



TensorFlow™

TensorFlow

- 2015年11月9日，Google发布第二代深度学习系统 TensorFlow 并宣布开源。
- 其命名来源于本身的运行原理。Tensor（张量）意味着N维数组，Flow（流）意味着基于数据流图的计算，TensorFlow 为张量从图象的一端流动到另一端计算过程。

TensorFlow



Demis Hassabis
@demishassabis



Following

Torch served us well, but DeepMind is moving to **#Tensorflow!** Excited to contribute to this new open source ML lib: goo.gl/Jud0i8

RETWEETS
244

LIKES
297



9:02 AM - 29 Apr 2016



...

学习资源

- Tensorflow.org
- github.com/tensorflow/tensorflow
- 极客学院 Wiki 团队的翻译
- Stack Overflow
- 中文圈QQ交流群

推荐环境(CPU)

- Ubuntu 14.04 (64位)
- Python 3.4/2.7

安装pip

```
1 | sudo apt-get install python3-pip python3-dev
```

安装tensorflow

```
1 | sudo pip3 install --upgrade https://storage.googleapis.com/tensorflow
```

基本使用

- 使用图 (graph) 来表示计算任务
图中的节点被称之为 op (operation)
- 使用张量(tensor) 表示数据
- 通过变量 (Variable) 维护状态
- 在会话 (Session) 中执行图

TEST

```
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
>>> print(sess.run(hello))
Hello, TensorFlow!
>>> a = tf.constant(10)
>>> b = tf.constant(32)
>>> print(sess.run(a + b))
42
>>>
```

TensorFlow Python 库有一个默认图 (*default graph*), op 构造器可以为其增加节点. 这个默认图对许多程序来说已经足够用了. 阅读 [Graph 类](#) 文档 来了解如何管理多个图.

```
import tensorflow as tf

# 创建一个常量 op, 产生一个 1x2 矩阵. 这个 op 被作为一个节点
# 加到默认图中.

#
# 构造器的返回值代表该常量 op 的返回值.

matrix1 = tf.constant([[3., 3.]])
# 创建另外一个常量 op, 产生一个 2x1 矩阵.

matrix2 = tf.constant([[2.], [2.]])
# 创建一个矩阵乘法 matmul op , 把 'matrix1' 和 'matrix2' 作为输入.

# 返回值 'product' 代表矩阵乘法的结果.

product = tf.matmul(matrix1, matrix2)
```

默认图现在有三个节点, 两个 `constant()` op, 和一个 `matmul()` op. 为了真正进行矩阵相乘运算, 并得到矩阵乘法的结果, 你必须在会话里启动这个图.

MNIST

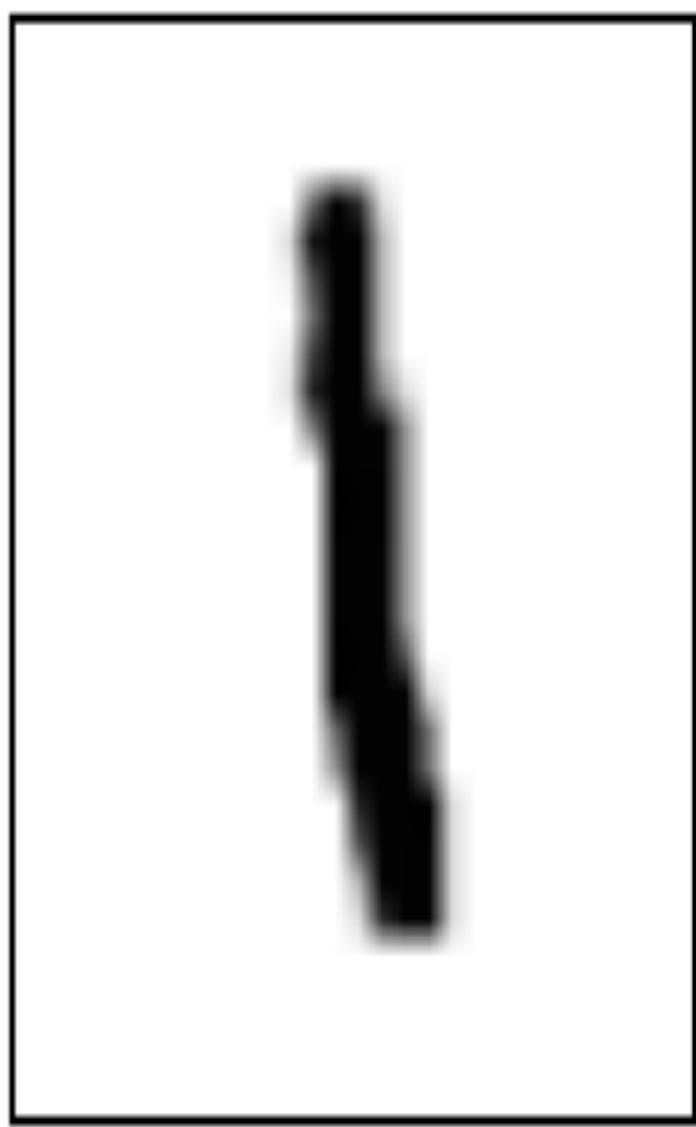
- 入门级的计算机视觉数据集，机器学习的“Hello World”
- 它包含各种手写数字图片和每一张图片对应的标签（告诉我们这个是数字几）。
- 我们将训练一个机器学习模型用于预测图片里面的数字。

