Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

Отчёт

по лабораторной работе № 4

«Разработка нейронной сети»

Выполнил:

студент группы ИП-013

Копытина Татьяна

Алексеевна

Проверил работу:

Дементьева Кристина Игоревна

Новосибирск 2023 г.

Оглавление

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.

Определение эмоционального окраса рецензии фильма

(IMDB movie review sentiment classification dataset)

При разработке нейронной сети следует соблюсти наличие необходимых составляющих исходя из следующих вариантов:

Нейросеть должна состоять из четырех полносвязных слоёв, обязательное использование GaussianDropout, в качестве оптимизатора использовать SGD;
Выбор количества нейронов на всех внутренних слоях, функций активации и других параметров должен быть обусловлен оптимальностью работы модели.

Код программы:

```
import matplotlib

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


from keras.layers import Dense, GaussianDropout, LeakyReLU, ELU

from keras.preprocessing import sequence

import keras

import matplotlib.pyplot as plt

from keras.optimizers import SGD

import numpy as np

from keras.datasets import imdb

from sklearn.preprocessing import StandardScaler

from keras.regularizers import l2

from keras.layers import Activation


(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)


sgd = SGD(learning_rate = 0.01)

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 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
```

```
data = vectorize(data)

targets = np.array(targets).astype("float32")

test_x = data[:10000]

test_y = targets[:10000]

train_x = data[10000:]

train_y = targets[10000:]

model = models.Sequential()

# Input - Layer


#model.add(layers.Dense(50, activation = "relu", input_shape=(10000, )))


# Hidden - Layers

model.add(Dense(64, activation=LeakyReLU(alpha=0.1), input_shape=(10000,)))

model.add(GaussianDropout(0.3))

model.add(Dense(64, activation=LeakyReLU(alpha=0.1)))

model.add(GaussianDropout(0.2))

model.add(Dense(64, activation=LeakyReLU(alpha=0.1)))

model.add(GaussianDropout(0.1))


# Output- Layer

model.add(layers.Dense(1, activation = "sigmoid"))

model.summary()


# compiling the model

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

history = model.fit(train_x, train_y,epochs= 15,batch_size = 500,validation_data = (test_x,
test_y))

# Визуализация результатов обучения

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)

plt.plot(history.history['accuracy'], label='Точность на обучении')

plt.plot(history.history['val_accuracy'], label='Точность на тесте')

plt.legend()

plt.xlabel('Эпохи')

plt.ylabel('Точность')

plt.title('Точность на обучении и тесте')


plt.subplot(1, 2, 2)
```

```
plt.plot(history.history['loss'], label='Потери на обучении')
plt.plot(history.history['val_loss'], label='Потери на тесте')
plt.legend()
plt.xlabel('Эпохи')
plt.ylabel('Потери')
plt.title('Потери на обучении и тесте')
```

Результат работы программы:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 64)                640064

 gaussian_dropout (Gaussian  (None, 64)                0
 Dropout)

 dense_1 (Dense)             (None, 64)                4160

 gaussian_dropout_1 (Gaussi  (None, 64)                0
 anDropout)

 dense_2 (Dense)             (None, 64)                4160

 gaussian_dropout_2 (Gaussi  (None, 64)                0
 anDropout)

