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. 7382 _____ Головина Е.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы.

Реализовать ИНС для прогнозирования 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.

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

Ход работы.

1. Подготовка датасета

В качестве данных используем датасет, состоящий из 50 000 обзоров фильмов на IMDb. Отрицательные обзоры отмечены нулем (0), положительные единицей (1). Данный датасет можно взять из библиотеки keras.

```
(training_data, training_targets), (testing_data, testing_targets) =
imdb.load_data(num_words=10000)
```

Так как по умолчанию разделение на тестовые и тренировочные данные 50/50, а нам нужно 20/80 (тестовые/тренировочные), то в начале соединим данные обеих категорий.

```
data = np.concatenate((training_data, testing_data), axis=0)
targets = np.concatenate((training_targets, testing_targets), axis=0)
```

Производим векторизацию – каждый обзор заполняется нулями до размера в 10 000, и преобразуем переменные в тип float.

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

Затем разделяем в отношении 20/80:

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

2. Создание модели

Для предотвращения переобучения будем использовать между слоями `dropout`. На каждом слое используется функция `dense` для полного соединения слоев друг с другом. В скрытых слоях будем использовать функцию активации `relu`, потому что это практически всегда приводит к удовлетворительным результатам. На выходном слое используем сигмоидную функцию, которая выполняет перенормировку значений в диапазоне от 0 до 1. Входной слой принимает элементы с размером 10 000, а выдает – с размером 50.

```
model = Sequential()
# Input - Layer
model.add(layers.Dense(50, activation = "relu", input_shape=(10000, )))
# 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.summary()
```

Будем использовать оптимизатор `adam`. Оптимизатор — это алгоритм, который изменяет веса и смещения во время обучения. В качестве функции потерь используем бинарную кросс-энтропию (так как мы работаем с бинарной классификацией), в качестве метрики оценки — точность.

Настройка параметров обучения:

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

Обучать модель будем с размером партии 500 и только двумя эпохами, поскольку модель начинает переобучаться, если тренировать ее дольше. Размер партии определяет количество элементов, которые будут распространяться по сети, а эпоха — это один проход всех элементов датасета. Обычно больший размер партии приводит к более быстрому обучению, но не всегда – к быстрой сходимости. Меньший размер партии обучает медленнее, но может быстрее сходиться. Выбор того или иного варианта определенно зависит от типа решаемой задачи, и лучше попробовать каждый из них.

Обучение:

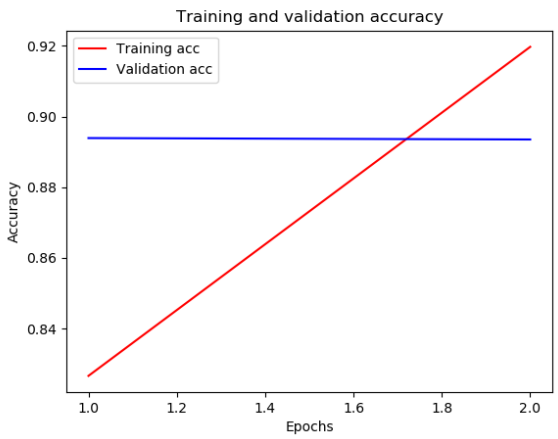
```
results = model.fit(
 train_x, train_y,
 epochs= 2,
 batch_size = 500,
 validation_data = (test_x, test_y)
)
```

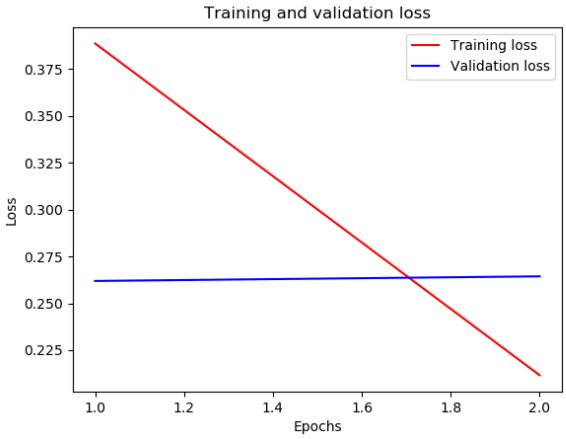
3. Результаты работы и эксперименты

Запустим обучение нейронной сети при стандартной конфигурации:

Результат: Accuracy: 89%

Графики точности и потерь:



