SINGULARIS LAB

TensorFlow MNIST example

Катаев Александр
Ведущий инженер-программист, к.т.н.
Алексеев Алексей
Инженер-программист
Singularis Lab, Ltd.

Проверка tf

```
import tensorflow as tf
hello = tf.constant('hello')
sess = tf.InteractiveSession()
print(sess.run(hello))
```

Start TF session

```
import tensorflow as tf
sess = tf.InteractiveSession()
```

Import MNIST data

```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets('MNIST_data', one_hot=True)
```

Simple linear model

- placeholders хранят данные (входы/выходы)

```
x = tf.placeholder(tf.float32, shape=[None, 784])
y_ = tf.placeholder(tf.float32, shape=[None, 10])
```

- variables находятся в процессе обучения

```
W = tf.Variable(tf.zeros([784,10]))
b = tf.Variable(tf.zeros([10]))
sess.run(tf.initialize_all_variables())
```

Prediction and Loss function

- Вычисление выхода (prediction)

```
y = tf.nn.softmax(tf.matmul(x,W) + b)
```

- Вычисление целевой функции (Loss)

```
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_
                * tf.log(y), reduction_indices=[1]))
```

Train the model

- Инициализация оптимизатора (обычный GD)

```
train_step = tf.train.GradientDescentOptimizer(0.5)
                                    .minimize(cross_entropy)
```

- Обучение модели

```
for i in range(1000):
    batch = mnist.train.next_batch(100)
    train_step.run(feed_dict={x: batch[0], y_: batch[1]})
```

Testing

- Массив флагов (верное/неверное решение)

  correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))

- Оценка точности

  accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

CNN

- Инициализация весов

```
def weight_variable(shape):
  initial = tf.truncated_normal(shape, stddev=0.1)
  return tf.Variable(initial)
```

- Инициализация смещений

```
def bias_variable(shape):
  initial = tf.constant(0.1, shape=shape)
  return tf.Variable(initial)
```

Свертка и пулинг

- Функция сверточного слоя, шаг 1, паддинг – под размер

```
def conv2d(x, W):
    return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')
```

- Функция пулинга, шаг 2, паддинг – под размер

```
def max_pool_2x2(x):
    return tf.nn.max_pool(x, ksize=[1, 2, 2, 1],
                          strides=[1, 2, 2, 1],padding='SAME')
```

Конструирование первого сверточного слоя

- 32 ядра свертки 5x5, входной 1 канал

```
W_conv1 = weight_variable([5, 5, 1, 32])
b_conv1 = bias_variable([32])
```

- Преобразуем вход к 4D тензору

```
x_image = tf.reshape(x, [-1,28,28,1])
```

- Вычисление слоя свертка+relu+pool

```
h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
h_pool1 = max_pool_2x2(h_conv1)
```

второй слой

- 2 слой: 64 ядра, 5x5

```
W_conv2 = weight_variable([5, 5, 32, 64])
b_conv2 = bias_variable([64])
h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)
```

Полносвязанный слой

- 1024 нейрона

```
W_fc1 = weight_variable([7 * 7 * 64, 1024])
b_fc1 = bias_variable([1024])
h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
```

Dropout-слой

```
keep_prob = tf.placeholder(tf.float32)
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)
```

Выходной слой

- Полносвязанный слой, 10 нейронов

```
W_fc2 = weight_variable([1024, 10])
b_fc2 = bias_variable([10])
y_conv=tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)
```

Инициализация обучения

```
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_
                                * tf.log(y_conv), reduction_indices=[1]))
```

- Используем ADAM - вариант стохастического GD

```
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
sess.run(tf.initialize_all_variables())
```

Обучение

```
for i in range(20000):
  batch = mnist.train.next_batch(50)
  if i%100 == 0:                    # вывод промежуточных результатов
    train_accuracy = accuracy.eval(feed_dict={
        x:batch[0], y_: batch[1], keep_prob: 1.0})
    print("step %d, training accuracy %g"%(i, train_accuracy))
  # шаг обучения
  train_step.run(feed_dict={x: batch[0], y_: batch[1], keep_prob: 0.5})

# вывод окончательного результата
print("test accuracy %g"%accuracy.eval(feed_dict={
    x: mnist.test.images, y_: mnist.test.labels, keep_prob: 1.0}))
```

Сохранение и загрузка обученной сессии

```
saver = tf.train.Saver()

# сохранение
saver.save(sess,  ‘model.cpkt’)

# загрузка
saver.restore(sess, ‘model.cpkt’)
```

Спасибо за внимание

Александр Катаев

- alexander.kataev@singularis-lab.com

Алексей Алексеев

- aleksey.alekseev@singularis-lab.com

https://www.singularis-lab.com/

https://www.linkedin.com/company/singularis-lab-llc

http://habrahabr.ru/company/singularis

http://vk.com/singularis_lab