МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №7

по дисциплине «Искусственные нейронные сети»

Тема: Классификация обзоров фильмов

Студентка гр. 7382 Головина Е.С.

Преподаватель Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы.

Реализовать рекуррентную нейронную сеть для классификации обзоров фильмов.

Задачи.

- Ознакомиться с рекуррентными нейронными сетями
- Изучить способы классификации текста
- Ознакомиться с ансамблированием сетей
- Построить ансамбль сетей, который позволит получать точность не менее 97%

Требования.

- Найти набор оптимальных ИНС для классификации текста
- Провести ансамблирование моделей
- Написать функцию/функции, которые позволят загружать текст и получать результат ансамбля сетей
- Провести тестирование сетей на своих текстах (привести в отчете)

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.

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

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)
```

Разделим в отношении 20/80:

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

Сделаем каждый обзор единого размера:

```
max_review_length = 500
train_x = sequence.pad_sequences(train_x, maxlen=max_review_length)
test_x = sequence.pad_sequences(test_x, maxlen=max_review_length)
```

2. Создание модели

2.1 Простая рекуррентная сеть

Первый слой – это Embedded, который использует 32 вектора длины для представления каждого слова. Следующий уровень – это слой LSTM с 100 единицами памяти (умными нейронами). Наконец, поскольку это проблема классификации, мы используем плотный выходной слой с одним нейроном и сигмоидной функцией активации, чтобы сделать 0 или 1 прогноз для двух классов (хорошего и плохого) в задаче.

```
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length,
input_length=max_review_length))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
```

Настройка параметров обучения:

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

Обучение:

```
results = model.fit(train_x, train_y, validation_data=(test_x, test_y),
epochs=3, batch_size=64)
```

2.2 Рекуррентная сверточная сеть

Сверточные нейронные сети (CNN) хорошо изучают пространственную структуру во входных данных, а данные обзора IMDB действительно имеют одномерную пространственную структуру (последовательность слов в обзорах). CNN может быть в состоянии выбрать инвариантные особенности для хорошего и плохого настроения. Эти изученные пространственные особенности могут затем быть изучены как последовательности уровнем LSTM.

Используем одномерный слой CNN и MaxPooling1D после слоя Embedding, которые затем передают объединенные элементы в LSTM.

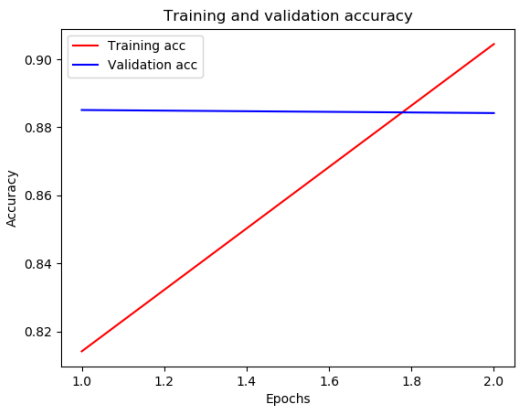
```
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length, input_length=max_r
eview_length))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='r
elu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.3))
model.add(LSTM(100))
model.add(Dropout(0.3))
model.add(Dense(1, activation='sigmoid'))
```

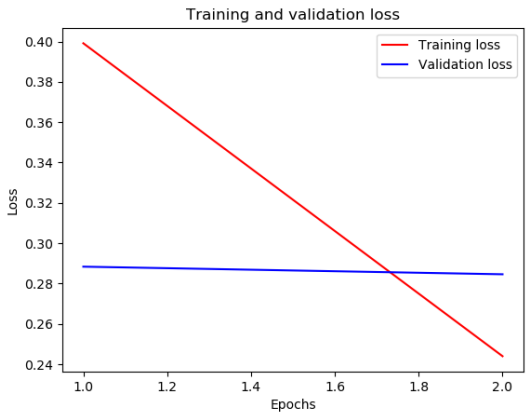
3. Результаты работы и эксперименты

Обучим первую модель, добавив слой Dropout и посмотрим на результат:

test_acc: 0.8841999769210815 = 88.4%

test_loss: 0.2846105741262436

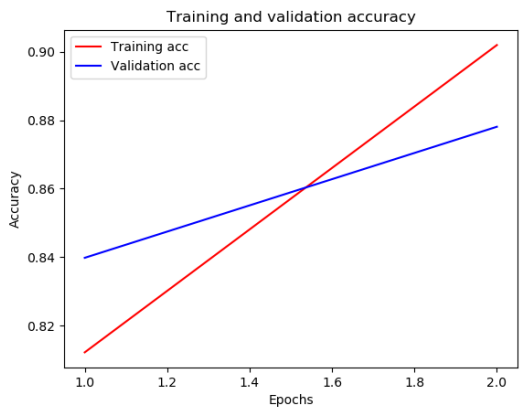


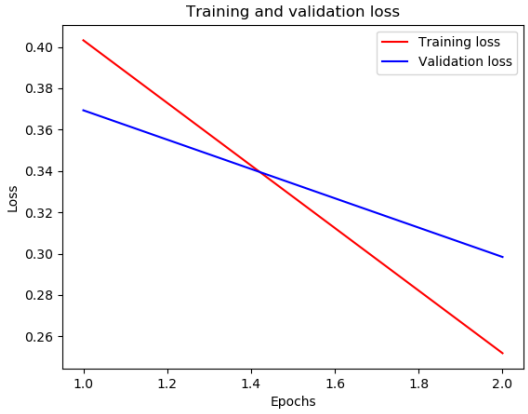


Теперь добавим еще один слой dense, как в предыдущей лабораторной работе с еще одним слоем Dropout:

test_acc: 0.8780999779701233 = 87.8%

test_loss: 0.2984172694444656



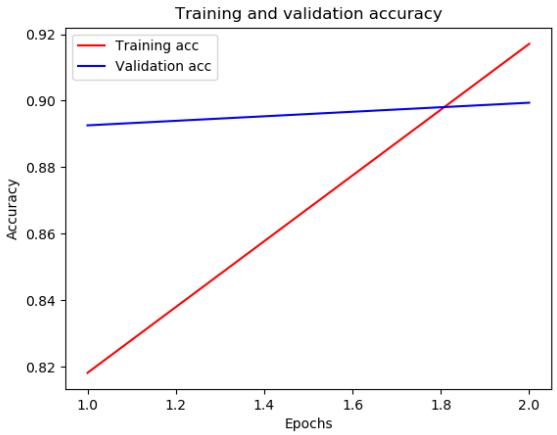


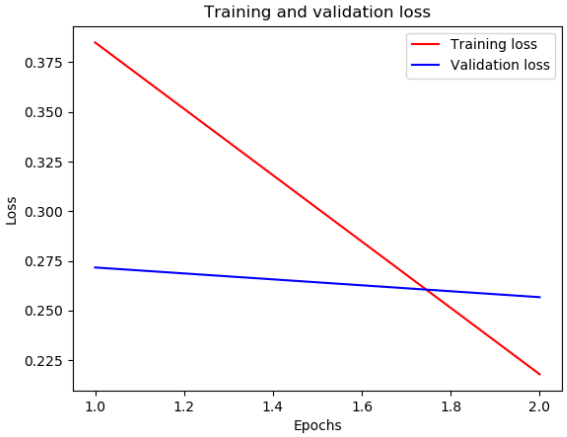
Результат ухудшился и к тому же переобучение усилилось, так что оставим предыдущую модель.

Теперь запустим обучение второй модели:

test_acc: 0.894599974155426 = 89.5%

test_loss: 0.26208437711000443





Получился почти такой же результат как в первом варианте первой модели, но обучение прошло быстрее.

Теперь проведем ансамблирование моделей, т.е. проведем обучение двух моделей, а затем получим среднее от двух предсказаний. Функция для получения значения точности:

