如何使用Tensorflow搭建神经网络

刘璐

2018.10.10

Tensorflow是什么

- Python
- 深度学习工具
- 不需要手写复杂的运算代码
- 不需要手动逐层求导

使用图表示计算任务

- Tensorflow程序被组织成一个构建阶段和一个执行阶段：
  - 构建阶段：将算法执行步骤描述成一个图，图中的每个节点表示一个操作
  - 执行阶段：输入实际数据，执行图中操作
- 数据以Tensor的形式在图中流动

使用Tensor表示数据

- 在Tensorflow程序中，所有的数据都使用Tensor表示。
- Tensor有三个基本属性：
  - dtype：表示Tensor的数据类型
  - rank：表示Tensor的维度
  - shape：表示Tensor的形状

dtype

- dtype表示tensor的数据类型：
  - tf.float32, tf.float64, …
  - tf.int32, tf.int64, …
  - tf.bool
  - tf.string
  - …

Rank & Shape

- Tensor的rank是Tensor的维数的数量描述：
- 不需要显式定义rank，定义shape时rank自然被确定

Rank	Shape（示例）	意义	实例
0	[]	标量	t = 1926
1	[4]	向量	o =[1, 9, 2, 6]
2	[2, 2]	矩阵	a = [[1, 9],[2, 6]]
3	[2,2,2]	立方体	d = [[[1, 9], [2, 6]],[0, 8], [1,7]]]
更高	[1,9,2,6]	。。。	。。。

