Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Лабораторная работа №7 по дисциплине

«Проектирование интеллектуальных систем»

ИСПОЛНИТЕЛЬ:

Чечнев А.А.

Группа ИУ5-23М

_____

"__"_____ 2020 г.

Москва 2020

1. Задание

Цель работы: Необходимо увеличить количество скрытых слоев в описанной нейронной сети до 3-ех, а количество нейронов увеличить таким образом, чтобы точность работы нейронной сети составляла не менее 75%. Темы текстов изменить в соответствии с вариантом.

В качестве тестового набора данных автор использует набор данных «Twenty newsgroups» из библиотеки scikit-learn (sklearn.datasets). Набор данных состоит из 18000 текстов на 20 различных тем. Данные разбиты на два набора — тренировочный и тестовый.

Вариант	Темы текстов
9	comp.os.ms-windows.misc, talk.politics.misc, talk.politics.mideast

2. Решение

Импорт библиотек

```
import numpy as np
import tensorflow as tf
from collections import Counter
from sklearn.datasets import fetch_20newsgroups
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()

Создание словаря

categories = ['comp.os.ms-
windows.misc', 'talk.politics.misc', 'talk.politics.mideast']
newsgroups_train = fetch_20newsgroups(subset='train', categories=categorie
s)
newsgroups_test = fetch_20newsgroups(subset='test', categories=categories)

def text2vocab(data_train, data_test):
  vocab = Counter()
  for text in data_train.data:
    for word in text.split(' '):
      vocab[word.lower()] += 1
  for text in data_test.data:
    for word in text.split(' '):
      vocab[word.lower()] += 1
  return vocab

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

```
    word2index[word.lower()] = i
  return word2index

vocab = text2vocab(newsgroups_train, newsgroups_test)
total_words = len(vocab)
word2index = get_word_2_index(vocab)
```

Функция получения батча

```
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((4), dtype=float)
    if category == 0:
      y[0] = 1.
    elif category == 1:
      y[1] = 1.
    elif category == 2:
      y[2] = 1.
    else:
      y[3] = 1.
  results.append(y)

  return np.array(batches), np.array(results)
```

Параметры обучения

```
# Параметры обучения
learning_rate = 0.01
training_epochs = 10
batch_size = 150
display_step = 1

# Network Parameters
n_hidden_1 = 2# скрытый слой
n_hidden_2 = 10 # скрытый слой
n_hidden_3 = 2 # скрытый слой
```

```
n_input = total_words # количество уникальных слов в наших текстах
n_classes = 4 # 4 класса
```

Создание модели

```
input_tensor = tf.placeholder(tf.float32,[None, n_input],name="input")
output_tensor = tf.placeholder(tf.float32,[None, n_classes],name="output")

def multilayer_perceptron(input_tensor, weights, biases):
  # скрытый слой
  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)

  # скрытый слой
  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)

  # скрытый слой
  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)

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

  return out_layer_addition

# инициализация параметров сети
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]))
}
# создание модели
prediction = multilayer_perceptron(input_tensor, weights, biases)

# Фукнция потерь
```

```
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=predi
ction, labels=output_tensor))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(l
oss)

init = tf.global_variables_initializer()
```

Обучение модели

```
import time
# Запуск
with tf.Session() as sess:
  sess.run(init)
  start_time = time.time()

  total_test_data = len(newsgroups_test.target)
  total_train_data = len(newsgroups_train.target)

  full_x_test,full_y_test = get_batch(newsgroups_test,0,total_test_data)
  full_x_train,full_y_train = get_batch(newsgroups_train,0,total_train_dat
a)

  # Цикл обучения
  for epoch in range(training_epochs):
    avg_cost = 0.
    total_batch = int(len(newsgroups_train.data)/batch_size)
    # Проход по всем батчам
    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,ou
tput_tensor:batch_y})
      # Вычисляем среднее фукнции потерь
      avg_cost += c / total_batch
    print("Эпоха:", '%04d' % (epoch+1), 'Время:%07d'%(time.time() - start_
time), "loss=", "{:.16f}".format(avg_cost))

    correct_prediction = tf.equal(tf.argmax(prediction, 1), tf.argmax(outp
ut_tensor, 1))
    # Расчет точности
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))


    #print("Точность треин:", accuracy.eval({input_tensor: batch_x, output
_tensor: batch_y}))
    print("\tТочность тест :", accuracy.eval({input_tensor: full_x_test, o
utput_tensor: full_y_test}))
    print("\tТочность треин:", accuracy.eval({input_tensor: full_x_train,
output_tensor: full_y_test}))
```

```
  print("Обучение завершено!")

