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 _____ Зуев 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 искусственной нейронной сети в соответствии с условиями (весь код представлен в приложении А).

Модель представлена на рис. 1.

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

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"))
```

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.

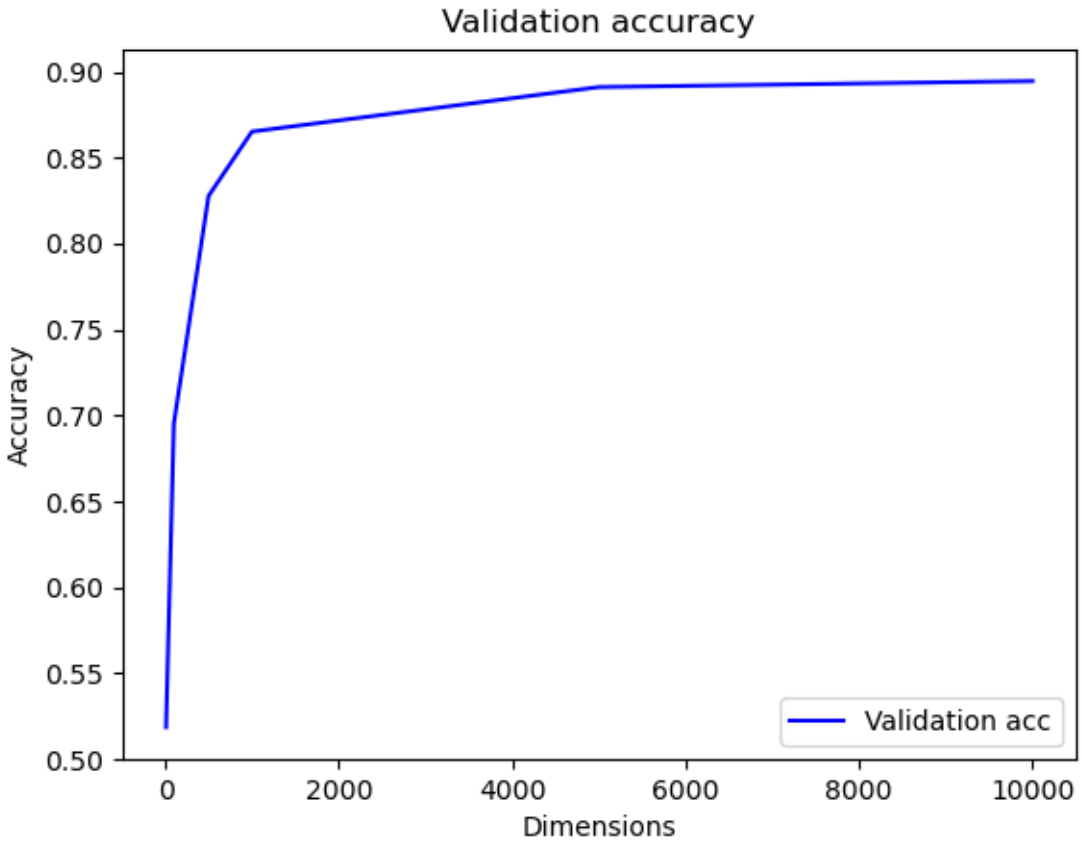

Рисунок 2 - Зависимость от размера вектора представления текста

Как видно, наибольшая точность на проверочных данных наблюдается при наибольшем размере вектора представления данных.

3. Была предпринята попытка повысить точность сети.

