

## **1Tensor**

在 TensorFlow 中，数据不是以整数、浮点数或者字符串形式存储的。这些值被封装在一个叫做 tensor 的对象中。在 hello\_constant = tf.constant('Hello World!') 代码中，hello\_constant 是一个 0 维度的字符串 tensor，tensor 还有很多不同大小：

*# A is a 0-dimensional int32 tensor*

A = tf.constant(1234) *# B is a 1-dimensional int32 tensor*

B = tf.constant([123,456,789])

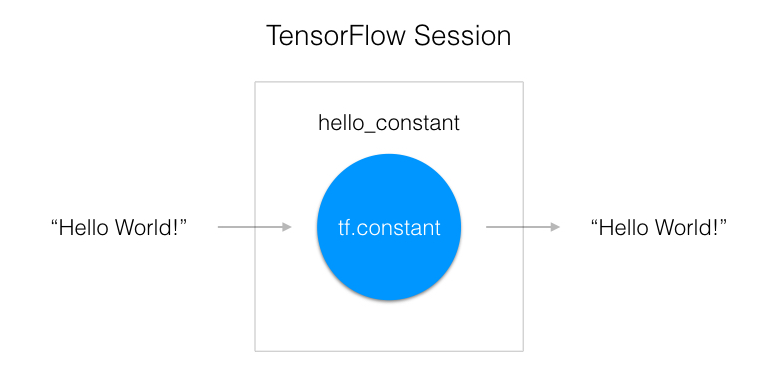
*# C is a 2-dimensional int32 tensor*

C = tf.constant([ [123,456,789], [222,333,444] ])

**[tf.constant()](https://www.tensorflow.org/api_docs/python/tf/constant" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 是你在本课中即将使用的多个 TensorFlow 运算之一。**[tf.constant()](https://www.tensorflow.org/api_docs/python/tf/constant" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 返回的 tensor 是一个常量 tensor，因为这个 tensor 的值不会变。

## **2Session**

TensorFlow 的 api 构建在 computational graph 的概念上，它是一种对数学运算过程进行可视化的方法（在 MiniFlow 这节课中学过）。让我们把你刚才运行的 TensorFlow 代码变成一个图：



如上图所示，一个 "TensorFlow Session" 是用来运行图的环境。这个 session 负责分配 GPU(s) 和／或 CPU(s)，包括远程计算机的运算。让我们看看如何使用它：

**with** tf.Session() **as** sess:

output = sess.run(hello\_constant)

代码已经从之前的一行中创建了 tensor hello\_constant。接下来是在 session 里对 tensor 求值。

这段代码用 **[tf.Session](https://www.tensorflow.org/api_docs/python/tf/Session" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 创建了一个 sess 的 session 实例。然后 **[sess.run()](https://www.tensorflow.org/api_docs/python/tf/Session" \l "run" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 函数对 tensor 求值，并返回结果。

# **3输入**

在上一小节中，你向 session 传入一个 tensor 并返回结果。如果你想使用一个非常量（non-constant）该怎么办？这就是 **[tf.placeholder()](https://www.tensorflow.org/api_docs/python/tf/placeholder" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 和 feed\_dict 派上用场的时候了。这一节将向你讲解向 TensorFlow 传输数据的基础知识。

## **tf.placeholder()**

很遗憾，你不能把数据集赋值给 x 再将它传给 TensorFlow。因为之后你会想要你的 TensorFlow 模型对不同的数据集采用不同的参数。你需要的是 **[tf.placeholder()](https://www.tensorflow.org/api_docs/python/tf/placeholder" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)**！

数据经过 **[tf.session.run()](https://www.tensorflow.org/api_docs/python/tf/Session" \l "run" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 函数得到的值，由 **[tf.placeholder()](https://www.tensorflow.org/api_docs/python/tf/placeholder" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 返回成一个 tensor，这样你可以在 session 运行之前，设置输入。

## **Session 的 feed\_dict**

x = tf.placeholder(tf.string)

**with** tf.Session() **as** sess:

output = sess.run(x, feed\_dict={x: 'Hello World'})

用 **[tf.session.run()](https://www.tensorflow.org/api_docs/python/tf/Session" \l "run" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 里的 feed\_dict 参数设置占位 tensor。上面的例子显示 tensor x被设置成字符串 "Hello, world"。如下所示，也可以用 feed\_dict 设置多个 tensor。

x = tf.placeholder(tf.string)

y = tf.placeholder(tf.int32)

z = tf.placeholder(tf.float32)