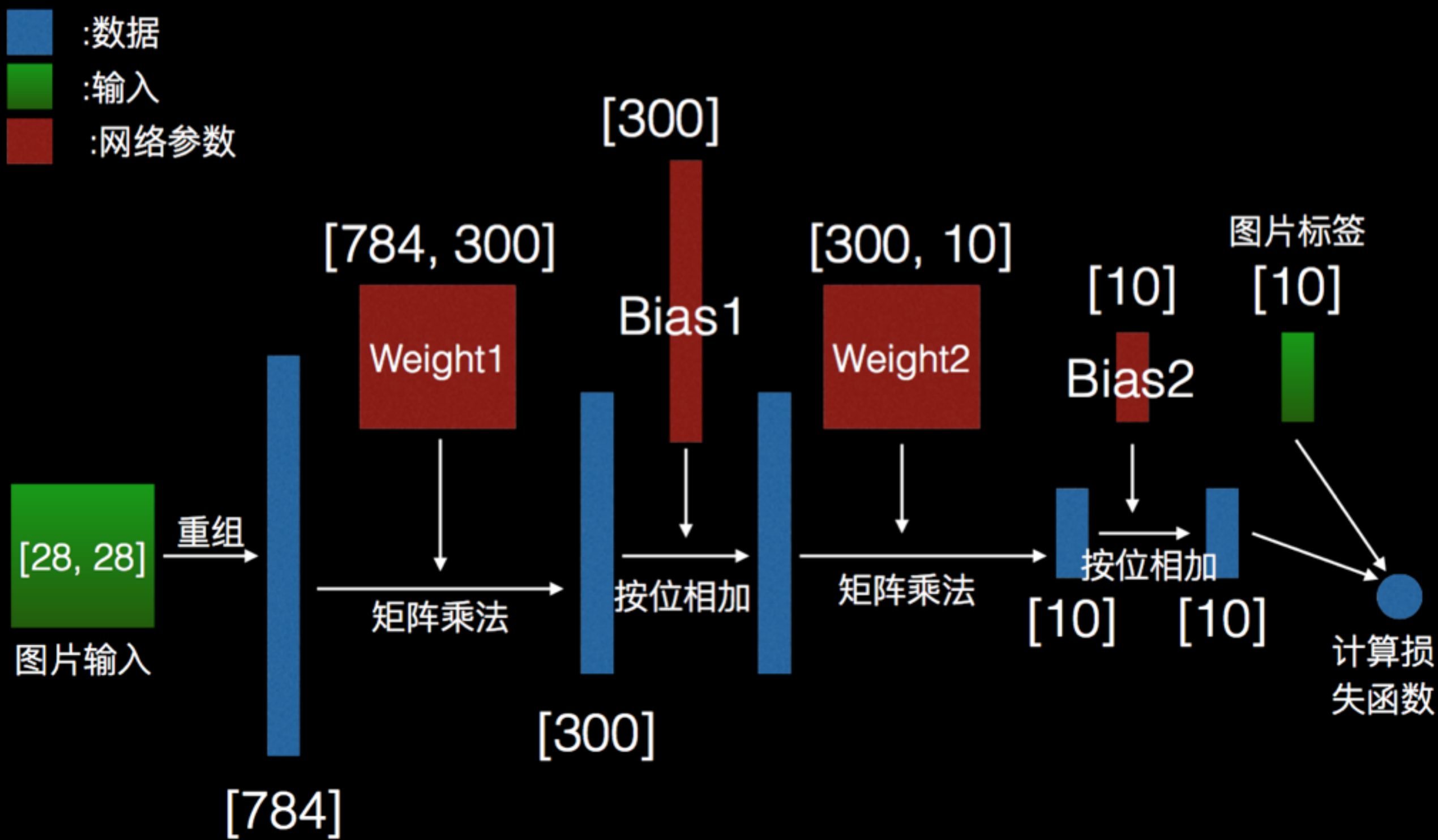
开始构建图

- 计算图是包含若干个操作节点的有向图，可以大致划分为以下三部分：
  - 源节点（数据输入）： placeholder, constant
  - 操作节点（网络结构）：各种网络运算
  - 结束节点（优化目标）： 各种优化方法

Tensorflow常用操作
以三层神经网络为例

- 需求：实现一个神经网络用于手写数字识别，输入图片大小为28 * 28，输出共有10个分类（0-9）

设计网络结构


:数据
:输入
:网络参数
[300]
[784, 300]
[300, 10]
图片标签
[10]
[10]
Bias1
Weight1
Weight2
Bias2
[28, 28]
重组
矩阵乘法
按位相加
矩阵乘法
按位相加
[10]
[10]
图片输入
[784]
[300]
计算损失函数

Step1:图片输入 & 图片标签

- tf.placeholder: 作为计算图的初始节点，实际运行的时候需要先被赋值。

- x_ = tf.placeholder(dtype=tf.float32, shape=[None, 28, 28])

- y = tf.placeholder(dtype=tf.float32, shape=[None, 10])

- 此处None表示不确定的batch_size，在运行图的时候会根据输入的情况调整

Step2: 重组

- tf.reshape：自由变换Tensor的形状
  - t = [1, 2, 3, 4, 5, 6], t的shape是[6]
  - t1 = tf.reshape(t, [2, 3]), t1 = [[1,2,3],[4,5,6]]
  - -1可以被自动计算:
    - t2 = tf.reshape(t, [-1, 3]), 效果与t1相同

- 此处需要将28 * 28的输入矩阵重组为长度为784的向量：
  - x = tf.reshape(x_, [-1, 784])
  - 现在x是一个shape为[None, 784]的Tensor

Step3: 定义模型参数

- 模型参数是随着训练进程可变的，因此需要使用tf.Variable定义

```
W = tf.Variable(tf.zeros([784, 300]))
b = tf.Variable(tf.zeros([300]))
```

Step4: 矩阵运算

- 有了输入层和W1, b1之后，可以使用矩阵乘法和按位加法计算中间层的值
  - 矩阵乘法: tf.matmul
  - 按位加法： +
- _h = tf.matmul(x, w) + b

Step5: 激活函数

- 激活函数直接使用tf.nn.sigmoid等函数即可
  - h = tf.nn.sigmoid(_h)

Step6: 计算误差

- 第二层网络的实现与第一层相同，不妨设已经得到了第二层网络的输出_o
- 分类器激活函数softmax:
  - o = tf.nn.softmax(_o)
- 计算损失函数
  - cross_entropy = -tf.reduce_sum(y * tf.log(o))

Step7: 优化损失函数

- train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entropy)
- 至此，已经完成构建图的步骤

Step8: 输入数据进行训练

- Tensorflow使用Session来启动图
  - sess = tf.InteractiveSession()
  - tf.global_variables_initializer().run()
  - 使用feed_dict来为placeholder赋值，使用sess.run运行

```
sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
```

CNN相关API

- tf.nn.conv2d

https://www.tensorflow.org/api_docs/python/tf/nn/conv2d

- tf.nn.max_pool

https://www.tensorflow.org/api_docs/python/tf/nn/max_pool

Tricks

- 对weights进行L2正则化
- 数据增强制造更多样本
- Dropout避免过拟合
- ……

谢谢！