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

Лабораторная работа №7 по курсу «Проектирование интеллектуальных систем»

Выполнил:
Хотин П.Ю.
ИУ5-24М

Москва, 2020 год

Задание

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 вариантом.

Вариант 10: comp.windows.x, rec.motorcycles, sci.crypt, sci.space

Выполнение задачи

Импорт библиотек

```
import bumpy as np
import tensorflow as tf
from collections import Counter
from sklearn.datasets import fetch_20newsgroups
```

Получаем тренировочную и тестовую выборку текстов по темам, которые указаны в categories

```
categories = ["comp.windows.x", "rec.motorcycles", "sci.crypt",
"sci.space"]
newsgroups_train = fetch_20newsgroups(subset='train',
categories=categories)
newsgroups_test = fetch_20newsgroups(subset='test',
categories=categories)
```

```
len(newsgroups_train.target)
```

```
2379
```

Количество слов в тестовой и тренировочной выборке

```
print('total texts in train:', len(newsgroups_train.data))
print('total texts in test:', len(newsgroups_test.data))
print('categories:', newsgroups_train.target,
newsgroups_train.target_names)
```

```
total texts in train: 2379
total texts in test: 1583
categories: [2 2 3 ... 3 1 0] ['comp.windows.x', 'rec.motorcycles',
'sci.crypt', 'sci.space']
```

Создаем словарь уникальных слов

```
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

total_words = len(vocab)
```

Создаем словарь word2index, чтобы сохранить индексы каждого слова

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

    return word2index


word2index = get_word_2_index(vocab)


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)
```

Задаем параметры обучения и параметры сети

```
# Параметры обучения
learning_rate = 0.01
training_epochs = 16
batch_size = 140
display_step = 1

# Network Parameters
n_hidden_1 = 420 # скрытый слой
n_hidden_2 = 210 # скрытый слой
n_hidden_3 = 90
n_input = total_words # количество уникальных слов в наших текстах
n_classes = 4 # 4 класса

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

Создание базовой модели

```
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=predicti
on, labels=output_tensor))
optimizer =
tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(loss)
init = tf.global_variables_initializer()
```

Обучение и результаты

```
# Запуск
with tf.Session() as sess:
    sess.run(init)
    # 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), "loss=",
"{:.16f}".format(avg_cost))

    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:", accuracy.eval({input_tensor: batch_x_test,
output_tensor: batch_y_test}))
```

```
Эпоха: 0001 loss= 18617.0064086914062500
Эпоха: 0002 loss= 4563.1136188507080078
Эпоха: 0003 loss= 3242.6922111511230469
Эпоха: 0004 loss= 4597.3344640731811523
Эпоха: 0005 loss= 3107.6046485900878906
Эпоха: 0006 loss= 577.2595361173152924
Эпоха: 0007 loss= 238.7371116876602173
Эпоха: 0008 loss= 9.4245774447917938
Эпоха: 0009 loss= 5.2540618411730975
Эпоха: 0010 loss= 0.0000000011175857
Эпоха: 0011 loss= 0.0000000000000000
Эпоха: 0012 loss= 0.0000000000000000
Эпоха: 0013 loss= 0.0000000000000000
Эпоха: 0014 loss= 0.0000000000000000
Эпоха: 0015 loss= 0.0000000000000000
Эпоха: 0016 loss= 0.0000000000000000
Обучение завершено!
Точность: 0.75868607
```

Ответы на вопросы

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

Классификация(выявление отношения к группам), кластеризация(выделение из групп текстов одинаковой тематики), построение ассоциативных правил(закономерности между словами), машинный перевод.

2. Зачем нужна предобработка текста для машинного обучения?

Предобработка текста переводит текст на естественном языке в формат удобный для дальнейшей работы. Ключевую роль играют такие факторы как, порядок слов, наличие словоформы.

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

Перевод всех букв в тексте в нижний или верхний регистры;

Удаление пробельных символов (whitespaces);

Удаление стоп слов;

Стемминг.

4. **Что такое стемминг?**

Количество корректных словоформ, значения которых схожи, но написания отличаются суффиксами, приставками, окончаниями и прочим, очень велико, что усложняет создание словарей и дальнейшую обработку. Стемминг позволяет привести слово к его основной форме.

5. **Что такое 20 Newsgroups?**

Набор из 18000 статей, разделенных между собой на 20 тематик, каждая из которых относится либо к тренировочной, либо к тестовой выборкам.

6. **Чему должно равняться число входных и выходных нейронов в задаче классификации текстов?**

Число входных нейронов должно равняться числу уникальных слов в тексте, а число выходных нейронов должно соответствовать количество классов.

Список литературы

[1] Google. Tensorflow. 2018. Apr. url - https://www.tensorflow.org/api_docs/python/tf/train/Saver.

[2] Google. TensorBoard. 2018. Apr. url - https://www.tensorflow.org/programmers_guide/summaries_and_tensorboard.