**with** tf.Session() **as** sess:

output = sess.run(x, feed\_dict={x: 'Test String', y: 123, z: 45.67})

****注意：****

如果传入 feed\_dict 的数据与 tensor 类型不符，就无法被正确处理，你会得到 “ValueError: invalid literal for...”。

# **4TensorFlow 数学**

获取输入很棒，但是现在你需要使用它。你将使用每个人都懂的基础数学运算，加、减、乘、除，来处理 tensor。（更多数学函数请查看**[文档](https://tensorflow.google.cn/api_docs/python/math_ops/" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)**）。

## **加法**

x = tf.add(5, 2) *# 7*

从加法开始，**[tf.add()](https://www.tensorflow.org/api_guides/python/math_ops" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 函数如你所想，它传入两个数字、两个 tensor、或数字和 tensor 各一个，以 tensor 的形式返回它们的和。

## **减法和乘法**

这是减法和乘法的例子：

x = tf.subtract(10, 4) *# 6*

y = tf.multiply(2, 5) *# 10*

x tensor 求值结果是 6，因为 10 - 4 = 6。y tensor 求值结果是 10，因为 2 \* 5 = 10。是不是很简单！

## **类型转换**

为了让特定运算能运行，有时会对类型进行转换。例如，你尝试下列代码，会报错：

tf.subtract(tf.constant(2.0),tf.constant(1)) # Fails with ValueError: Tensor conversion requested dtype float32 for Tensor with dtype int32:

这是因为常量 1 是整数，但是常量 2.0 是浮点数，subtract 需要它们的类型匹配。

在这种情况下，你可以确保数据都是同一类型，或者强制转换一个值为另一个类型。这里，我们可以把 2.0 转换成整数再相减，这样就能得出正确的结果：

tf.subtract(tf.cast(tf.constant(2.0), tf.int32), tf.constant(1)) # 1

# **5TensorFlow 里的线性函数**

神经网络中最常见的运算，就是计算输入、权重和偏差的线性组合。回忆一下，我们可以把线性运算的输出写成：

**[IMG_256](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/baf36422-c1b4-4005-960f-63a550e635d4)**

这里 \mathbf{W}**W** 是连接两层的权重矩阵。输出 \mathbf{y}**y**，输入 \mathbf{x}**x**，偏差 \mathbf{b}**b** 全部都是向量。

## **TensorFlow 里的权重和偏差**

训练神经网络的目的是更新权重和偏差来更好地预测目标。为了使用权重和偏差，你需要一个能修改的 Tensor。这就排除了 **[tf.placeholder()](https://www.tensorflow.org/api_docs/python/tf/placeholder" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 和 **[tf.constant()](https://www.tensorflow.org/api_docs/python/tf/constant" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)**，因为它们的 Tensor 不能改变。这里就需要 **[tf.Variable](https://www.tensorflow.org/api_docs/python/tf/Variable" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 了。

### **tf.Variable()**

x = tf.Variable(5)

**[tf.Variable](https://www.tensorflow.org/api_docs/python/tf/Variable" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 类创建一个 tensor，其初始值可以被改变，就像普通的 Python 变量一样。该 tensor 把它的状态存在 session 里，所以你必须手动初始化它的状态。你将使用 **[tf.global\_variables\_initializer()](https://www.tensorflow.org/api_docs/python/tf/global_variables_initializer" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 函数来初始化所有可变 tensor。

##### **初始化**

init = tf.global\_variables\_initializer()**with** tf.Session() **as** sess:

sess.run(init)

**[tf.global\_variables\_initializer()](https://www.tensorflow.org/api_docs/python/tf/global_variables_initializer" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 会返回一个操作，它会从 graph 中初始化所有的 TensorFlow 变量。你可以通过 session 来调用这个操作来初始化所有上面的变量。用 **[tf.Variable](https://www.tensorflow.org/api_docs/python/tf/Variable" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 类可以让我们改变权重和偏差，但还是要选择一个初始值。