В первую очередь число эпох обучения было увеличено с двух до десяти. Результат обучения сети представлен на рис. 3, 4.

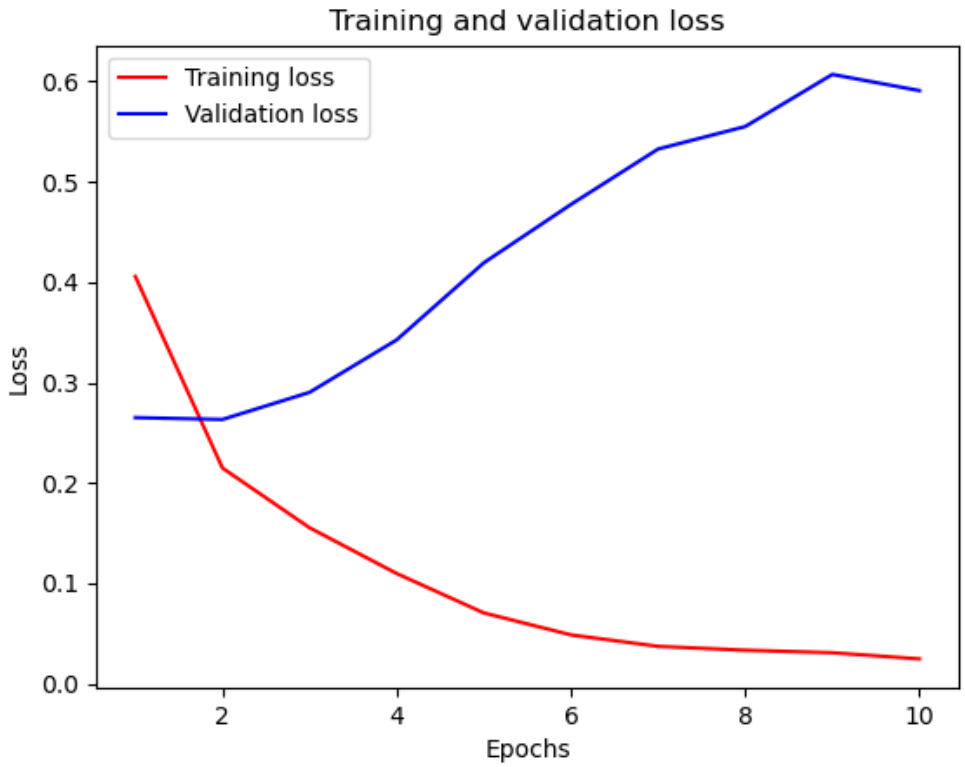

Рисунок 3 - Потери при 10 эпохах на первой модели

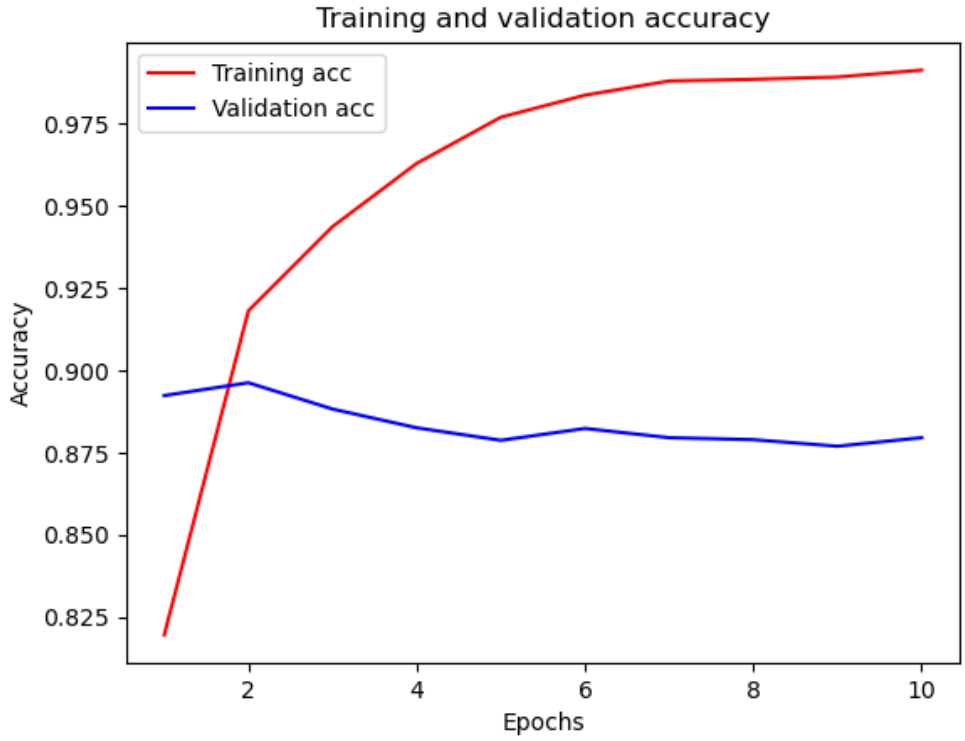

Рисунок 4 - Точность при 10 эпохах на первой модели

По результатам обучения видно, что сеть начинает переобучаться после второй эпохи.

Чтобы избавиться от переобучения была упрощена архитектура сети и увеличен шанс удаления нейрона в слое Dropout. Получившаяся архитектура сети представлена на рис. 5.

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

model.add(layers.Dropout(0.6, 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
```