# Тестирование
  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 треин:", accuracy.eval({input_tensor: batch_x, output_t
ensor: batch_y}))
  print("Точность тест :", accuracy.eval({input_tensor: batch_x_test, outp
ut_tensor: batch_y_test}))
```

Результат

```
Эпоха: 0001 Время:0000013 loss= 2.2812974810600282
        Точность тест : 0.9685185
        Точность треин: 0.9512346
Эпоха: 0002 Время:0000017 loss= 1.9397820472717284
        Точность тест : 0.97962964
        Точность треин: 0.9617284
Эпоха: 0003 Время:0000021 loss= 1.8329634666442873
        Точность тест : 0.9787037
        Точность треин: 0.9537037
Эпоха: 0004 Время:0000025 loss= 1.7523664414882660
        Точность тест : 0.9759259
        Точность треин: 0.95308644
Эпоха: 0005 Время:0000029 loss= 1.6814268022775647
        Точность тест : 0.9759259
        Точность треин: 0.95
Эпоха: 0006 Время:0000033 loss= 1.6166168749332428
        Точность тест : 0.97407407
        Точность треин: 0.94876546
Эпоха: 0007 Время:0000037 loss= 1.5579245269298552
        Точность тест : 0.97407407
        Точность треин: 0.9506173
Эпоха: 0008 Время:0000042 loss= 1.5041313827037812
        Точность тест : 0.9611111
        Точность треин: 0.94135803
Эпоха: 0009 Время:0000047 loss= 1.4561185896396638
        Точность тест : 0.9638889
        Точность треин: 0.9419753
Эпоха: 0010 Время:0000052 loss= 1.4111862778663633
        Точность тест : 0.9638889
        Точность треин: 0.9432099
Обучение завершено!
Точность тест : 0.9638889
```

Заключение: Таким образом были подобраны параметры модели, точность на тесте составила 0.96

Контрольные вопросы

1. **Какие вы знаете задачи обработки текстов, в чем они заключаются?**

- Морфологический – анализ текста связанный с контекстом, заключается в определении роли слова (часть речи, падеж, число и т.п.). Используют два метода морфологического анализа:
    - Декларативный метод заключается в записи в словарь всех грамматических форм слова(лемматизация).
    - Процедурный метод основан на записи в словарь только основ слов и выделении при собственно анализе этих основ, т.е. анализ фактически сводится к отбрасыванию суффиксов (стемминг)
- Синтаксический анализ – предназначен для определения структуры фрагментов (предложений) текста
- Семантический анализ — определение (в интеллектуальных системах) смысловых характеристик слов или словосочетаний.

2. **Зачем нужна предобработка текста для машинного обучения?**

Задачи машинного обучения представляют информацию в численном формате, поэтому необходимо векторизовать (превратить в числа) текст. Также на естественном языке присутствуют лишние, неважные для смысла, слова, которые убирают при помощи стоп-слов. Также многие словоформы могут писаться одинаково, однако иметь различный смысл. Для понимания смысла применяется лемматизация.

3. **Какие виды предобработки текста вы знаете?**

- Мешок слов – представляем набор слов в виде словаря значений
- Удаление стоп-слов
- Стемминг – вычленение основы слова
- Лемматизация – приведение слова к смысловому классу (например есть -> иметь; есть -> кушать)
- Word2Vec – перенос слов в многомерное пространство векторов, в котором схожие по смыслу слова стоят ближе друг к другу, а аболютно противоположные далеко друг от друга

4. **Что такое стемминг?**

Стеммиг – приведение слова к основе (удаление суффиксов, префиксов)

5. **Что такое 20 Newsgroups?**

20 Newsgroups – это датасет постов, распределенных по 20 различным темам.

6. **Чему должно равняться число входных и выходных нейронов в задаче классификации текстов?**
Число входных нейронов равно количеству слов в созданном словаре
Число выходных нейронов равно количеству классов

Список литературы

1. Документация по tensorflow. https://www.tensorflow.org/.

2. Глубокое обучение для NLP. https://nlp.stanford.edu/courses/NAACL2013/