获取MNIST数据

- 下载input_data.py
- 55000行训练数据集 (mnist.train)
10000行测试数据集 (mnist.test)
5000行验证数据 (mnist.validation)
- 手写数字的图片，设为 “xs”
对应的标签，设为 “ys”

```
import tensorflow as tf
import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)
```

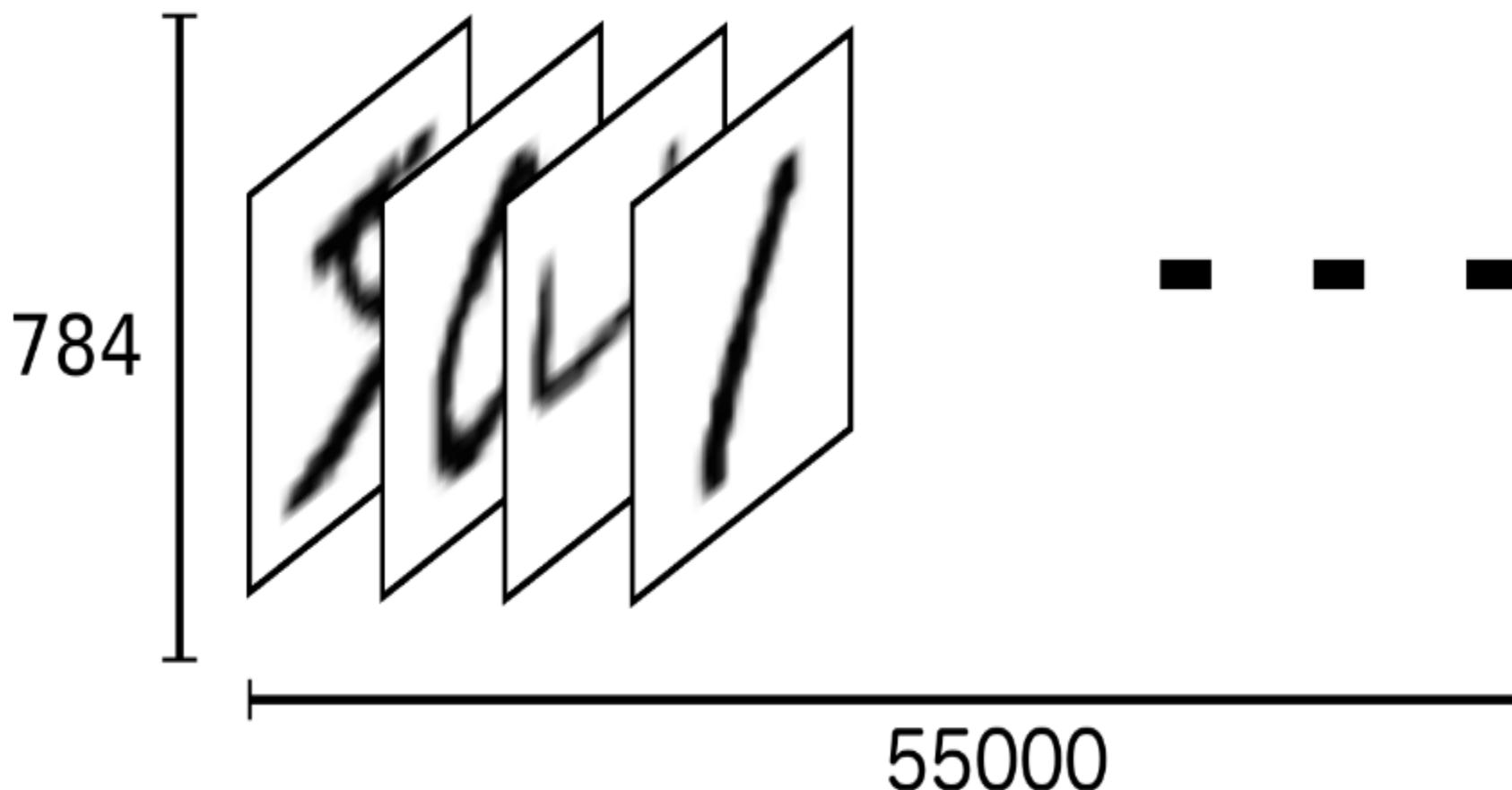
每张图片包含 $28 \times 28 = 784$ 个像素点，如图。



2

