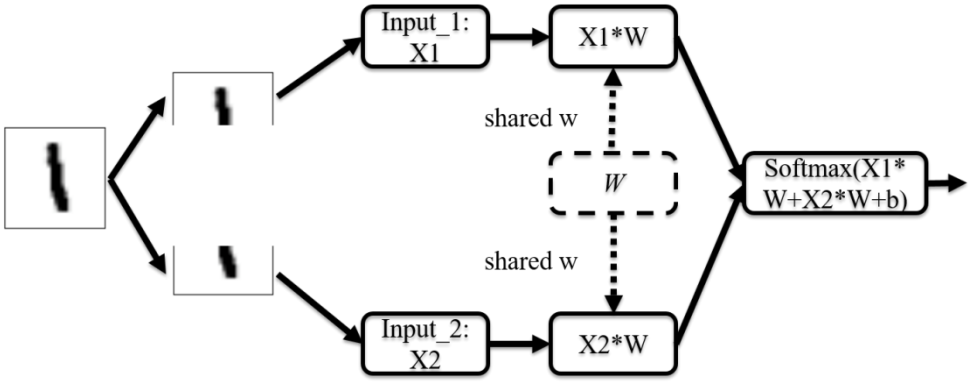
深度学习方法与实践第三次作业

姓名：杨玉雷
学号：18023040

1.基础作业

设计变量共享网络进行 MNIST 分类：
网络结构如图所示：



其将图片样本分为上下两半 X1,X2；分别送入 input1,input2。后续的两个路径的线性加权模块 X_W=X*W 共享一个变量 name='w'
整个分类模型可描述为 softmax(X_W(X1)+X_W(X2)+b)
模型及流程可以参考我们课件 part1 上最后的那个一层全连接分 MNIST 的代码例子
要求：1. 线性加权模块 X_W 需定义为一个函数，在此函数中创建并共享变量 W name='w'
函数 X_W(X)只有一个输入参数 X
W 必须在 X_W(X)中用 get_variable 定义

```
def X_W(X)
...
      return tf.matmul(X,W)
```

预期结果：

```
Step 0,Training Accuracy 0.2463
Step 200,Training Accuracy 0.8706
Step 400,Training Accuracy 0.8306
Step 600,Training Accuracy 0.8856
Step 800,Training Accuracy 0.8801
Step 1000,Training Accuracy 0.871
Step 1200,Training Accuracy 0.8734
Step 1400,Training Accuracy 0.8853
Step 1600,Training Accuracy 0.8646
Step 1800,Training Accuracy 0.8907
[accuracy,loss]: [0.8613, 5286.7505]
```

训练精度大概最后在 0.85 左右
提交：1. 文档（训练过程截图，训练、测试精度等）。2. 代码

1.1 实验过程

（1）splitX(x)：切分训练数据为上下两部分。

```
#数据集由（1,784）切分成两份，每份为（1,392）
def splitX(x):
    x1,x2=tf.split(x,num_or_size_splits=2,axis=1)
    return x1,x2
```

（2）X_W(X)：在函数中 tf.get_variable 方式定义 W 并返回 W 和 X 的乘积。

```
def X_W(X):
    W=tf.get_variable(shape=[392,10],name='weight')
    print("W_name:",W.name)
    return tf.matmul(X, W)
```

（3）compute_y(x)：在函数中使用 tf.variable_scope("share_weight")定义作用域，并通过 scope.reuse_variables()允许变量 W 共享，最后返回经过 softmax 计算后的预测值 y。

```
def compute_y(x):
    X1,X2=splitX(x)
    with tf.variable_scope("share_weight") as scope:
        out1 = X_W(X1)
        #允许变量 W 共享
        scope.reuse_variables()
        out2 = X_W(X2)
        y = tf.nn.softmax(out1 + out2 + b)  # 预测值
        return y
```

（4）完整代码

```
import tensorflow as tf
from tensorflow.contrib.learn.python.learn.datasets.mnist import read_data_sets

#数据集由（1,784）切分成两份，每份为（1,392）
def splitX(x):
    x1,x2=tf.split(x,num_or_size_splits=2,axis=1)
    return x1,x2

def X_W(X):
    W=tf.get_variable(shape=[392,10],name='weight')
    print("W_name:",W.name)
```

```
    return tf.matmul(X, W)

def compute_y(x):
    X1,X2=splitX(x)
    with tf.variable_scope("share_weight") as scope:
        out1 = X_W(X1)
        #允许变量共享
        scope.reuse_variables()
        out2 = X_W(X2)
        y = tf.nn.softmax(out1 + out2 + b)  # 预测值
        return y

mnist = read_data_sets("data/",one_hot=True)
x = tf.placeholder(dtype='float',shape =[None ,784])
b = tf.Variable(tf.zeros ([10]))

y=compute_y(x)
y_ = tf.placeholder(dtype='float',shape =[None ,10]) #真实值
#计算交叉熵
cross_entropy = -tf.reduce_sum(y_ * tf.log(y))
train_step    =    tf.train.    GradientDescentOptimizer    (learning_rate
=0.01).minimize(cross_entropy)

init = tf. global_variables_initializer ()
sess = tf.Session ()
sess.run(init)

step = 500
loss_list = []
for i in range(step):
    #从训练集里一次提取100张图片数据来训练
    batch_xs ,batch_ys = mnist.train.next_batch (100) #shape: (100, 784) (100,
10)
    steps=i*100
    _,loss=      sess.run([train_step      ,cross_entropy      ],feed_dict
={x:batch_xs,y_:batch_ys })
    loss_list.append(loss)
    #预测并打印精度
    correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, 'float'))
    print('step:',steps,'[accuracy    ,loss    ]:',    sess.run([accuracy,
cross_entropy], feed_dict={x: mnist.test.images, y_: mnist.
                                        test.labels}))
```