Рисунок 5 - Новая архитектура сети

Сеть так же была обучена на десяти эпохах. Результат обучения сети представлен на рис. 6, 7.

Рисунок 6 - Потери при 10 эпохах на второй модели

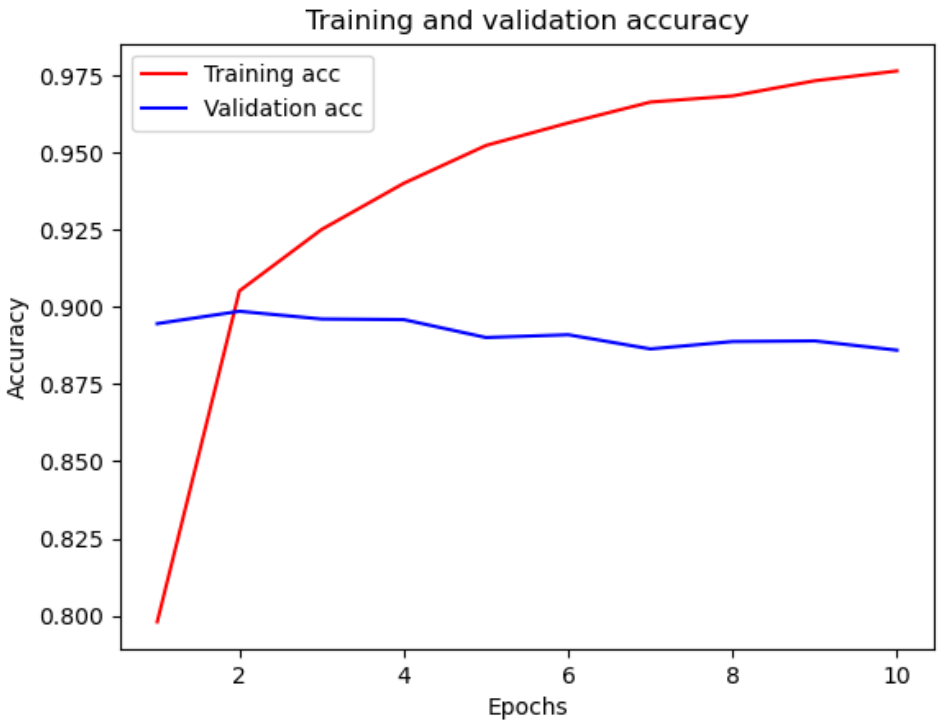

Рисунок 7 - Точность при 10 эпохах на второй модели

Как видно из результатов обучения переобучение сети при новой модели так же начинается после второй эпохи, а точность сети на второй эпохе не изменяется, поэтому было принято решение обучать сеть на двух эпохах, а архитектуру оставить начальной.