因此，在MNIST训练数据集中，`mnist.train.images`是一个形状为 [55000, 784] 的张量，第一个维度数字用来索引图片，第二个维度数字用来索引每张图片中的像素点。在此张量里的每一个元素，都表示某张图片里的某个像素的强度值，值介于0和1之间。

`mnist.train_xs`



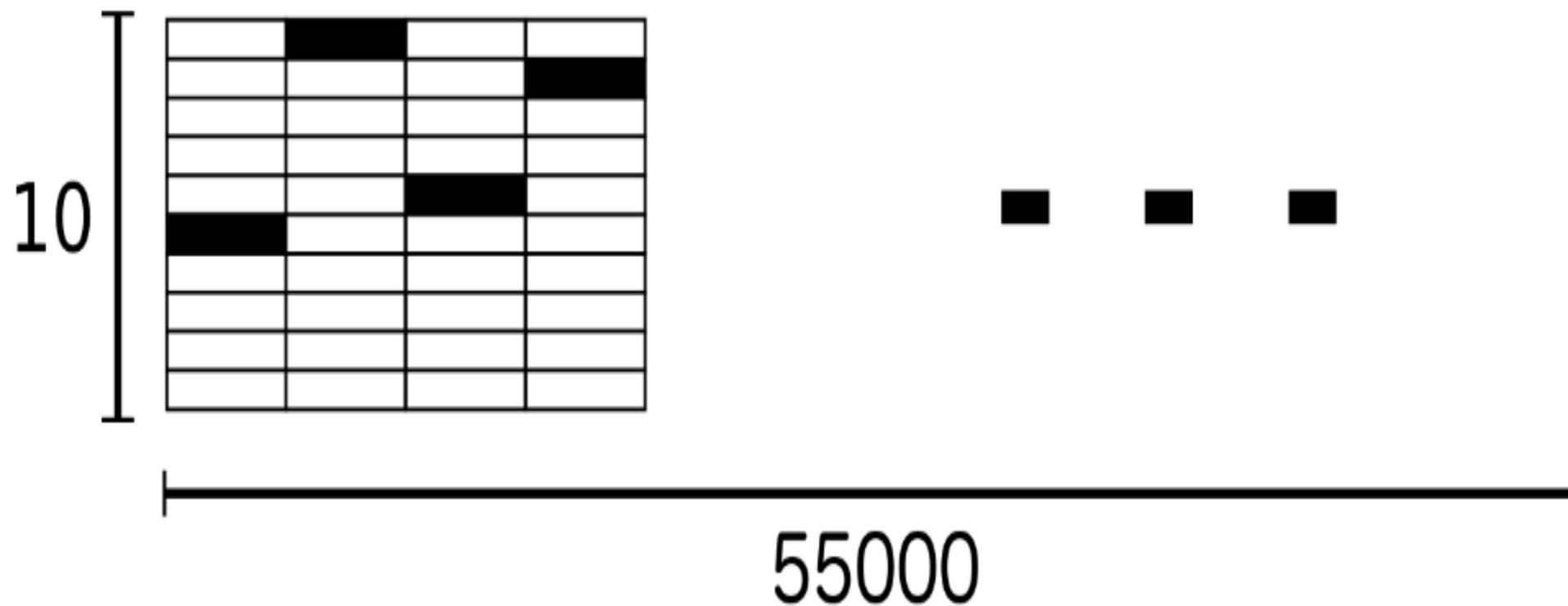
独热编码

- 使用N位状态寄存器来对N个状态进行编码，每个状态都有它独立的寄存器位，并且在任意时候，其中只有一位有效。
- 自然顺序码为 000,001,010,011,100,101
独热编码则是
000001,000010,000100,001000,010000,
100000

