**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. 8383 |  | Киреев К.А. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2021

**Цель работы**

Прогноз 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%

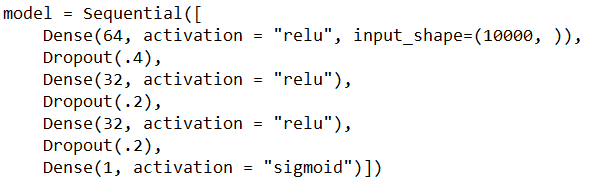
**Требования**

* Построить и обучить нейронную сеть для обработки текста
* Исследовать результаты при различном размере вектора представления текста
* Написать функцию, которая позволяет ввести пользовательский текст

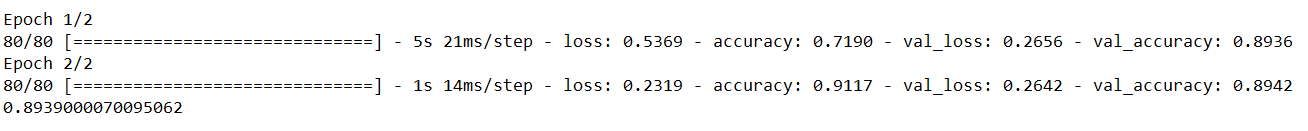
**Ход работы**

***Построение и обучение нейронной сети***

Была реализована архитектура нейронной сети. Также модель представлена в файле model.png.



Была достигнута точность 89% на валидационных данных.

Модель обучалась на протяжении 2 эпох, пакетами по 500 образцов. На рисунке 1 можно увидеть, что при увеличении количества эпох обучения действительно наступает переобучение после 2 эпохи.

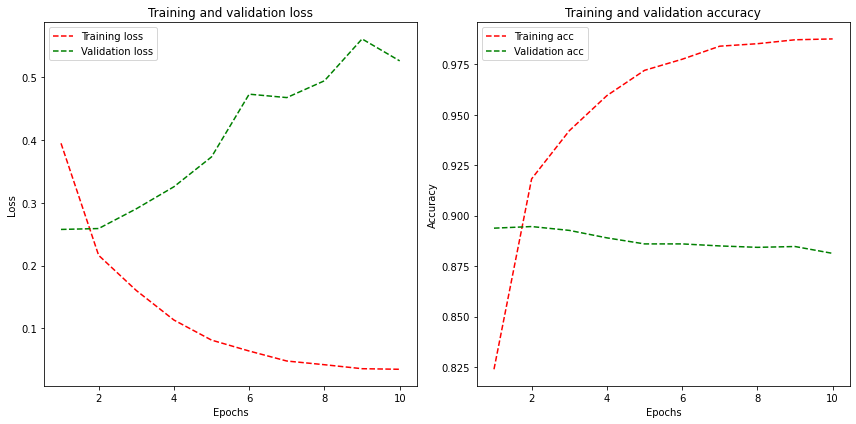
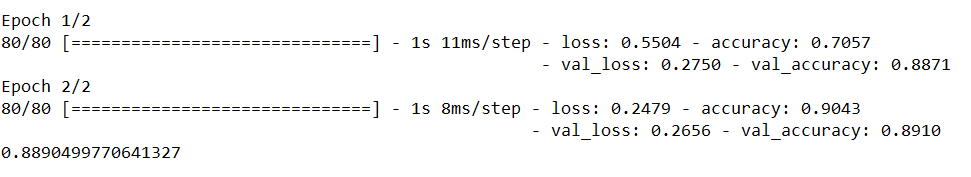


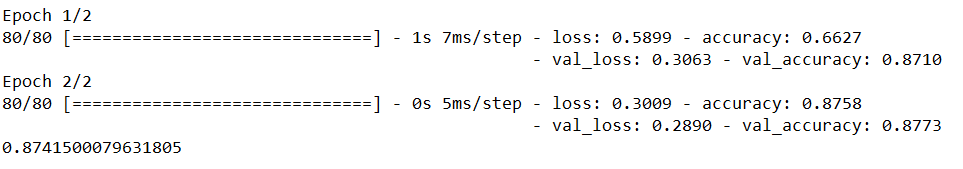
Рис. 1 – Переобучение нейронной сети

***Исследование результатов при различном размере вектора представления текста***

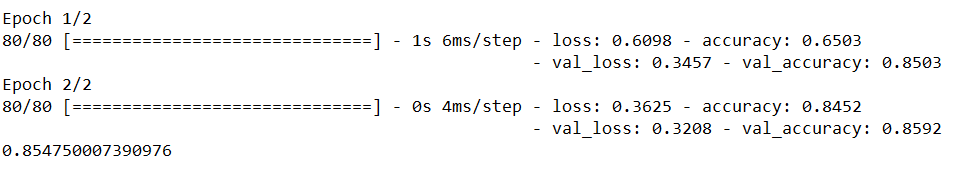
Исследуем результаты обучения нейросети при измененном размере вектора представления. При уменьшении размера вектора представления текста с 10000 до 5000 результаты ухудшились совсем незначительно до 88.9%.



