数字媒体技术作业3

标签： 树媒技术基础作业

郭柱明 15331094

实验准备

语言

> python

模块以及版本

实验步骤

> 从mnist数据集中加载训练和测试的数据，第一次运行时需要多一点时间下载数据集合的压缩包

```
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

使用线性回归模型进行识别，下面声明各个模型中的变量，x用来存放图像，维度为[None, 784]，None意味着图像数量不限定，784表示每张图表示成784个点的一维向量，线性回归的参数W，维度为[784,
10]，线性回归的参数b，维度为[10]，让W和x矩阵相乘，加上b，然后计算softmax，softmax(x)=normalize(exp(x))，y_为用来存放正确的y结果，使用交叉熵cross_entropy，以在后面使用梯度下降法根据交叉熵确定最佳的W和b，train_step，使用梯度下降法最小化交叉熵

```
#x用来存放图像，维度为[None， 784]，None意味着图像数量不限定，784表示每张图表示成784个点的一维向量
x = tf.placeholder(tf.float32, [None, 784])
#线性回归的参数W，维度为[784， 10]
W = tf.Variable(tf.zeros([784,10]))
#线性回归的参数b，维度为[10]
b = tf.Variable(tf.zeros([10]))
#让W和x矩阵相乘，加上b，然后计算softmax，softmax(x)=normalize(exp(x))
y = tf.nn.softmax(tf.matmul(x,W) + b)
#y_为用来存放正确的y结果
y_ = tf.placeholder("float", [None,10])
#使用交叉熵cross_entropy，以在后面使用梯度下降法根据交叉熵确定最佳的W和b
cross_entropy = -tf.reduce_sum(y_*tf.log(y))
#使用梯度下降法最小化交叉熵
train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entropy)
```

初始化模型中的各种变量

```
#初始化所有变量
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)
```

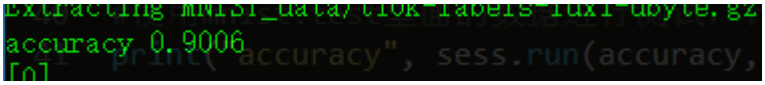
使用mnist.train中的图像矩阵和标签训练线性回归模型，确定W和b的最佳值

```
#训练1000次
for i in range(1000):
    #每次拿100张图像和标签来训练，batch_xs为训练的图像，batch_ys为对应的标签
    batch_xs, batch_ys = mnist.train.next_batch(100)
    #根据最小梯度下降法降低交叉熵
    sess.run(train_step, {x: batch_xs, y_: batch_ys})
```

> 训练完之后，W和b的值被确定下来，使用mnist.test中的图像矩阵和标签进行测试，计算准确率

```
#correct_prediction表示下面的预测是否正确
correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
#accuracy表示准确率
accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
#使用mnist.test里面的数据进行测试，计算准确率
print("accuracy", sess.run(accuracy, feed_dict = {x: mnist.test.images, y
_: mnist.test.labels}))
```

输出准确率为：0.9006



> 拿test_digits文件夹中的10张图像进行测试，输出预测结果

```
#拿文件夹test_digits里面的0-9图片进行测试
for i in range(10):
    #读取图像为灰度图
    i = Image.open("test_digits/" + str(i) + ".png").convert("L")
    #缩小为28*28
    i = i.resize((28, 28), Image.ANTIALIAS)
    #获取灰度矩阵
    arr = numpy.array(i, dtype = "float32")
    #图片二值化
    ret, arr = cv2.threshold(255 - arr, 90, 255, cv2.THRESH_BINARY)
    #像素值归一化和展平为一维向量
    arr = (arr / 255).flatten()
    #进行测试，输出预测结果
    narr = numpy.zeros((1, 784))
    narr[0] = arr
    y_r = sess.run(y, {x: narr})
    print(sess.run(tf.argmax(y_r, 1)))
```

输入测试图像：


0.png


1.png


2.png


3.png


4.png


5.png


6.png


7.png


8.png


9.png

输出结果正确：

```
G:\大三上\数图基础\tensorflow_test>python handwriting.py
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
accuracy 0.9006
[0]
[1]
[2]
[3]
[4]
[5]
[6]
[7]
[8]
[9]
```

拿test_digits文件夹中的10张图像进行测试，输出预测结果

```
#拿文件夹test_digits里面的0-9图片进行测试
for i in range(10):
    #读取图像为灰度图
```