```
def get_acc(test_y, predictions):
    all = len(test_y)
    correct = 0
    for i in range(len(test_y)):
        if test_y[i] == predictions[i]:
            correct = correct+1
    return correct/all
```

Predictions рассчитано следующим образом:

```
predictions = [model.predict(test_x) for model in models]
predictions = np.array(predictions).mean(axis = 0)
predictions = np.round(predictions).astype(int)
predictions = predictions.flatten()
```

Точность в этом случае получилась равной: 0.8966 = 89.7 %, что немного лучше, чем выдает каждая из моделей по одиночке.

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
```

Также была написана функция для тестирования отзывов на двух моделях и на ансамбле из них.

```
def test_custom_reviews(reviews, dictionary, max_review_length):
    for i in range(len(reviews)):
        reviews[i] = str_to_list(reviews[i], dictionary)
    reviews = sequence.pad_sequences(reviews, maxlen=max_review_length)
    num_models = 2
    models = load_models(num_models)
    separate_predictions = [model.predict(reviews) for model in models]
    predictions = np.array(separate_predictions).mean(axis = 0)
    for i in range(num_models):
        print("model",i+1," prediction: ",separate_predictions[i])
    print("ensemle prediction: ", predictions)
```

Обзоры выглядят следующим образом:

```
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!'
    ]
```

Понятно, что 1 и 4 – положительные, а 2 и 3 отрицательные отзывы.

Результат на 1 модели:

[[0.8734284][0.525199][0.32867455][0.6824992]]

Произошла ошибка, второй отзыв модель охарактеризовала как положительный.

Результат 2 модели:

[[0.67681265][0.26600316][0.2539224][0.6570965]] – все верно.

Результат для ансамбля 1 и 2 модели:

[[0.7751205][0.3956011][0.29129848][0.66979784]] – все верно.

Таким образом, результат полученный с помощью 2 модели и ансамбля корректен.

Вывод.

В ходе выполнения работы было реализовано две модели ИНС, проверяющие положительный или отрицательный отзыв на фильм был получен. Также для проверки отзывов был использован ансамбль из двух моделей, который показал наилучший результат. Также модели и ансамбль из двух моделей были протестированы на своих данных, для этого были написаны специальные функции.