从正态分布中取随机数来初始化权重是个好习惯。随机化权重可以避免模型每次训练时候卡在同一个地方。在下节学习梯度下降的时候，你将了解更多相关内容。

类似地，从正态分布中选择权重可以避免任意一个权重与其他权重相比有压倒性的特性。你可以用 **[tf.truncated\_normal()](https://www.tensorflow.org/api_docs/python/tf/truncated_normal" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 函数从一个正态分布中生成随机数。

### **tf.truncated\_normal()**

n\_features = 120

n\_labels = 5

weights = tf.Variable(tf.truncated\_normal((n\_features, n\_labels)))

**[tf.truncated\_normal()](https://www.tensorflow.org/api_docs/python/tf/truncated_normal" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 返回一个 tensor，它的随机值取自一个正态分布，并且它们的取值会在这个正态分布平均值的两个标准差之内。

因为权重已经被随机化来帮助模型不被卡住，你不需要再把偏差随机化了。让我们简单地把偏差设为 0。

### **tf.zeros()**

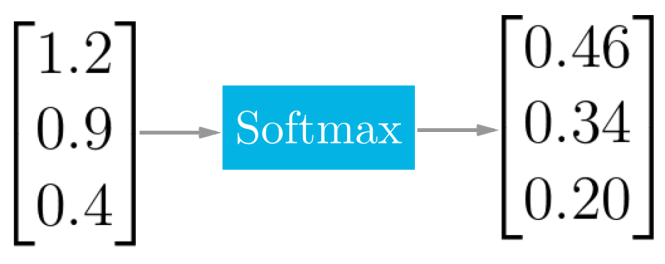
n\_labels = 5

bias = tf.Variable(tf.zeros(n\_labels))

**[tf.zeros()](https://www.tensorflow.org/api_docs/python/tf/zeros" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 函数返回一个都是 0 的 tensor。

# **6TensorFlow Softmax**

Softmax 函数可以把它的输入，通常被称为 ****logits**** 或者 ****logit scores****，处理成 0 到 1 之间，并且能够把输出归一化到和为 1。这意味着 **softmax 函数与分类的概率分布等价**。它是一个网络预测多分类问题的最佳输出激活函数。

**[](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/bb19eec1-3fb2-413c-8043-9e514c4e3396)**

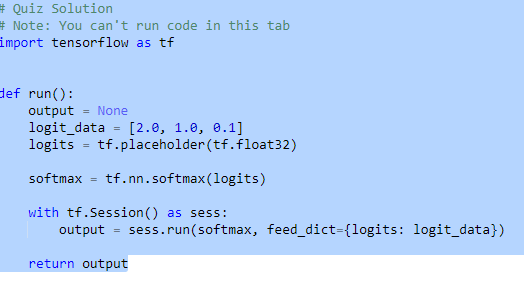
**[softmax 函数的实际应用示例](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/bb19eec1-3fb2-413c-8043-9e514c4e3396)**

当我们用 TensorFlow 来构建一个神经网络时，相应地，它有一个计算 softmax 的函数。

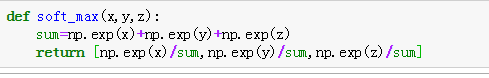
x = tf.nn.softmax([2.0, 1.0, 0.2])

就是这么简单，**[tf.nn.softmax()](https://www.tensorflow.org/api_docs/python/tf/nn/softmax" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 直接为你实现了 softmax 函数，它输入 logits，返回 softmax 激活函数。