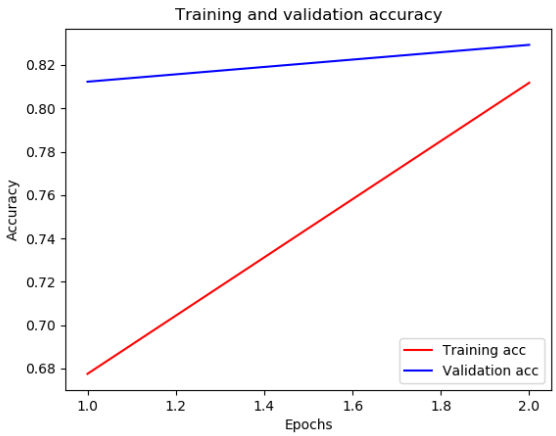


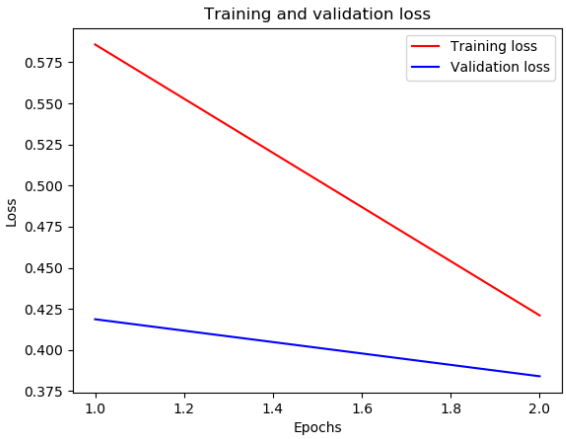
Уменьшим размер вектора представления текста.

Размер = 500.

Результат: Accuracy: 82%

Графики точности и потерь:

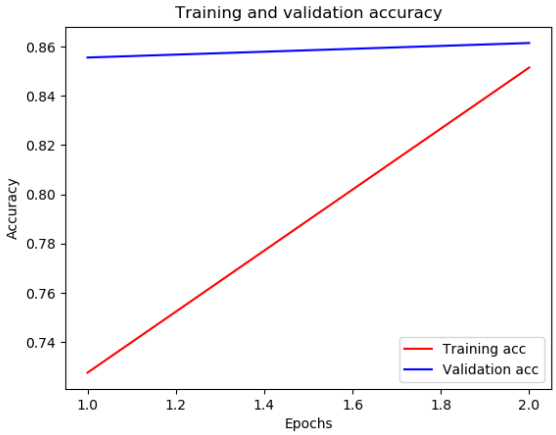


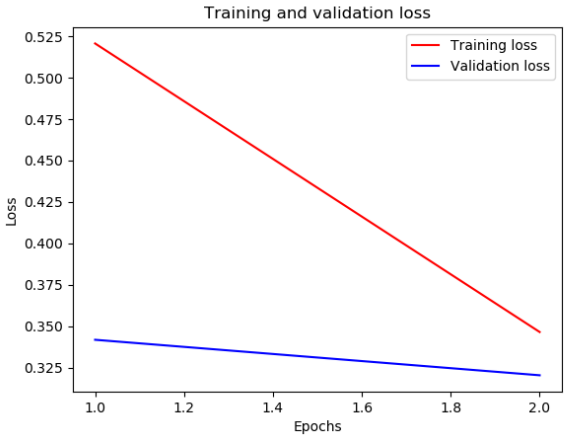


Размер = 1000.

Результат: Accuracy: 86%

Графики точности и потерь:

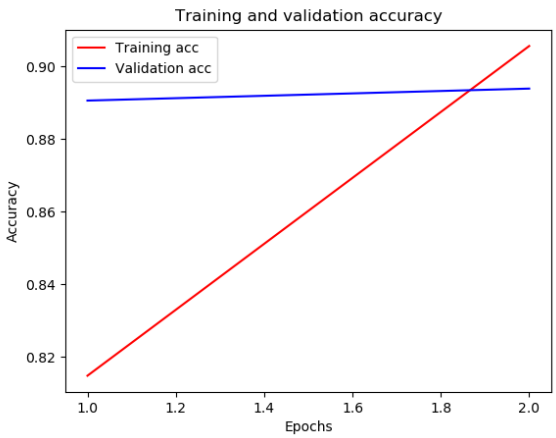


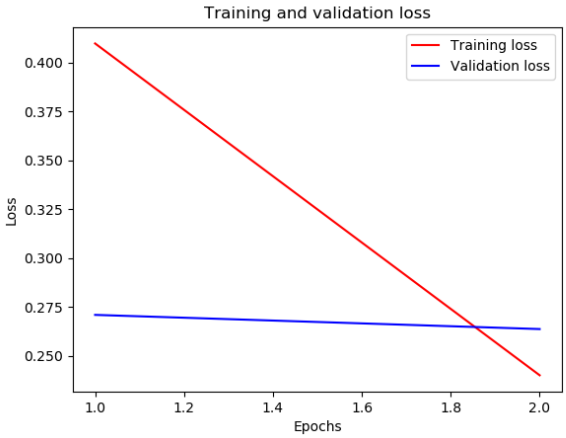


Размер = 5000.

