#一、数据准备

实验数据使用MNIST数据集。

MNIST 数据集已经是一个被”嚼烂”了的数据集, 很多教程都会对它”下手”, 几乎成为一个 “典范”。

在很多tensorflow教程中，用下面这一句下载mnist数据集：

```

mnist = input\_data.read\_data\_sets('MNIST\_data', one\_hot=True)

```

但实际运行时根本无法通过网络下载，解决方案就是手工下载数据，然后直接导入使用。

下载地址：http://yann.lecun.com/exdb/mnist/

4个文件，注意下载后不需要解压。

如果把上述下载的文件放在与运行的.py文件同一个目录下，那么导入数据的代码是这样的：

```

mnist = input\_data.read\_data\_sets('./', one\_hot=True)

```

#二、代码

```

import tensorflow as tf

from tensorflow.examples.tutorials.mnist import input\_data

# number 1 to 10 data

mnist = input\_data.read\_data\_sets('./', one\_hot=True)

def compute\_accuracy(v\_xs, v\_ys):

global prediction

y\_pre = sess.run(prediction, feed\_dict={xs: v\_xs, keep\_prob: 1})

correct\_prediction = tf.equal(tf.argmax(y\_pre,1), tf.argmax(v\_ys,1))

accuracy = tf.reduce\_mean(tf.cast(correct\_prediction, tf.float32))

result = sess.run(accuracy, feed\_dict={xs: v\_xs, ys: v\_ys, keep\_prob: 1})

return result

# 产生随机变量，符合 normal 分布

# 传递 shape 就可以返回weight和bias的变量

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)

# 定义2维的 convolutional 图层

def conv2d(x, W):

# stride [1, x\_movement, y\_movement, 1]

# Must have strides[0] = strides[3] = 1

# strides 就是跨多大步抽取信息

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

# 定义 pooling 图层

def max\_pool\_2x2(x):

# stride [1, x\_movement, y\_movement, 1]

# 用pooling对付跨步大丢失信息问题

return tf.nn.max\_pool(x, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

# define placeholder for inputs to network

xs = tf.placeholder(tf.float32, [None, 784]) # 784＝28x28

ys = tf.placeholder(tf.float32, [None, 10])

keep\_prob = tf.placeholder(tf.float32)

x\_image = tf.reshape(xs, [-1, 28, 28, 1]) # 最后一个1表示数据是黑白的

# print(x\_image.shape) # [n\_samples, 28,28,1]

## 1. conv1 layer ##

# 把x\_image的厚度1加厚变成了32

W\_conv1 = weight\_variable([5, 5, 1, 32]) # patch 5x5, in size 1, out size 32

b\_conv1 = bias\_variable([32])

# 构建第一个convolutional层，外面再加一个非线性化的处理relu

h\_conv1 = tf.nn.relu(conv2d(x\_image, W\_conv1) + b\_conv1) # output size 28x28x32

# 经过pooling后，长宽缩小为14x14

h\_pool1 = max\_pool\_2x2(h\_conv1) # output size 14x14x32

## 2. conv2 layer ##

# 把厚度32加厚变成了64

W\_conv2 = weight\_variable([5,5, 32, 64]) # patch 5x5, in size 32, out size 64

b\_conv2 = bias\_variable([64])

# 构建第二个convolutional层

h\_conv2 = tf.nn.relu(conv2d(h\_pool1, W\_conv2) + b\_conv2) # output size 14x14x64

# 经过pooling后，长宽缩小为7x7

h\_pool2 = max\_pool\_2x2(h\_conv2) # output size 7x7x64

## 3. func1 layer ##

# 飞的更高变成1024

W\_fc1 = weight\_variable([7\*7\*64, 1024])

b\_fc1 = bias\_variable([1024])

# [n\_samples, 7, 7, 64] ->> [n\_samples, 7\*7\*64]

# 把pooling后的结果变平

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)

h\_fc1\_drop = tf.nn.dropout(h\_fc1, keep\_prob)

## 4. func2 layer ##

# 最后一层，输入1024，输出size 10，用 softmax 计算概率进行分类的处理

W\_fc2 = weight\_variable([1024, 10])

b\_fc2 = bias\_variable([10])

prediction = tf.nn.softmax(tf.matmul(h\_fc1\_drop, W\_fc2) + b\_fc2)

# the error between prediction and real data

cross\_entropy = tf.reduce\_mean(-tf.reduce\_sum(ys \* tf.log(prediction),

reduction\_indices=[1])) # loss

train\_step = tf.train.AdamOptimizer(1e-4).minimize(cross\_entropy)

sess = tf.Session()

# important step

sess.run(tf.global\_variables\_initializer())

for i in range(1000):

batch\_xs, batch\_ys = mnist.train.next\_batch(100)

sess.run(train\_step, feed\_dict={xs: batch\_xs, ys: batch\_ys, keep\_prob: 0.5})

if i % 50 == 0:

print(compute\_accuracy(

mnist.test.images, mnist.test.labels))

```

#三、Github代码下载

[下载]( https://github.com/zhenghaishu/MachineLearning/tree/master/CNN)

#四、参考

<http://v.youku.com/v_show/id_XMTYyMTUyMjc0OA==.html?spm=a2hzp.8253869.0.0>

<https://github.com/MorvanZhou/tutorials/tree/master/tensorflowTUT/tf18_CNN3>

<https://www.jianshu.com/p/e2f62043d02b>

<https://blog.csdn.net/i8088/article/details/79126150>