以下两者等效

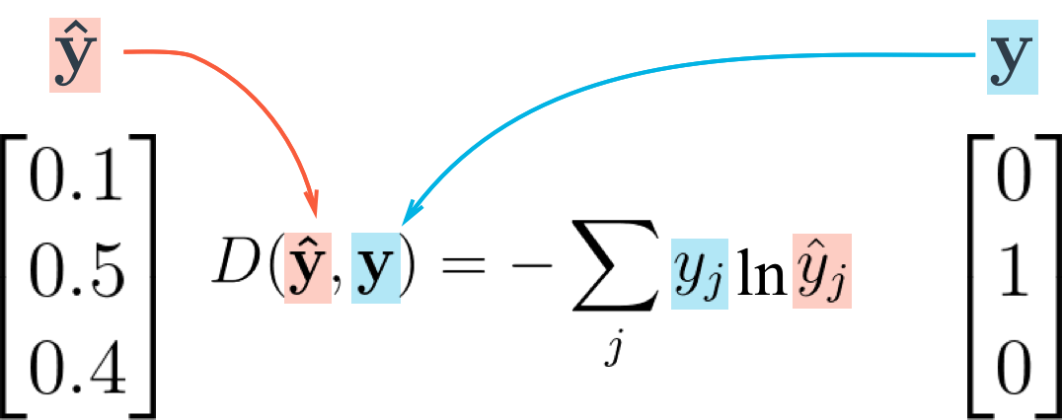


即对数几率分数[2,1,0.1]，通过sortmax函数变成概率分布，后期通过这个概率分布来计算loss函数,计算loss函数的方法为交叉熵或者二次代价函数等等



# **7TensorFlow 中的交叉熵（Cross Entropy）**

与 softmax 一样，TensorFlow 也有一个函数可以方便地帮我们实现交叉熵。

**[](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b6f63b94-90e3-463b-8f40-5a5bf3bfcfb9)**

**[Cross entropy loss function 交叉熵损失函数](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b6f63b94-90e3-463b-8f40-5a5bf3bfcfb9)**

让我们把你从视频当中学到的知识，在 TensorFlow 中来创建一个交叉熵函数。创建一个交叉熵函数，你需要用到这两个新的函数：

**交叉熵：即对sortmax获取其他激活函数计算出来的概率分布来计算loss函数，便于后期反向传播误差**

* **[tf.reduce\_sum()](https://www.tensorflow.org/api_docs/python/tf/reduce_sum" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)**
* **[tf.log()](https://www.tensorflow.org/api_docs/python/tf/log" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)**

## **Reduce Sum**

x = tf.reduce\_sum([1, 2, 3, 4, 5]) *# 15*

**[tf.reduce\_sum()](https://www.tensorflow.org/api_docs/python/tf/reduce_sum" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 函数输入一个序列，返回它们的和

## **Natural Log**

x = tf.log(100) *# 4.60517*

**[tf.log()](https://www.tensorflow.org/api_docs/python/tf/log" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 所做跟你所想的一样，它返回所输入值的自然对数。

## **8Mini-batching**

在这一节，你将了解什么是 mini-batching，以及如何在 TensorFlow 里应用它。

Mini-batching 是一个一次训练数据集的一小部分，而不是整个训练集的技术。它可以使内存较小、不能同时训练整个数据集的电脑也可以训练模型。

Mini-batching 从运算角度来说是低效的，因为你不能在所有样本中计算 loss。但是这点小代价也比根本不能运行模型要划算。

它跟随机梯度下降（SGD）结合在一起用也很有帮助。方法是在每一代训练之前，对数据进行随机混洗，然后创建 mini-batches，对每一个 mini-batch，用梯度下降训练网络权重。因为这些 batches 是随机的，你其实是在对每个 batch 做随机梯度下降（SGD）。

让我们看看你的机器能否训练出 MNIST 数据集的权重和偏置项。

**from** tensorflow.examples.tutorials.mnist **import** input\_data**import** tensorflow **as** tf

n\_input = 784 *# MNIST data input (img shape: 28\*28)*

n\_classes = 10 *# MNIST total classes (0-9 digits)*

*# Import MNIST data*

mnist = input\_data.read\_data\_sets('/datasets/ud730/mnist', one\_hot=**True**)

*# The features are already scaled and the data is shuffled*

train\_features = mnist.train.images

test\_features = mnist.test.images

train\_labels = mnist.train.labels.astype(np.float32)

test\_labels = mnist.test.labels.astype(np.float32)

*# Weights & bias*

weights = tf.Variable(tf.random\_normal([n\_input, n\_classes]))

bias = tf.Variable(tf.random\_normal([n\_classes]))

### **问题1**

计算 train\_features, train\_labels, weights, 和 bias 分别占用了多少字节（byte）的内存。可以忽略头部空间，只需要计算实际需要多少内存来存储数据。

你也可以看**[这里](https://en.wikipedia.org/wiki/Single-precision_floating-point_format" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)**了解一个 float32 占用多少内存。