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

Тексты обзоров и соответствующие им оценки представлены на рис. 8.

```
strings = ["This film made a good impression on me. I'm glad I bought a ticket and watched this movie!",
           "This movie makes you think about the role of man on this planet. I recommend watching the movie.",
           "Sarik Andreasyan once again confirmed that he is the king of the Comedy genre. "
           "This movie is a must-see for people with a sense of humor.",
           "This Quentin Tarantino movie is as usual full of senseless violence and inappropriate behavior. "
           "This movie is terrible and should be banned from cinemas.",
           "The graphics in this movie are terrible. It is clear that this film was made with a slipshod hand.",
           "The characters in this film have absolutely no motivation, their actions are devoid of any meaning. "
           "I hope the Director of this film will not make any more films."]
values = [1, 1, 1, 0, 0, 0]
```

Рисунок 8 - Обзоры и оценки

Точность прогнозирования оценки по тексту обзора сети, определенной в п. 3, представлена на рис. 9.

```
Validation accuracy is 0.166667
```

Рисунок 9 - Точность на пользовательском тексте

По результатам прогнозирования видно, что точность достаточно низкая. Возможно, это связано с тем, что в обзорах недостаточно слов, характеризующих отрицательное или положительное отношение. Добавим их и протестируем сеть снова. Получившиеся обзоры представлены на рис. 10. Результаты тестирования представлены на рис. 11, 12.

```
strings = ["This film made a good impression on me. I'm glad I bought a ticket and watched this movie! "
           "perfect beautiful great",
           "This movie makes you think about the role of man on this planet. I recommend watching the movie. "
           "wonder exiting amazing",
           "Sarik Andreasyan once again confirmed that he is the king of the Comedy genre. "
           "This movie is a must-see for people with a sense of humor. impossible wonderful laugh",
           "This Quentin Iarantino movie is as usual full of senseless violence and inappropriate behavior. "
           "This movie is terrible and should be banned from cinemas. poor shoddy low grade",
           "The graphics in this movie are terrible. It is clear that this film was made with a slipshod hand. "
           "awful terrible substandart",
           "The characters in this film have absolutely no motivation, their actions are devoid of any meaning. "
           "I hope the Director of this film will not make any more films. worse horrible unfit"]
values = [1, 1, 1, 0, 0, 0]
```

Рисунок 10 - Обзоры и оценки после изменения

```
Validation accuracy is 0.666667
```

Рисунок 11 - Точность прогнозирования на изменившихся обзорах

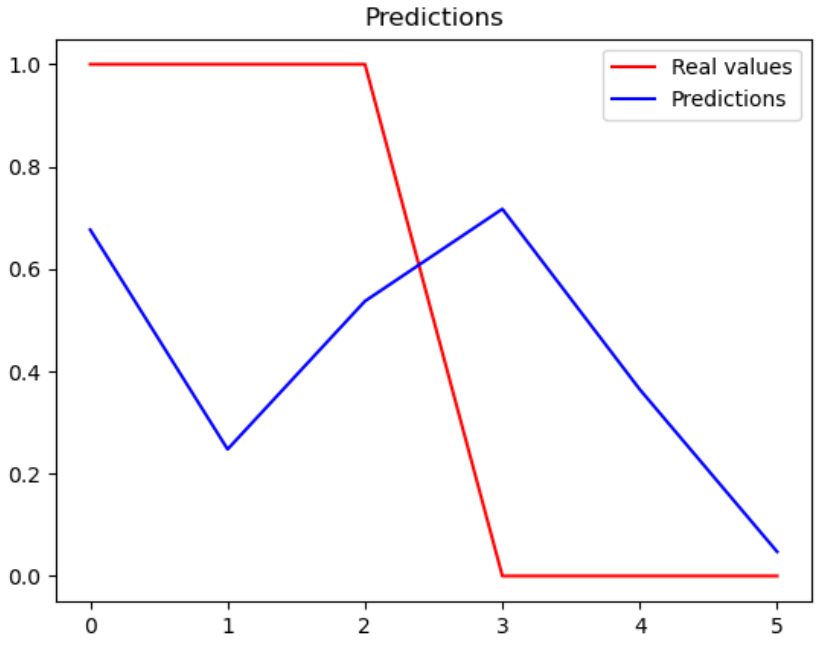

Рисунок 12 - Соответствие реальной оценки и предсказанной сетью

По результатам прогнозирования оценки изменившихся обзоров видно, что точность значительно увеличилась, это говорит о том, что сеть достаточно точно предсказывает оценку по обзору, в котором находятся слова, определяющие негативный или позитивный окрас обзора.

