Tensorflow 学习文档

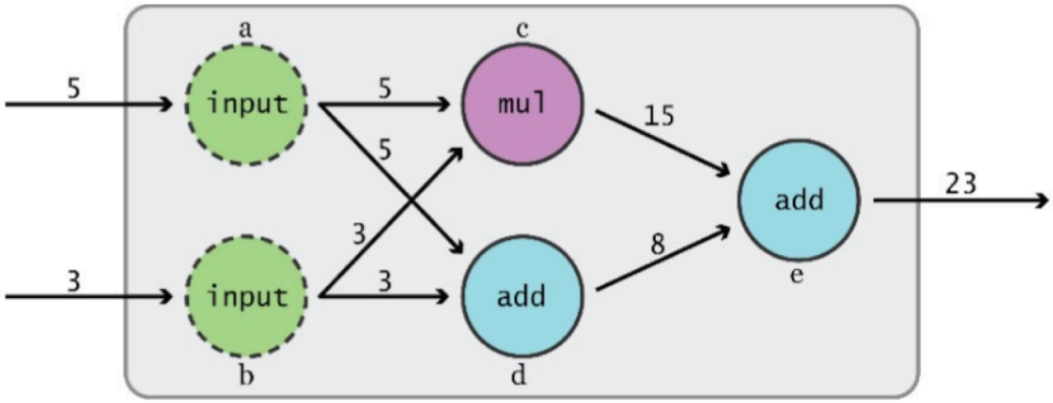
1、张量（tensor）定义：A tensor is something that transforms like a tensor! 一个量，在不同的参考系下按照某种特定的法则进行变换，就是张量。

2、tensorflow 设计思想：用户自己设计模型流程，Tensorflow 依赖于一个高效的 C++后端来进行计算。与后端的这个连接叫做 session。一般而言，使用 TensorFlow 程序的流程是先创建一个图，然后在 session 中启动它。

3、使用 TensorFlow，必须明确几个点:

- 使用图 (graph) 来表示计算任务.
- 在被称之为 会话 (Session) 的上下文 (context) 中执行图.
- 使用 tensor 表示数据。
- 通过 变量 (Variable) 维护状态。
- 使用 feed 和 fetch 可以为任意的操作(arbitrary operation) 赋值或者从其中获取数据。

4、Tensorflow 将计算与执行分离开来：
阶段 1：创建一张图，定义好图中的就算
阶段 2：使用 session（会话）去执行图中的计算



5、tensorflow api：

- Class env
- Class env wrapper
- Class random access file
- Class seesion：

InteractiveSession 类：你在运行图的时候，插入一些计算图，这些计算图是由某些操作

(operations)构成的。这对于工作在交互式环境中的人们来说非常便利，比如使用 IPython。如果你没有使用 InteractiveSession，那么你需要在启动 session 之前构建整个计算图，然后启动该计算图。

参考：http://www.tensorfly.cn/tfdoc/api_docs/cc/ClassSession.html

- Class status
- Class tensor

A Neural Network Example

神经网络的例子，数据集为 MNIST 数据集。
1. 加载数据：

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

one_hot=True 表示对 label 进行 one-hot 编码，比如标签 4 可以表示为[0, 0, 0, 0, 1, 0, 0, 0, 0, 0]。这是神经网络输出层要求的格式。

2. 定义超参数和 placeholder

```
# 超参数
learning_rate = 0.5
epochs = 10
batch_size = 100

# placeholder
# 输入图片为 28 x 28 像素 = 784
x = tf.placeholder(tf.float32, [None, 784])
# 输出为 0-9 的 one-hot 编码
y = tf.placeholder(tf.float32, [None, 10])
```

再次强调，[None, 784]中的 None 表示任意值，特别对应 tensor 数目。

3. 定义参数 w 和 b

```
# hidden layer => w, b
W1 = tf.Variable(tf.random_normal([784, 300], stddev=0.03), name='W1')
b1 = tf.Variable(tf.random_normal([300]), name='b1')
# output layer => w, b
W2 = tf.Variable(tf.random_normal([300, 10], stddev=0.03), name='W2')
b2 = tf.Variable(tf.random_normal([10]), name='b2')
```

在这里，要了解全连接层的两个参数 w 和 b 都是需要随机初始化的，tf.random_normal()生成正态分布的随机数。

4. 构造隐层网络

```
# hidden layer
hidden_out = tf.add(tf.matmul(x, W1), b1)
hidden_out = tf.nn.relu(hidden_out)
```

上面代码对应于公式：
z=wx+b
h=relu(z)

5. 构造输出（预测值）

```
# 计算输出
y_ = tf.nn.softmax(tf.add(tf.matmul(hidden_out, W2), b2))
```

对于单标签多分类任务，输出层的激活函数都是 tf.nn.softmax()。更多关于 softmax 的知识见维基百科。

6. BP 部分—定义 loss
损失为交叉熵，公式为

$$J=-1m\sum i=1m\sum j=1nyijlog(y(i)j)+(1-y(i)jlog(1-y(i)j)$$

公式分为两步：

对 n 个标签计算交叉熵
对 m 个样本取平均

```
y_clipped = tf.clip_by_value(y_, 1e-10, 0.9999999)
cross_entropy = -tf.reduce_mean(tf.reduce_sum(y * tf.log(y_clipped) + (1 - y) * tf.log(1 - y_clipped), axis=1))
```

7. BP 部分—定义优化算法

```
# 创建优化器，确定优化目标
optimizer = tf.train.GradientDescentOptimizer(learning_rate=learning_rate).minimizer(cross_entropy)
```

8. 定义初始化 operation 和准确率 node

```
# init operator
init_op = tf.global_variables_initializer()

# 创建准确率节点
```

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

correct_predicion 会返回一个 m×1
的 tensor，tensor 的值为 True/False 表示是否正确预测。

Setting up the trianing

9. 开始训练

```
# 创建 session
with tf.Session() as sess:
    # 变量初始化
    sess.run(init)
    total_batch = int(len(mnist.train.labels) / batch_size)
    for epoch in range(epochs):
        avg_cost = 0
        for i in range(total_batch):
            batch_x, batch_y = mnist.train.next_batch(batch_size=batch_size)
            _, c = sess.run([optimizer, cross_entropy], feed_dict={x: batch_x, y: batch_y})
            avg_cost += c / total_batch
        print("Epoch:", (epoch + 1), "cost = ", "{:.3f}".format(avg_cost))
    print(sess.run(accuracy, feed_dict={x: mnist.test.images, y: mnist.test.labels}))
```