Softmax, MLP, CNN 三种方法识别手写数字MNIST——《TensorFlow 实战》读书笔记

2017-04-01 | 人工智能 | 0

不要代码写多了就变得那么没有人情味了

0x00 Intro

1. 读入MNIST数据库

执行 mnist = input_data.read_data_sets("MNIST_data/", one_hot=True) 后，会检查 MNIST_data/ 文件夹下有没有数据库文件，如果没有会自动下载。这一步如果执行比较慢，可以用迅雷手动下载下面四个文件，保存到MNIST_data目录（不需要解压）

- train-images-idx3-ubyte.gz: training set images (9912422 bytes)
- train-labels-idx1-ubyte.gz: training set labels (28881 bytes)

- t10k-images-idx3-ubyte.gz: test set images (1648877 bytes)
- t10k-labels-idx1-ubyte.gz: test set labels (4542 bytes)

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

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
```

2. 初始化 Tensorflow

Tensorflow 运行时默认会把GPU显存一次性占满，添加 config.gpu_options.allow_growth = True 使其可以动态分配显存

```
import tensorflow as tf
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
sess = tf.InteractiveSession(config = config)
```

0x01 Softmax

只用 Softmax Regression 进行分类，正确率约为92%

1. 定义变量

Softmax Regression 的公式可以写成：

$$y = softmax(Wx + b)$$

其中， x 为输入数据（手写数字图片），不限条数的 784 维的 Float32 型数据；

W 为 784×10（特征维数×图片种类） 的 Variable 向量；

b 是bias（偏置）

y 为Softmax分类后得出的结果

loss function为 cross_entropy ，定义如下：

$$H'_y(y) = -\sum_i y'_i log(y_i)$$

而 reduce_mean 为对每个batch求均值（reduction_indices=[1]的意思请看后面的附录， 在新版的Tensorflow tutorial中，这部分稍有区别

）

```
x = tf.placeholder(tf.float32,[None,784])
W = tf.Variable(tf.zeros([784,10]))
b = tf.Variable(tf.zeros([10]))

y = tf.nn.softmax(tf.matmul(x,W) + b)
y_ = tf.placeholder(tf.float32,[None,10])

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

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

2. 训练

每次训练抽取100个样本作为 mini-batch ，并传给 placeholder(x,y_) 进行训练

```
tf.global_variables_initializer().run()
for i in range(1000):
    batch_xs, batch_ys = mnist.train.next_batch(1000)
    train_step.run({x: batch_xs, y_: batch_ys})
```

3. 计算正确率

accuracy.eval 与 train_step.run 的区别官方给出了如下解释

> Note: the Tensor class will be replaced by Output in the future. Currently these two are aliases for each other.

```
correct_prediction = tf.equal(tf.arg_max(y,1),tf.arg_max(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction,tf.float32))
print(accuracy.eval({x:mnist.test.images,y_:mnist.test.labels}))

summary_writer = tf.summary.FileWriter('Softmax', sess.graph)
```

0.9211

0x02 MLP

使用多层感知机（MLP）进行分类，准确率约为98%

1. 定义变量

```
in_units = 784
h1_units = 300
W1 = tf.Variable(tf.truncated_normal([in_units,h1_units],stddev=0.1))
b1 = tf.Variable(tf.zeros([h1_units]))
W2 = tf.Variable(tf.zeros([h1_units, 10]))
b2 = tf.Variable(tf.zeros([10]))

x = tf.placeholder(tf.float32,[None,in_units])
keep_prob = tf.placeholder(tf.float32)

hidden1 = tf.nn.relu(tf.matmul(x,W1)+b1)
hidden1_drop = tf.nn.dropout(hidden1,keep_prob)
y = tf.nn.softmax(tf.matmul(hidden1_drop,W2)+b2)

y_ = tf.placeholder(tf.float32,[None,10])
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y),reduction_indices=[1]))
train_step = tf.train.GradientDescentOptimizer(0.3).minimize(cross_entropy)
```

2. 训练

```
tf.global_variables_initializer().run()
for i in range(5000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    train_step.run({x: batch_xs, y_: batch_ys, keep_prob: 0.75})
```

3. 计算正确率

```
correct_prediction = tf.equal(tf.argmax(y,1),tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction,tf.float32))

print(accuracy.eval({x:mnist.test.images,y_:mnist.test.labels, keep_prob: 1.0}))
```

```
summary_writer = tf.summary.FileWriter('MLP', sess.graph)
```

0.9787

0x03 CNN

1. 定义变量

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

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

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

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

```
x = tf.placeholder(tf.float32,[None,in_units])
y_ = tf.placeholder(tf.float32,[None,10])
x_image = tf.reshape(x, [-1,28,28,1])

W_conv1 = weight_variable([5,5,1,32])
b_conv1 = bias_variable([32])
h_conv1 = tf.nn.relu(conv2d(x_image,W_conv1) + b_conv1)
h_pool1 = max_pool_2x2(h_conv1)