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 максимальным размером вектора, 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


strings = ["This film made a good impression on me. I'm glad I bought a ticket and watched this movie! "
           "perfect beautiful great",
           "This movie makes you think about the role of man on this planet. I recommend watching the movie. "
           "wonder exiting amazing",
           "Sarik Andreasyan once again confirmed that he is the king of the Comedy genre. "
           "This movie is a must-see for people with a sense of humor. impossible wonderful laugh",
           "This Quentin Tarantino movie is as usual full of senseless violence and inappropriate behavior. "
           "This movie is terrible and should be banned from cinemas. poor shoddy low grade",
           "The graphics in this movie are terrible. It is clear that this film was made with a slipshod hand. "
           "awful terrible substandart",
           "The characters in this film have absolutely no motivation, their actions are devoid of any meaning. "
           "I hope the Director of this film will not make any more films. worse horrible unfit"]
values = [1, 1, 1, 0, 0, 0]



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

```
def prepare_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(dimension):
    model = models.Sequential()
    model.add(layers.Dense(50, activation="relu", input_shape=(dimension,)))

    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 model_fit(train_x, train_y, test_x, test_y, dimension):
    model = build_model(dimension)
    H = model.fit(train_x, train_y, epochs=2, batch_size=500, validation_data=(test_x, test_y))
    return H



def test_dim(dimension):
```

```
    (train_x, train_y), (test_x, test_y) = prepare_data(dimension)
    H = model_fit(train_x, train_y, test_x, test_y, dimension)
    return H.history['val_accuracy'][-1]


def draw_plot(H):
    loss = H.history['loss']
    val_loss = H.history['val_loss']
    acc = H.history['accuracy']
    val_acc = H.history['val_accuracy']
    epochs = range(1, len(loss) + 1)
    print(len(loss))
    plt.plot(epochs, loss, 'r', label='Training loss')
    plt.plot(epochs, val_loss, 'b', label='Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
    plt.clf()
    plt.plot(epochs, acc, 'r', label='Training acc')
    plt.plot(epochs, val_acc, 'b', label='Validation acc')
    plt.title('Training and validation accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()


def test_dimensions():
    dimensions = [10, 100, 500, 1000, 5000, 10000]
    val_accuracies = []
    for dim in dimensions:
        val_accuracies.append(test_dim(dim))
    plt.plot(dimensions,  val_accuracies,  'b',  label='Validation
acc')
    plt.title('Validation accuracy')
```

```
    plt.xlabel('Dimensions')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()


def test_10000_dim():
    (train_x, train_y), (test_x, test_y) = prepare_data(10000)
    H = model_fit(train_x, train_y, test_x, test_y, 10000)
    draw_plot(H)


def text_load():
    dictionary = dict(imdb.get_word_index())
    test_x = []
    test_y = np.array(values).astype("float32")
    for string in strings:
        words   =   string.replace(',',   '   ').replace('.',   '
').replace('?', ' ').replace('\n', ' ').split()
        num_words = []
        for word in words:
            word = dictionary.get(word)
            if word is not None and word < 10000:
                num_words.append(word)
        test_x.append(num_words)
    print(test_x)
    test_x = vectorize(test_x)
    # print(test_x)
    model = build_model(10000)
    (train_x, train_y), (s1, s2) = prepare_data(10000)
    model.fit(train_x, train_y, epochs=2, batch_size=500)
    val_loss, val_acc = model.evaluate(test_x, test_y)
    print("Validation accuracy is %f" % val_acc)
    predictions = model.predict(test_x)
    plt.title("Predictions")
    plt.plot(test_y, 'r', label='Real values')
    plt.plot(predictions, 'b', label='Predictions')
```

```
    plt.legend()
    plt.show()
    plt.clf()


test_dimensions()
test_10000_dim()
text_load()
```