Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет

телекоммуникаций и информатики»

Кафедра ПМиК

Методы машинного обучения

Лабораторная работа №4

Разработка нейронной сети

Выполнил: студент 4 курса

Ф. ИВТ, группа: ИП-711

Мартасов И. О.

Проверил: доцент кафедры ПМиК

Ракитский Антон Андреевич

Новосибирск, 2020

**Содержание**

1. Задание.
2. Текст программы.
3. Результаты работы

**Задание**

Целью данной лабораторной работы является разработка нейронной

сети для решения задачи классификации или регрессии в зависимости от

набора данных в рамках варианта. Лабораторная работа предполагает

разработку на языке программирования Python с использованием

библиотеки Keras.

Задание:

Определение эмоционального окраса рецензии фильма

(IMDB movie review sentiment classification dataset)

Нейросеть должна состоять из трёх полносвязных слоёв,

обязательное использование Dropout, в качестве оптимизатора

использовать Adam;

**Текст программы**

import numpy as np

from keras.datasets import imdb

from keras.models import Sequential

from keras.layers import Dense, Dropout, Activation

from keras.preprocessing.text import Tokenizer

from keras.utils import to\_categorical

import matplotlib.pyplot as plt

(x\_train, y\_train), (x\_test, y\_test) = imdb.load\_data(num\_words=1000)

tokenizer = Tokenizer(num\_words=1000)

x\_train = tokenizer.sequences\_to\_matrix(x\_train, mode='binary')

x\_test = tokenizer.sequences\_to\_matrix(x\_test, mode='binary')

y\_train = to\_categorical(y\_train, num\_classes = 2)

y\_test = to\_categorical(y\_test, num\_classes = 2)

model = Sequential()

model.add(Dense(128, activation='relu', input\_shape=(x\_train.shape[1],)))

model.add(Dense(64, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(2, activation='softmax'))

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

model.summary()

model.fit(x\_train, y\_train, batch\_size=50, epochs=15, validation\_split=0.2, verbose=1)

L=len(y\_test)

correct=0

YP=model.predict(x\_test)

for i in range(L):

  y1=np.argmax(y\_test[i])

  y2=np.argmax(YP[i])

  if y1==y2:

    correct += 1

print(correct, ' ', L)

print("Accuracy: ", correct / L \* 100)

prediction=model.predict(x\_test)

print(prediction)

idx=5

index = imdb.get\_word\_index()

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

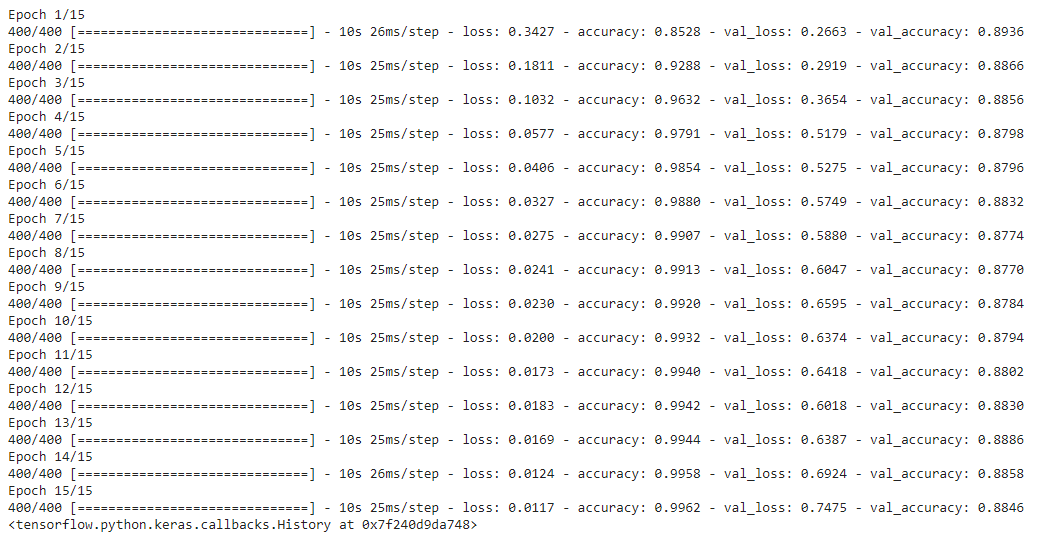
decoded = " ".join( [reverse\_index.get(i - 3, "#") for i in x[idx]] )

print(decoded)

print((prediction[idx, np.argmax(prediction[29])] < 0.5).astype("int32"))

**Результаты работы**

**Процесс обучения сети:**

****

**Точность предсказания:**

****

**Проверка работы сети:**

****