Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа №7
по дисциплине
«Методы машинного обучения»
на тему
«Использование нейронных сетей для анализа текста»

Выполнил:
студент группы ИУ5-24М
Зубаиров В. А.

_____

Москва — 2020 г.

1. Задание на ЛР

Итоговый код для обучения нейросети и оценки ее точности содержится в приложении. Необходимо увеличить количество скрытых слоев до 3-ех, а количество нейронов в этих слоях так, чтобы обеспечить точность работы нейросети не менее 75%. Темы текстов необходимо изменить в соответствии с вариатом: Вариант 5 - rec.autos, rec.sport.hockey, sci.crypt, sci.med, talk.religion.misc

2. Выполнение

[0]:
```
import numpy as np
from collections import Counter
```

[0]:
```
def get_word_2_index(vocab):
    word2index = {}
    for i, word in enumerate(vocab):
        word2index[word] = i
    return word2index
```

[3]:
```
from sklearn.datasets import fetch_20newsgroups
categories = ["sci.space","rec.sport.hockey","sci.crypt", "sci.med", "talk.religion.misc"]
newsgroups_train = fetch_20newsgroups(subset='train', categories=categories)
newsgroups_test = fetch_20newsgroups(subset='test', categories=categories)
print('total texts in train:',len(newsgroups_train.data))
print('total texts in test:',len(newsgroups_test.data))
print('text',newsgroups_train.data[0])
print('category:',newsgroups_train.target[0])

vocab = Counter()

for text in newsgroups_train.data:
  for word in text.split(" "):
    vocab[word.lower()] += 1
for text in newsgroups_test.data:
  for word in text.split(" "):
    vocab[word.lower()] += 1

word2index = get_word_2_index(vocab)
total_words = len(vocab)
```

```
total texts in train: 2759
total texts in test: 1836
text From: ekr@kyle.eitech.com (Eric Rescorla)
Subject: Re: After 2000 years, can we say that Christian Morality is
Organization: EIT
Lines: 29
NNTP-Posting-Host: kyle.eitech.com

In article <1qjd3o$nlv@horus.ap.mchp.sni.de> frank@D012S658.uucp (Frank O'Dwyer)
```

writes:
>In article <sandvik-140493230024@sandvik-kent.apple.com#
sandvik@newton.apple.com (Kent Sandvik) writes:
>#In article <1qie61$fkt@horus.ap.mchp.sni.de>, frank@D012S658.uucp (Frank
>#O'Dwyer) wrote:
>#> Objective morality is morality built from objective values.
>#
>#You now pushed down the defintion of objectivity into realm of
>#objective values. So you need to explain that as well, as well
>#as the objective sub-parts, the objective atoms, quarks…
>Firstly, science has its basis in values, not the other way round.
You keep saying that. I do not think it means what you think it
means.
Perhaps you should explain what you think "science has it's basis
in values" means. The reason why people DO science is that
they value it's results. That does not mean that science has
it's basis in values. Any more than DES stops working if I stop
valuing my privacy.

>So you better explain what objective atoms are, and how we get them
>from subjective values, before we go any further.
See above.

-Ekr

--
Eric Rescorla ekr@eitech.com
Would you buy used code from this man?

category: 4

[4]:
```
%tensorflow_version 1.14
import tensorflow as tf
```

`%tensorflow_version` only switches the major version: 1.x or 2.x.
You set: `1.14`. This will be interpreted as: `1.x`.

TensorFlow 1.x selected.

[0]:
```
# Параметры обучения
learning_rate = 0.01
training_epochs = 15
batch_size = 150
display_step = 1
# Network Parameters
n_hidden_1 = 600 # скрытый слой
```

```
n_hidden_2 = 300 # скрытый слой
n_hidden_3 = 150
n_input = total_words # количество уникальных слов в наших текстах
n_classes = 5 # 5 классов
input_tensor = tf.placeholder(tf.float32,[None, n_input],name="input")
output_tensor = tf.placeholder(tf.float32,[None, n_classes],name="output")
```

[0]:
```
def multilayer_perceptron(input_tensor, weights, biases):
    layer_1_multiplication = tf.matmul(input_tensor, weights['h1'])
    layer_1_addition = tf.add(layer_1_multiplication, biases['b1'])
    layer_1 = tf.nn.relu(layer_1_addition)

    layer_2_multiplication = tf.matmul(layer_1, weights['h2'])
    layer_2_addition = tf.add(layer_2_multiplication, biases['b2'])
    layer_2 = tf.nn.relu(layer_2_addition)

    layer_3_multiplication = tf.matmul(layer_2, weights['h3'])
    layer_3_addition = tf.add(layer_3_multiplication, biases['b3'])
    layer_3 = tf.nn.relu(layer_3_addition)

    out_layer_multiplication = tf.matmul(layer_3, weights['out'])
    out_layer_addition = out_layer_multiplication + biases['out']

    return out_layer_addition
```

[0]:
```
weights = {
    'h1': tf.Variable(tf.random_normal([n_input, n_hidden_1])),
    'h2': tf.Variable(tf.random_normal([n_hidden_1, n_hidden_2])),
    'h3': tf.Variable(tf.random_normal([n_hidden_2, n_hidden_3])),
    'out': tf.Variable(tf.random_normal([n_hidden_3, n_classes])) }
biases = {
    'b1': tf.Variable(tf.random_normal([n_hidden_1])),
    'b2': tf.Variable(tf.random_normal([n_hidden_2])),
    'b3': tf.Variable(tf.random_normal([n_hidden_3])),
    'out': tf.Variable(tf.random_normal([n_classes]))
}
```