При дальнейшем уменьшении размера вектора до 2000 результаты ухудшились уже до 87.4%.



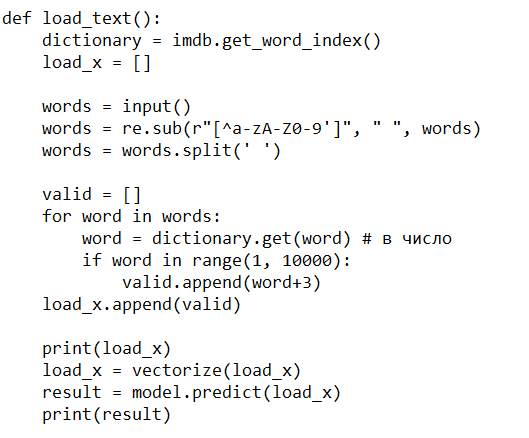
Результаты при уменьшении до 1000 представлены ниже.



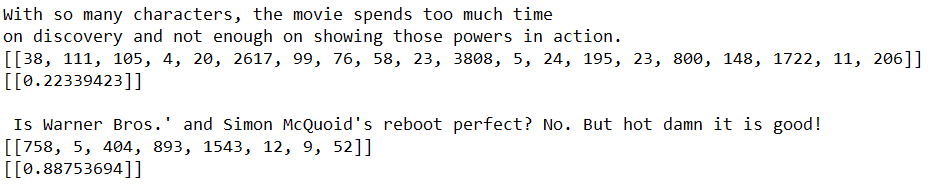
Можно сделать вывод, что при недостаточном размере вектора представления текста результаты работы нейросети ухудшаются.

***Функция ввода пользовательского текста***

Была написана функция для ввода пользовательского текста load\_text()



Получаем словарь со словами и их индексами. Далее обрабатываем введенный текст, удаляя лишние символы. Заменяем числа на их индексы, оставляя только 10000 самых частых слов. Далее векторизуем текст, написанной ранее функцией и выводим результат предсказания нейронной сети. Пример работы представлен ниже.



**Выводы**

В ходе выполнения лабораторной работы была создана нейронная сеть для прогнозирования успеха фильма по обзору. Была реализована функция для ввода пользовательского текста. Был изучен один из способов представления текста для передачи в нейронную сеть.

Приложение А. Исходный код программы

import os

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '2'

os.environ['TF\_FORCE\_GPU\_ALLOW\_GROWTH'] = 'true'

import re

import matplotlib

import matplotlib.pyplot as plt

from matplotlib import gridspec

import numpy as np

from keras.utils import to\_categorical

from keras.layers import Dense, Dropout

from keras.models import Sequential

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()

print(index['good'])

reverse\_index = dict([(value, key) for (key, value) in index.items()])

print(reverse\_index[49])

print(reverse\_index[14-3])

print([i for i in data[0]])

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

print(type(data))

data = vectorize(data)

targets = np.array(targets).astype("float16")

test\_x = data[:10000]

test\_y = targets[:10000]

train\_x = data[10000:]

train\_y = targets[10000:]

model = Sequential([

Dense(64, activation = "relu", input\_shape=(10000, )),

Dropout(.4),

Dense(64, activation = "relu"),

Dropout(.2),

Dense(32, activation = "relu"),

Dropout(.1),

Dense(1, activation = "sigmoid")])

# model.save('model.h5')

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))

print(np.mean(results.history["val\_accuracy"]))

def draws(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)

fig = plt.figure(figsize=(12, 6))

gs = gridspec.GridSpec(1, 2, width\_ratios=[3, 3])

plt.subplot(gs[0])

plt.plot(epochs, loss, 'r--', label='Training loss')

plt.plot(epochs, val\_loss, 'g--', label='Validation loss')

plt.title('Training and validation loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.subplot(gs[1])

plt.plot(epochs, acc, 'r--', label='Training acc')

plt.plot(epochs, val\_acc, 'g--', label='Validation acc')

plt.title('Training and validation accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

plt.tight\_layout()

plt.show()

draws(results)

def load\_text():

dictionary = imdb.get\_word\_index()

load\_x = []

words = input()

words = re.sub(r"[^a-zA-Z0-9']", " ", words)

words = words.split(' ')

valid = []

for word in words:

word = dictionary.get(word) # в число

if word in range(1, 10000):

valid.append(word+3)

load\_x.append(valid)

print(load\_x)

load\_x = vectorize(load\_x)

result = model.predict(load\_x)

print(result)

load\_text()