*train\_features Shape: (55000, 784) Type: float32*

*train\_labels Shape: (55000, 10) Type: float32*

*weights Shape: (784, 10) Type: float32*

*bias Shape: (10,) Type: float32*

输入、权重和偏置项总共的内存空间需求是 174MB，并不是太多。你可以在 CPU 和 GPU 上训练整个数据集。

但将来你要用到的数据集可能是以 G 来衡量，甚至更多。你可以买更多的内存，但是会很贵。例如一个 12GB 显存容量的 Titan X GPU 会超过 1000 美金。所以，为了在你自己机器上运行大模型，你需要学会用 mini-batching。

让我们看下如何在 TensorFlow 下实现 mini-batching

## **TensorFlow Mini-batching**

要使用 mini-batching，你首先要把你的数据集分成 batch。

不幸的是，有时候不可能把数据完全分割成相同数量的 batch。例如有 1000 个数据点，你想每个 batch 有 128 个数据。但是 1000 无法被 128 整除。你得到的结果是其中 7 个 batch 有 128 个数据点，一个 batch 有 104 个数据点。(7\*128 + 1\*104 = 1000)

batch 里面的数据点数量会不同的情况下，你需要利用 TensorFlow 的 **[tf.placeholder()](https://www.tensorflow.org/api_docs/python/tf/placeholder" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)**函数来接收这些不同的 batch。

继续上述例子，如果每个样本有 n\_input = 784 特征，n\_classes = 10 个可能的标签，features 的维度应该是 [None, n\_input]，labels 的维度是 [None, n\_classes]。

*# Features and Labels*

features = tf.placeholder(tf.float32, [**None**, n\_input])

labels = tf.placeholder(tf.float32, [**None**, n\_classes])

None 在这里做什么用呢？

None 维度在这里是一个 batch size 的占位符。在运行时，TensorFlow 会接收任何大于 0 的 batch size。

回到之前的例子，这个设置可以让你把 features 和 labels 给到模型。无论 batch 中包含 128，还是 104 个数据点。

# **9TensorFlow ReLUs 多层网络**

TensorFlow 提供了 ReLU 函数 **[tf.nn.relu()](https://www.tensorflow.org/api_docs/python/tf/nn/relu" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)**，如下所示：

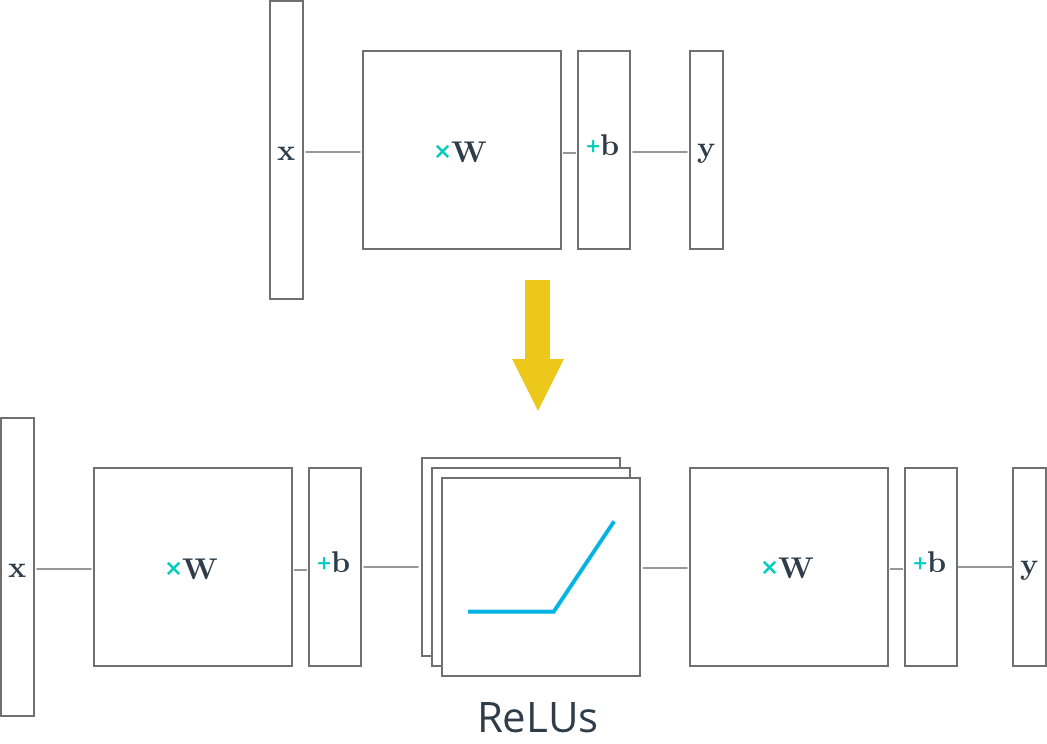
*# Hidden Layer with ReLU activation function# 隐藏层用 ReLU 作为激活函数*

hidden\_layer = tf.add(tf.matmul(features, hidden\_weights), hidden\_biases)

hidden\_layer = tf.nn.relu(hidden\_layer)

output = tf.add(tf.matmul(hidden\_layer, output\_weights), output\_biases)

