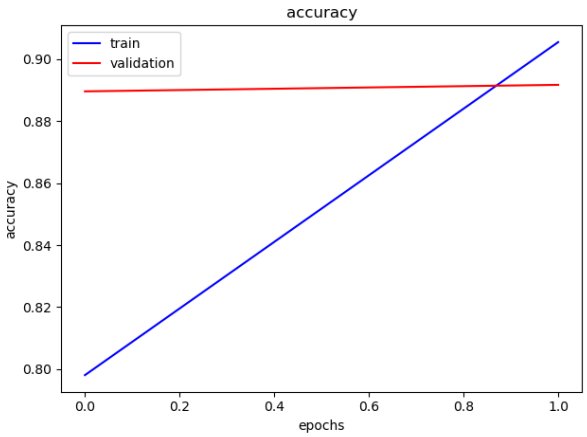

Рисунок 3 – График точности

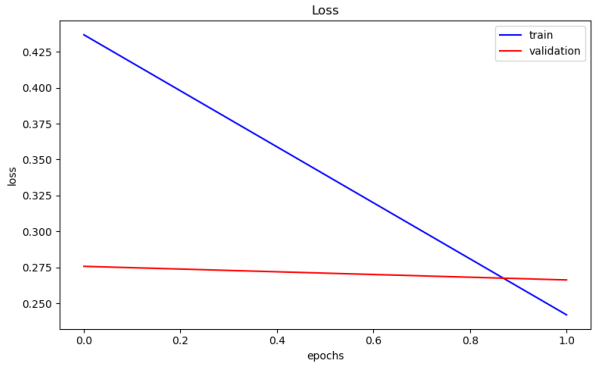

Рисунок 4 – График ошибки

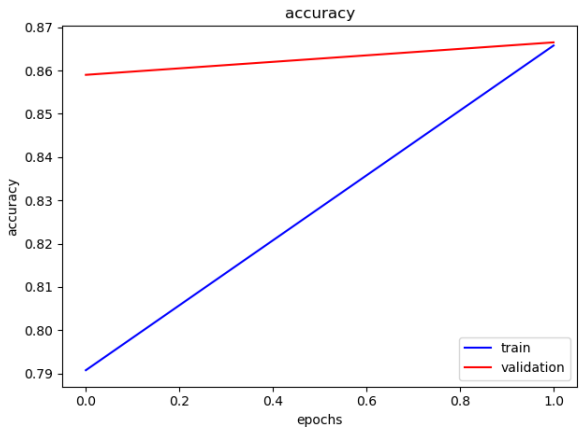

Рисунок 5 – График точности

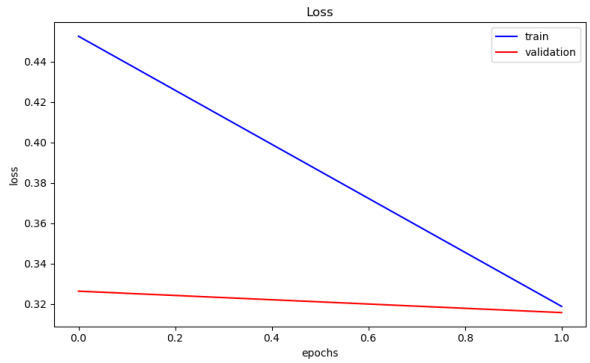

Рисунок 6 – График ошибки

Была проверена работа функции, позволяющей оценивать пользовательские комментарии. Комментарий «Very good film with amazing cast» сеть оценила на 0.71437484, что говорит о том, что отзыв положителен. А комментарий «worst movie of my life. terrible acting. the biggest failure of a

Director in his entire career. I'm not going to see any more of his movies» был оценен 0.09839169, что говорит о его отрицательности.

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.utils import to_categorical
from keras import models, Sequential, regularizers
from keras import layers
from keras.datasets import imdb
(training_data, training_targets), (testing_data, testing_targets) =
imdb.load_data(num_words=10000)
data = np.concatenate((training_data, testing_data), axis=0)
targets = np.concatenate((training_targets, testing_targets), axis=0)

print("Categories:", np.unique(targets))
print("Number of unique words:", len(np.unique(np.hstack(data))))
length = [len(i) for i in data]
print("Average Review length:", np.mean(length))
print("Standard Deviation:", round(np.std(length)))

print("Label:", targets[0])
print(data[0])

index = imdb.get_word_index()
reverse_index = dict([(value, key) for (key, value) in index.items()])
decoded = " ".join( [reverse_index.get(i - 3, "#") for i in data[0]] )
print(decoded)

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

def loadText(filename):
    punctuation = ['.',',',':',';','!','?','(',')']
    text = []
    with open(filename, 'r') as f:
        for line in f.readlines():
            text += [s.strip(''.join(punctuation)).lower() for s in
line.strip().split()]
    print(text)
    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 = loadText('bad.txt')
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()
model.add(layers.Dense(50, activation="relu", input_shape=(10000,)))
model.add(layers.Dropout(0.2, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation="linear",
```

```
kernel_regularizer=regularizers.l2()))
model.add(layers.Dropout(0.5, noise_shape=None, seed=None))
model.add(layers.Dense(100, activation="relu", kernel_regularizer=regularizers.l2()))
model.add(layers.Dropout(0.5, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation="relu"))
model.add(layers.Dense(1, activation="sigmoid"))

model.summary()

model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"])
results = model.fit(train_x, train_y, epochs=2, batch_size=500,
validation_data=(test_x, test_y))
text = vectorize([text])
res = model.predict(text)
print(np.mean(results.history["val_accuracy"]))
print(res)

plt.figure(1, figsize=(8, 5))
plt.plot(results.history['loss'], 'b', label='train')
plt.plot(results.history['val_loss'], 'r', label='validation')
plt.title('Loss')
plt.ylabel('loss')
plt.xlabel('epochs')
plt.legend()
plt.show()
plt.clf()

plt.figure(1, figsize=(8, 5))
plt.plot(results.history['accuracy'], 'b', label='train')
plt.plot(results.history['val_accuracy'], 'r', label='validation')
plt.title('accuracy')
plt.ylabel('accuracy')
plt.xlabel('epochs')
plt.legend()
plt.show()
plt.clf()
```