 dense_3 (Dense)             (None, 1)                 65

=================================================================
Total params: 648449 (2.47 MB)
Trainable params: 648449 (2.47 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
Epoch 1/15
80/80 [==============================] - 6s 66ms/step - loss: 0.6923 - accuracy: 0.5186 - val_loss: 0.6903 - val_accuracy: 0.5560
Epoch 2/15
80/80 [==============================] - 4s 48ms/step - loss: 0.6895 - accuracy: 0.5513 - val_loss: 0.6870 - val_accuracy: 0.6085
Epoch 3/15
80/80 [==============================] - 4s 47ms/step - loss: 0.6857 - accuracy: 0.5832 - val_loss: 0.6817 - val_accuracy: 0.6592
Epoch 4/15
80/80 [==============================] - 4s 54ms/step - loss: 0.6796 - accuracy: 0.6162 - val_loss: 0.6735 - val_accuracy: 0.6993
Epoch 5/15
80/80 [==============================] - 4s 54ms/step - loss: 0.6715 - accuracy: 0.6463 - val_loss: 0.6608 - val_accuracy: 0.7308
Epoch 6/15
80/80 [==============================] - 4s 48ms/step - loss: 0.6580 - accuracy: 0.6752 - val_loss: 0.6420 - val_accuracy: 0.7552
Epoch 7/15
80/80 [==============================] - 5s 57ms/step - loss: 0.6380 - accuracy: 0.7024 - val_loss: 0.6144 - val_accuracy: 0.7697
Epoch 8/15
80/80 [==============================] - 4s 49ms/step - loss: 0.6137 - accuracy: 0.7221 - val_loss: 0.5788 - val_accuracy: 0.7836
Epoch 9/15
80/80 [==============================] - 4s 47ms/step - loss: 0.5824 - accuracy: 0.7438 - val_loss: 0.5373 - val_accuracy: 0.7950
Epoch 10/15
80/80 [==============================] - 4s 56ms/step - loss: 0.5456 - accuracy: 0.7647 - val_loss: 0.4947 - val_accuracy: 0.8072
Epoch 11/15
80/80 [==============================] - 4s 52ms/step - loss: 0.5116 - accuracy: 0.7786 - val_loss: 0.4570 - val_accuracy: 0.8194
Epoch 12/15
80/80 [==============================] - 4s 48ms/step - loss: 0.4783 - accuracy: 0.7957 - val_loss: 0.4253 - val_accuracy: 0.8293
Epoch 13/15
80/80 [==============================] - 4s 46ms/step - loss: 0.4525 - accuracy: 0.8056 - val_loss: 0.4009 - val_accuracy: 0.8384
Epoch 14/15
80/80 [==============================] - 6s 71ms/step - loss: 0.4291 - accuracy: 0.8181 - val_loss: 0.3817 - val_accuracy: 0.8441
Epoch 15/15
80/80 [==============================] - 3s 44ms/step - loss: 0.4067 - accuracy: 0.8302 - val_loss: 0.3659 - val_accuracy: 0.8494
```

Рис1,2 процесс обучения модели на выборках

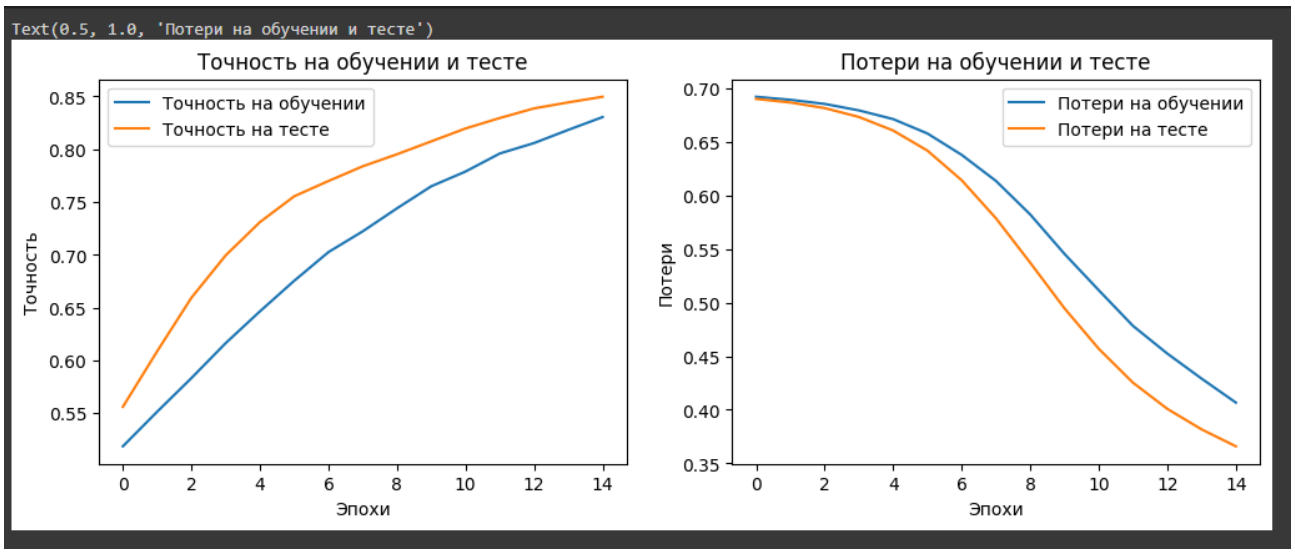

Рис3. графическое представление обучения.

Выводы:

В ходе работы я изучила и применила на практике способы обучения нейронной сети на больших данных, с помощью метода GaussianDropout, применения 4 слоев для обучения и разбиения данных на тестовые и обучающие выборки.