Результат: Accuracy: 89%

Графики точности и потерь:



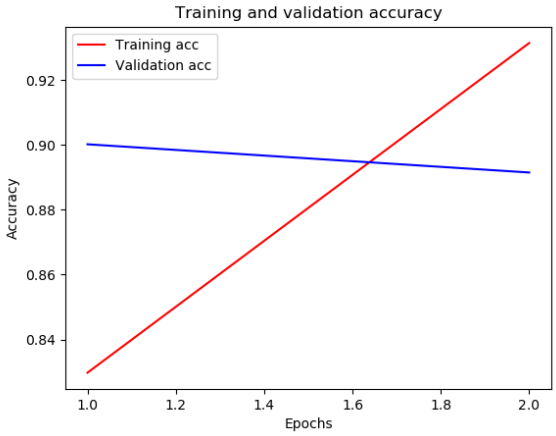


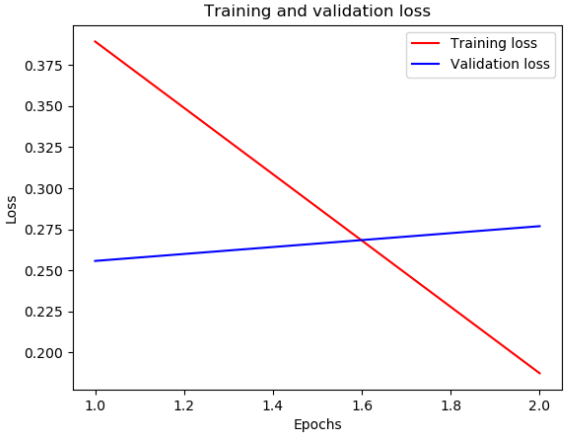
Увеличим размер вектора представления текста.

Размер = 20 000.

Результат: Accuracy: 90%

Графики точности и потерь:

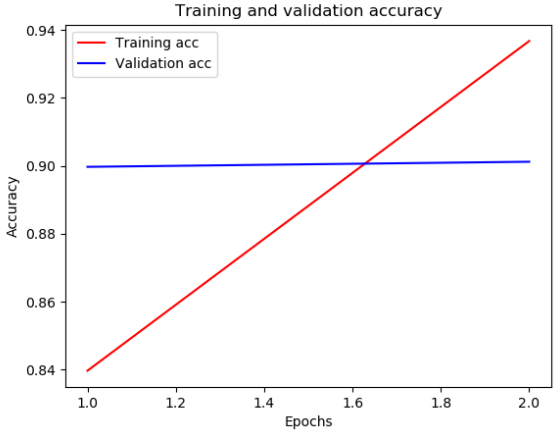


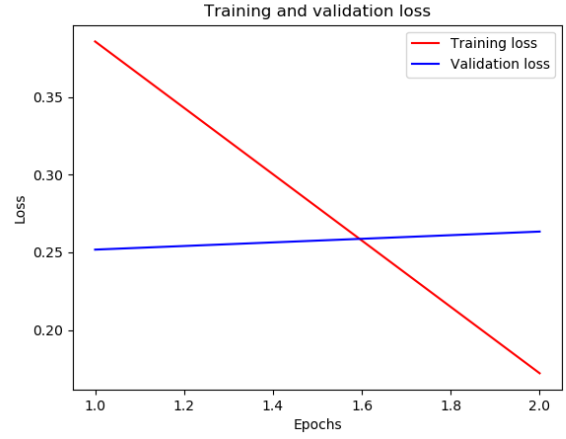


Размер = 40 000.

Результат: Accuracy: 90%

Графики точности и потерь:





4. Проверка со своими данными

Напишем функцию, конвертирующую строку в список индексов слов. Функция выглядит следующим образом. Принимает на вход словарь и строчку и возвращает список соответствующих индексов для каждого слова.

```
def str_to_list(review, dictionary):
    tt = str.maketrans(dict.fromkeys(string.punctuation))
    review = review.translate(tt)
    temp_list = review.lower().split()
    for i in range(len(temp_list)):
        if temp_list[i] in dictionary:
            temp_list[i] = dictionary[temp_list[i]]
        else:
            temp_list[i] = 0
    return temp_list
```

Соответственно, запустив данную функцию для каждой строчки в цикле и применив затем функцию vectorize – получим данные, по которым можно предсказывать положительный или отрицательный отзыв был получен.

```
reviews = [
    'Movie is great. It was very exciting.',
    'Bad movie. I hate it.',
    'It is an ordinary film. I saw tons of it. Nothing special.',
    'I love this film. It is my favourite now!'
    ]
for i in range(len(reviews)):
    reviews[i] = str_to_list(reviews[i], index)
reviews = vectorize(reviews, text_length)
prediction = model.predict(reviews)
```

Результат для этих данных получается:

[[0.81832576]

[0.2331516]

[0.22665325]

[0.61604553]]

Что соответствует:

1 : positive review

2 : negative review

3 : negative review

4 : positive review

Результат корректен.

Вывод.

В ходе выполнения работы было реализована ИНС, проверяющая положительный или отрицательный отзыв был получен. Для этого создали искусственную нейронную сеть с помощью библиотеки keras в python, протестировали получившуюся модель, затем, проведя ряд экспериментов, проверили влияние длины входного вектора на результат. Также проверили модель на своих данных, написав для этих целей соответствующую функцию.