[0]:
```
def get_batch(df, i, batch_size):
  batches = []
  results = []
  texts = df.data[i * batch_size:i * batch_size + batch_size]
  categories = df.target[i * batch_size:i * batch_size + batch_size]
  for text in texts:
    layer = np.zeros(total_words, dtype=float)
    for word in text.split(" "):
      layer[word2index[word.lower()]] += 1
    batches.append(layer)
  for category in categories:
    y = np.zeros((5), dtype=float)
    if category == 0:
```

```
    y[0] = 1.
  elif category == 1:
    y[1] = 1.
  elif category == 2:
    y[2] = 1.
  elif category == 3:
    y[3] = 1.
  else:
    y[4] = 1.
  results.append(y)
 return np.array(batches), np.array(results)
```

[0]:
```
prediction = multilayer_perceptron(input_tensor, weights, biases)
```

[10]:
```
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=prediction,
 ↪labels=output_tensor))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(loss)
init = tf.global_variables_initializer()
```

```
WARNING:tensorflow:From <ipython-input-10-6a0ae33cb788>:1:
softmax_cross_entropy_with_logits (from tensorflow.python.ops.nn_ops) is
deprecated and will be removed in a future version.
Instructions for updating:

Future major versions of TensorFlow will allow gradients to flow
into the labels input on backprop by default.

See `tf.nn.softmax_cross_entropy_with_logits_v2`.
```

[11]:
```
with tf.Session() as sess:
 sess.run(init)
 for epoch in range(training_epochs):
  avg_cost = 0.
  total_batch = int(len(newsgroups_train.data)/batch_size)
  for i in range(total_batch):
   batch_x,batch_y = get_batch(newsgroups_train,i,batch_size)
   c,_ = sess.run([loss,optimizer], feed_dict={input_tensor: batch_x,output_tensor:batch_y}) #
 ↪Вычисляем среднее функции потерь
   avg_cost += c / total_batch
  print("Эпоха:", '%04d' % (epoch+1), "loss=", "{:.16f}".format(avg_cost))
  print("Обучение завершено!")


 correct_prediction = tf.equal(tf.argmax(prediction, 1), tf.argmax(output_tensor, 1)) # Расчет
 ↪точности
 accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
 total_test_data = len(newsgroups_test.target)
 batch_x_test,batch_y_test = get_batch(newsgroups_test,0,total_test_data)
 print("Точность:", accuracy.eval({input_tensor: batch_x_test, output_tensor: batch_y_test}))
```

Эпоха: 0001 loss= 81499.8946397569670808
Обучение завершено!
Эпоха: 0002 loss= 13539.8662923177089397
Обучение завершено!
Эпоха: 0003 loss= 2284.7179705301923605
Обучение завершено!
Эпоха: 0004 loss= 2288.2652531729804650
Обучение завершено!
Эпоха: 0005 loss= 1516.4307522508831880
Обучение завершено!
Эпоха: 0006 loss= 2517.8781060377755239
Обучение завершено!
Эпоха: 0007 loss= 1392.7277664078605994
Обучение завершено!
Эпоха: 0008 loss= 1741.0651477177937068
Обучение завершено!
Эпоха: 0009 loss= 5618.5449407100686585
Обучение завершено!
Эпоха: 0010 loss= 4785.6064506570492085
Обучение завершено!
Эпоха: 0011 loss= 1545.4813162485756948
Обучение завершено!
Эпоха: 0012 loss= 1678.9702631632487737
Обучение завершено!
Эпоха: 0013 loss= 93.0814882914225308
Обучение завершено!
Эпоха: 0014 loss= 0.2654427157508003
Обучение завершено!
Эпоха: 0015 loss= 0.0000000000000000
Обучение завершено!
Точность: 0.73801744

3. Контрольные вопросы

1. Какие вы знаете задачи обработки текстов, в чем они заключаются? Классификация (разбиение по темам), кластеризация (семинатический анализ текстов), построение ассоциативных правил, машинный перевод.
2. Зачем нужна предобработка текста для машинного обучения?
   Машина не умеет работать со словами, поэтому их преобразуют в эмбеддинги. Не всегда важен порядок слов, а когда-то он играет ключевую роль. Поэтому можно по-разному производить предобработку текстов. Данные могут быть зашумлены либо несогласованы.
3. Какие виды предобработки текста вы знаете?
   Стемминг, Лемматизация, векторизация, дедубликация,
4. Что такое стемминг?
   Учет словоформ. Отсекание суффиксов и прочих морфем у слова, чтобы не было множества вариаций по сути одного и того же слова.
5. Что такое 20 Newsgroups?
   Набор текстовых данных, состоящий из примерно 20 тысяч постов по 20 различным темам.

6. Чему должно равняться число входных и выходных нейронов в задаче классификации текстов? Выходных должно быть столько, сколько у нас классов.
   Входных должно быть столько, сколько уникальых слов содержится в словаре.

4. Список литературы

[1] Google. Tensorflow. 2018. Feb. url - https://www.tensorflow.org/install/install_windows.
[2] url - https://virtualenv.pypa.io/en/stable/userguide/.
[3] Microsoft. about_Execution_Policies. 2018. url - https://technet.microsoft.com/en-us/library/dd347641.aspx.
[4] Jupyter Project. Installing Jupyter. 2018. url - http://jupyter.org/install.