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 |

1. **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=categories)

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=prediction, labels=output\_tensor))

optimizer = tf.train.AdamOptimizer(learning\_rate=learning\_rate).minimize(loss)

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\_data)

  # 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,output\_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(output\_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, output\_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"))

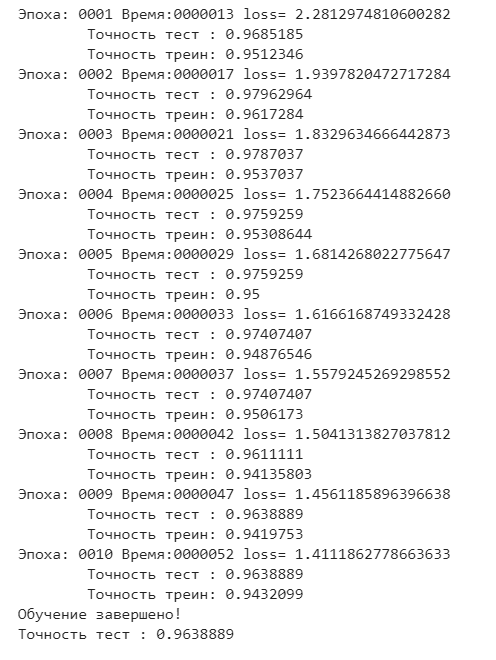
  total\_test\_data = len(newsgroups\_test.target)

  batch\_x\_test,batch\_y\_test = get\_batch(newsgroups\_test,0,total\_test\_data)

  #print("Точность треин:", accuracy.eval({input\_tensor: batch\_x, output\_tensor: batch\_y}))

  print("Точность тест :", accuracy.eval({input\_tensor: batch\_x\_test, output\_tensor: batch\_y\_test}))

**Результат**



**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.

1. **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.

1. **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

1. **Что такое стемминг?**

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

1. **Что такое 20 Newsgroups?**

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

1. **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/