1.2 实验结果

（1）变量共享

将数据集 x 分成两部分 X1 和 X2 后两次调用 X_W(X1)和 X_W(X2)，中间使用 scope.reuse_variables()允许变量共享，最后输出两次变量名相同，即实现变量共享。

```
Extracting data/train-images-idx3-ubyte.gz
Extracting data/train-labels-idx1-ubyte.gz
Extracting data/t10k-images-idx3-ubyte.gz
Extracting data/t10k-labels-idx1-ubyte.gz
W_name: share_weight/weight:0
W_name: share_weight/weight:0
2019-05-16 11:05:15.770401: W tensorflow/core/platforr
2019-05-16 11:05:15.770417: W tensorflow/core/platforr
2019-05-16 11:05:15.770420: W tensorflow/core/platforr
2019-05-16 11:05:15.770423: W tensorflow/core/platforr
2019-05-16 11:05:15.770425: W tensorflow/core/platforr
step: 0 [accuracy ,loss ]: [0.20110001, 30025.75]
step: 100 [accuracy ,loss ]: [0.45210001, 50663.906]
step: 200 [accuracy ,loss ]: [0.37490001, 53038.602]
step: 300 [accuracy ,loss ]: [0.3382, 55523.062]
step: 400 [accuracy ,loss ]: [0.3608, 51820.727]
```

1.1 变量共享结果

（2）训练过程

训练时进行 500 个 step，每个 step 选取 100 张图片进行训练，并输出每个 step 的精确度和 loss。训练精度最后大致为 0.85 左右。

```
step: 46900 [accuracy ,loss ]: [0.87440002, 4651.1011]
step: 47000 [accuracy ,loss ]: [0.87129998, 4863.9062]
step: 47100 [accuracy ,loss ]: [0.87889999, 4525.2476]
step: 47200 [accuracy ,loss ]: [0.875, 4917.0537]
step: 47300 [accuracy ,loss ]: [0.88190001, 4648.0303]
step: 47400 [accuracy ,loss ]: [0.83829999, 6018.1328]
step: 47500 [accuracy ,loss ]: [0.83179998, 6251.5283]
step: 47600 [accuracy ,loss ]: [0.84689999, 5851.2676]
step: 47700 [accuracy ,loss ]: [0.87220001, 4886.0107]
step: 47800 [accuracy ,loss ]: [0.87510002, 4646.0488]
step: 47900 [accuracy ,loss ]: [0.86540002, 4850.3745]
step: 48000 [accuracy ,loss ]: [0.86750001, 4971.3955]
step: 48100 [accuracy ,loss ]: [0.80699998, 7147.8008]
step: 48200 [accuracy ,loss ]: [0.81, 8360.9697]
step: 48300 [accuracy ,loss ]: [0.7784, 8061.0928]
step: 48400 [accuracy ,loss ]: [0.85390002, 5482.7314]
step: 48500 [accuracy ,loss ]: [0.88020003, 4685.0059]
step: 48600 [accuracy ,loss ]: [0.87800002, 4517.5605]
step: 48700 [accuracy ,loss ]: [0.85519999, 5271.1338]
step: 48800 [accuracy ,loss ]: [0.87959999, 4758.6348]
step: 48900 [accuracy ,loss ]: [0.85949999, 5178.542]
step: 49000 [accuracy ,loss ]: [0.889, 4281.1162]
step: 49100 [accuracy ,loss ]: [0.87050003, 5093.4819]
step: 49200 [accuracy ,loss ]: [0.86150002, 5374.7383]
step: 49300 [accuracy ,loss ]: [0.83569998, 6174.8638]
step: 49400 [accuracy ,loss ]: [0.79710001, 8662.8271]
step: 49500 [accuracy ,loss ]: [0.88120002, 4745.708]
step: 49600 [accuracy ,loss ]: [0.88630003, 4432.1709]
step: 49700 [accuracy ,loss ]: [0.88440001, 4384.8198]
step: 49800 [accuracy ,loss ]: [0.84060001, 5569.1094]
step: 49900 [accuracy ,loss ]: [0.85250002, 5208.5303]

Process finished with exit code 0
```

1.2 实验精度和 loss 损失