1、简介

任务时间：30min ~ 60min

本实验介绍浅层神经网络在 TensorFlow 上的实现，并使用模型处理 MNIST 数据集。

MNIST数据集介绍

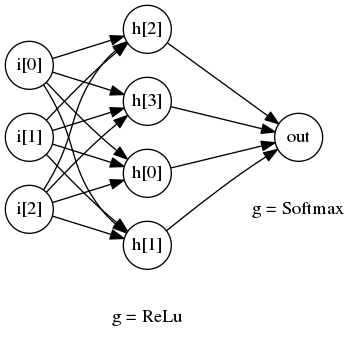
[MNIST](http://yann.lecun.com/exdb/mnist/)是一个手写阿拉伯数字的数据集。

其中包含有60000个已经标注了的训练集，还有10000个用于测试的测试集。

本次实验的任务就是通过手写数字的图片，识别出具体写的是0-9之中的哪个数字。

理论知识回顾

一个典型的浅层神经网络结构如下：



1. 上图所示的是一个只有一层隐藏层的浅层神经网络
2. 我们有3个输入层节点，分别对应i[1] i[2] i[3]
3. 隐藏层有4个节点，分别对应h[0] h[1] h[2] h[3]，对应的激活函数为ReLu函数
4. 对于典型的二分类任务，我们只需要1个输出节点，就是out节点，对应的激活函数是[softmax](http://ufldl.stanford.edu/wiki/index.php/Softmax%E5%9B%9E%E5%BD%92" \o "null" \t "_blank)函数

**激活函数定义（activation function）：**

<formula>

softmax(x^i) = \left[

\begin{matrix}

p(y^i=1|x^i,\theta) \\

p(y^i=2|x^i,\theta) \\

\cdots \\

p(y^i=n|x^i,\theta)

\end{matrix}

\right] = \frac{1}{\sum\_{j=1}^{k}e^{\theta\_j^Tx^i}} \left[

\begin{matrix}

e^{\theta\_1^Tx^i} \\

e^{\theta\_2^Tx^i} \\

\cdots \\

e^{\theta\_k^Tx^i}

\end{matrix}

\right]

</formula>

<formula>

ReLu(x) = max(0, x)

</formula>

模型设计

1. MNIST数据一共有784个输入，所以我们需要一个有784个节点的输入层。
2. 对于中间层，我们设置为1000个节点，使用的激活函数为ReLu
3. MNIST数据使用One-Hot格式输出，有0-9 10个label，分别对应是否为数字0-9，所以我们在输出层有10个节点，由于0-9的概率是互斥的，我们使用 [Softmax](http://ufldl.stanford.edu/wiki/index.php/Softmax%E5%9B%9E%E5%BD%92" \o "null" \t "_blank) 函数作为该层的激活函数

训练模型

任务时间：30min ~ 60min

数据准备

首先我们需要先下载MNIST的数据集。使用以下的命令进行下载：

wget https://devlab-1251520893.cos.ap-guangzhou.myqcloud.com/t10k-images-idx3-ubyte.gz

wget https://devlab-1251520893.cos.ap-guangzhou.myqcloud.com/t10k-labels-idx1-ubyte.gz

wget https://devlab-1251520893.cos.ap-guangzhou.myqcloud.com/train-images-idx3-ubyte.gz

wget https://devlab-1251520893.cos.ap-guangzhou.myqcloud.com/train-labels-idx1-ubyte.gz

创建代码

现在您可以在 /home/ubuntu 目录下创建源文件 shallow\_neural\_networks.py，内容可参考：

**示例代码：/home/ubuntu/shallow\_neural\_networks.py**

import numpy as np

import tensorflow as tf

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

def add\_layer(inputs, in\_size, out\_size, activation\_function=None):

W = tf.Variable(tf.random\_normal([in\_size, out\_size]))

b = tf.Variable(tf.zeros([1, out\_size]) + 0.01)

Z = tf.matmul(inputs, W) + b

if activation\_function is None:

outputs = Z

else:

outputs = activation\_function(Z)

return outputs

if \_\_name\_\_ == "\_\_main\_\_":

MNIST = input\_data.read\_data\_sets("./", one\_hot=True)

learning\_rate = 0.05

batch\_size = 128

n\_epochs = 10

X = tf.placeholder(tf.float32, [batch\_size, 784])

Y = tf.placeholder(tf.float32, [batch\_size, 10])

l1 = add\_layer(X, 784, 1000, activation\_function=tf.nn.relu)

prediction = add\_layer(l1, 1000, 10, activation\_function=None)

entropy = tf.nn.softmax\_cross\_entropy\_with\_logits(labels=Y, logits=prediction)

loss = tf.reduce\_mean(entropy)

optimizer = tf.train.GradientDescentOptimizer(learning\_rate).minimize(loss)

init = tf.initialize\_all\_variables()

with tf.Session() as sess:

sess.run(init)

n\_batches = int(MNIST.train.num\_examples/batch\_size)

for i in range(n\_epochs):

for j in range(n\_batches):

X\_batch, Y\_batch = MNIST.train.next\_batch(batch\_size)

\_, loss\_ = sess.run([optimizer, loss], feed\_dict={X: X\_batch, Y: Y\_batch})

if j == 0:

print "Loss of epochs[{0}] batch[{1}]: {2}".format(i, j, loss\_)

# test the model

n\_batches = int(MNIST.test.num\_examples/batch\_size)

total\_correct\_preds = 0

for i in range(n\_batches):

X\_batch, Y\_batch = MNIST.test.next\_batch(batch\_size)

preds = sess.run(prediction, feed\_dict={X: X\_batch, Y: Y\_batch})

correct\_preds = tf.equal(tf.argmax(preds, 1), tf.argmax(Y\_batch, 1))

accuracy = tf.reduce\_sum(tf.cast(correct\_preds, tf.float32))

total\_correct\_preds += sess.run(accuracy)

print "Accuracy {0}".format(total\_correct\_preds/MNIST.test.num\_examples)

代码讲解

**add\_layer 函数**

允许用户指定上一层的输出节点的个数作为input\_size, 本层的节点个数作为output\_size, 并指定激活函数activation\_function 可以看到我们调用的时候位神经网络添加了隐藏层1和输出层

**示例代码：/home/ubuntu/shallow\_neural\_networks.py**

l1 = add\_layer(X, 784, 1000, activation\_function=tf.nn.relu) # 添加隐藏层1

prediction = add\_layer(l1, 1000, 10, activation\_function=None) # 添加输出层

entropy = tf.nn.softmax\_cross\_entropy\_with\_logits(labels=Y, logits=prediction)

loss = tf.reduce\_mean(entropy)

执行代码

python shallow\_neural\_networks.py

运行过程中，如果出现网络错误，请重试。

运行输出：

Loss of epochs[0] batch[0]: 219.555664062

Loss of epochs[1] batch[0]: 4.43757390976

Loss of epochs[2] batch[0]: 6.34549808502

Loss of epochs[3] batch[0]: 4.51894617081

Loss of epochs[4] batch[0]: 1.93666791916

Loss of epochs[5] batch[0]: 1.16164898872

Loss of epochs[6] batch[0]: 2.4195971489

Loss of epochs[7] batch[0]: 0.164100989699

Loss of epochs[8] batch[0]: 2.35461592674

Loss of epochs[9] batch[0]: 1.77732157707

Accuracy 0.9434

可以看到经过10轮的训练，准确度大约在94%左右

完成实验

任务时间：1min