上面的代码把**[tf.nn.relu()](https://www.tensorflow.org/api_docs/python/tf/nn/relu" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)** 放到隐藏层，就像开关一样把负权重关掉了。在激活函数之后，添加像输出层这样额外的层，就把模型变成了非线性函数。这个非线性的特征使得网络可以解决更复杂的问题。

**[](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)**

### **[优化器 Optimizer](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)**

*[# Define loss and optimizer# 定义误差值和优化器](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)*

[cost = tf.reduce\_mean(\](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)

[tf.nn.softmax\_cross\_entropy\_with\_logits(logits=logits, labels=y))](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)

[optimizer = tf.train.GradientDescentOptimizer(learning\_rate=learning\_rate)\](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)

[.minimize(cost)](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)

# **[10TensorFlow Dropout](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)**

**[](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/d5cf4454-1324-4524-9e2c-0ecca1f5c40e)**

**[图 1：来自论文 "Dropout: A Simple Way to Prevent Neural Networks from Overfitting" (](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/d5cf4454-1324-4524-9e2c-0ecca1f5c40e)[https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf](https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)[)](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/d5cf4454-1324-4524-9e2c-0ecca1f5c40e)**

[Dropout 是一个降低过拟合的正则化技术。它在网络中暂时的丢弃一些单元（](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)**[神经元](https://en.wikipedia.org/wiki/Artificial_neuron" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)**[），以及与它们的前后相连的所有节点。图 1 是 dropout 的工作示意图。](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)

[TensorFlow 提供了一个](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)**[tf.nn.dropout()](https://www.tensorflow.org/api_docs/python/tf/nn/dropout" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)**[函数，你可以用来实现 dropout。](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)

[让我们来看一个](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)**[tf.nn.dropout()](https://www.tensorflow.org/api_docs/python/tf/nn/dropout" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)**[的使用例子。](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)

[keep\_prob = tf.placeholder(tf.float32)](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25) *[# probability to keep units](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)*

[hidden\_layer = tf.add(tf.matmul(features, weights[0]), biases[0])](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)

[hidden\_layer = tf.nn.relu(hidden\_layer)](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)

[hidden\_layer = tf.nn.dropout(hidden\_layer, keep\_prob)](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)

[logits = tf.add(tf.matmul(hidden\_layer, weights[1]), biases[1])](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)

[上面的代码展示了如何在神经网络中应用 dropout。](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)

**[tf.nn.dropout()](https://www.tensorflow.org/api_docs/python/tf/nn/dropout" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)**[函数有两个参数：](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)

1. [hidden\_layer：你要应用 dropout 的 tensor](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)
2. [keep\_prob：任何一个给定单元的留存率（](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)****[没有](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)****[被丢弃的单元）](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)

[keep\_prob 可以让你调整丢弃单元的数量。为了补偿被丢弃的单元，](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)**[tf.nn.dropout()](https://www.tensorflow.org/api_docs/python/tf/nn/dropout" \t "https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/_blank)**[把所有保留下来的单元（](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)****[没有](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)****[被丢弃的单元）\* 1/keep\_prob](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)

[在训练时，一个好的keep\_prob初始值是0.5。](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)

[在测试时，把 keep\_prob 值设为1.0 ，这样保留所有的单元，最大化模型的能力。](https://classroom.udacity.com/nanodegrees/nd101-cn-advanced/parts/484efacf-1085-4481-9218-982074a4627c/modules/675d02b4-7881-4c86-8df8-7a6b3e12bbce/lessons/299af9c1-2ec7-4055-bdf9-5248ead9490d/concepts/b3de6cfa-ccd8-4a4d-b8c0-cdb47d81fd25)