```

```
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)

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)

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

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)

cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y_conv), reduction_indices=[1]))
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
```

2. 训练

```
correct_prediction = tf.equal(tf.arg_max(y_conv,1),tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction,tf.float32))

tf.global_variables_initializer().run()
for i in range(20000):
    batch = mnist.train.next_batch(100)
    if i%100 == 0:
        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))
    train_step.run(feed_dict={x:batch[0],y_:batch[1],keep_prob:0.5})
```

```
step 0, training accuracy 0.07
step 100, training accuracy 0.84
step 200, training accuracy 0.9
step 300, training accuracy 0.88
step 400, training accuracy 0.95
step 500, training accuracy 0.98
step 600, training accuracy 0.96
step 700, training accuracy 0.94
step 800, training accuracy 1
step 900, training accuracy 0.98
step 1000, training accuracy 0.99
step 1100, training accuracy 0.96
step 1200, training accuracy 0.99
step 1300, training accuracy 0.99
step 1400, training accuracy 0.98
step 1500, training accuracy 1
step 1600, training accuracy 0.98
step 1700, training accuracy 0.97
step 1800, training accuracy 0.99
step 1900, training accuracy 0.98
step 2000, training accuracy 0.98
step 2100, training accuracy 0.98
step 2200, training accuracy 0.99
step 2300, training accuracy 0.98
step 2400, training accuracy 0.99
step 2500, training accuracy 0.97
step 2600, training accuracy 0.97
step 2700, training accuracy 0.97
step 2800, training accuracy 0.99
step 2900, training accuracy 1
step 3000, training accuracy 1
step 3100, training accuracy 1
step 3200, training accuracy 0.98
step 3300, training accuracy 0.99
step 3400, training accuracy 0.98
```

step 3500, training accuracy 1
step 3600, training accuracy 1
step 3700, training accuracy 0.98
step 3800, training accuracy 1
step 3900, training accuracy 0.98
step 4000, training accuracy 1
step 4100, training accuracy 0.99
step 4200, training accuracy 0.99
step 4300, training accuracy 0.99
step 4400, training accuracy 1
step 4500, training accuracy 0.99
step 4600, training accuracy 1
step 4700, training accuracy 1
step 4800, training accuracy 1
step 4900, training accuracy 0.98
step 5000, training accuracy 0.99
step 5100, training accuracy 1
step 5200, training accuracy 0.98
step 5300, training accuracy 1
step 5400, training accuracy 1
step 5500, training accuracy 1
step 5600, training accuracy 1
step 5700, training accuracy 1
step 5800, training accuracy 1
step 5900, training accuracy 0.99
step 6000, training accuracy 1
step 6100, training accuracy 1
step 6200, training accuracy 1
step 6300, training accuracy 0.99
step 6400, training accuracy 1
step 6500, training accuracy 0.99
step 6600, training accuracy 1
step 6700, training accuracy 1
step 6800, training accuracy 1
step 6900, training accuracy 0.97

...

step 18500, training accuracy 1
step 18600, training accuracy 1
step 18700, training accuracy 1
step 18800, training accuracy 1
step 18900, training accuracy 1
step 19000, training accuracy 1
step 19100, training accuracy 1
step 19200, training accuracy 1
step 19300, training accuracy 1
step 19400, training accuracy 1
step 19500, training accuracy 1
step 19600, training accuracy 1
step 19700, training accuracy 1
step 19800, training accuracy 1
step 19900, training accuracy 1

```
print("test accuracy %g"%accuracy.eval(feed_dict={x: mnist.test.images, y_: mnist.test.labels,keep_prob:1.0})) # 这
summary_writer = tf.summary.FileWriter('CNN', sess.graph)
```

test accuracy 0.9931

> 回家过节去，剩下的下周再写>_< 更新链接：http://s1nh.org/post/Tensorflow-MNIST/ （鄙视转载还不署名的某网站）

0x04 附录

1. 关于reduction_indices

```python
x=[[2,2,2],[2,2,2]]
y0=tf.reduce_sum(x,reduction_indices=[0])
y1=tf.reduce_sum(x,reduction_indices=[1])

print("x  = ", x)
with tf.Session() as sess:
    print("y0 = ", sess.run(y0), "\t(x在第0维度相加)")
    print("y1 = ", sess.run(y1), "\t(x在第1维度相加)")
```

```
x  =  [[2, 2, 2], [2, 2, 2]]
y0 =  [4 4 4]     (x在第0维度相加)
y1 =  [6 6]     (x在第1维度相加)
```

MNIST   # Tensorflow

HackRF 入门 -- GPS欺骗、GSM嗅探

Jetson TX-2 入门 -- 全部你应该知道的

加入評論

HC

Enter Email