于是，标签 3 可以表示成 $[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]$ 。

因此，`mnist.train.labels` 是一个 $[55000, 10]$ 的浮点值的矩阵。

`mnist.train.ys`

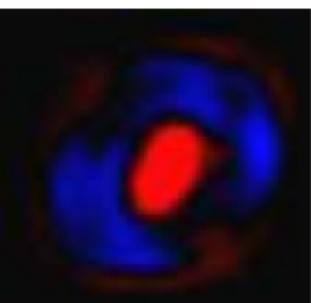


Softmax回归

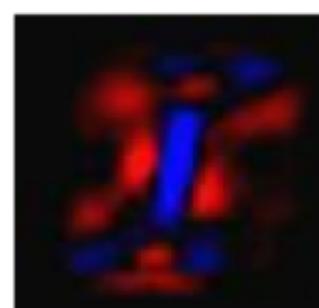
- 适用于: 分类, 类别数量大于2
- 得到一张图片代表每个数字的概率
- 比如说, 我们的模型可能推测一张包含9的图片代表数字9的概率是80%但是判断它是8的概率是5% (因为8和9都有上半部分的小圆), 代表其他数字的概率更小。

为了得到一张给定图片属于某个特定数字类的证据，我们对图片像素值进行加权求和。如果这个像素具有很强的证据说明这张图片不属于该类，那么相应的权值为负数，相反如果这个像素拥有有利的证据支持这张图片属于这个类，那么权值是正数。

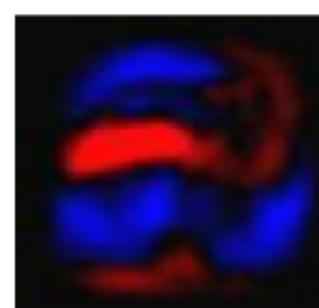
下面的图片显示了一个模型学习到的图片上每个像素对于特定数字类的权值。红色代表负数权值，蓝色代表正数权值。



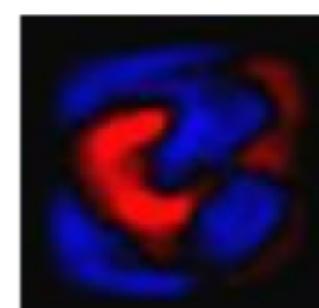
0



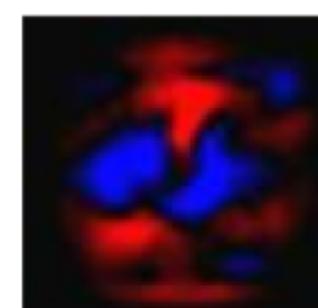
1



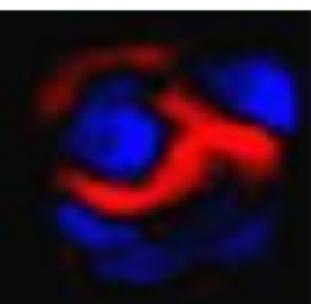
2



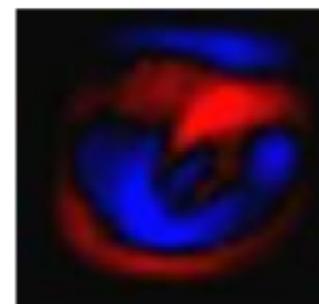
3



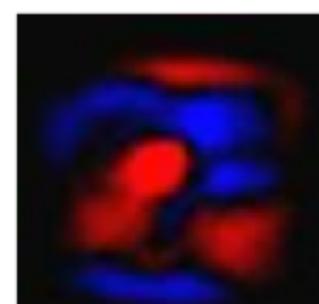
4



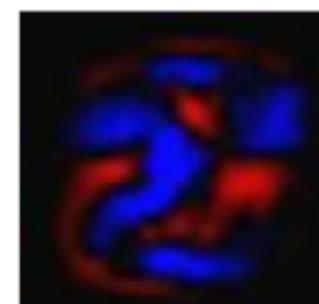
5



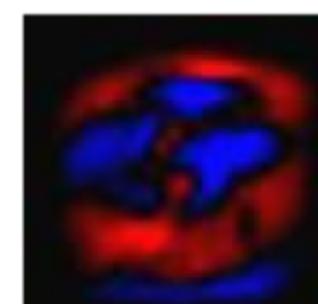
6



7



8



9

Softmax回归

- Softmax回归分成两步：
 1. 统计每张图片的证据 (evidence) 和

我们也需要加入一个额外的偏置量 (*bias*) , 因为输入往往带有一些无关的干扰量。因此对于给定的输入图片 x 它代表的是数字 i 的证据可以表示为:

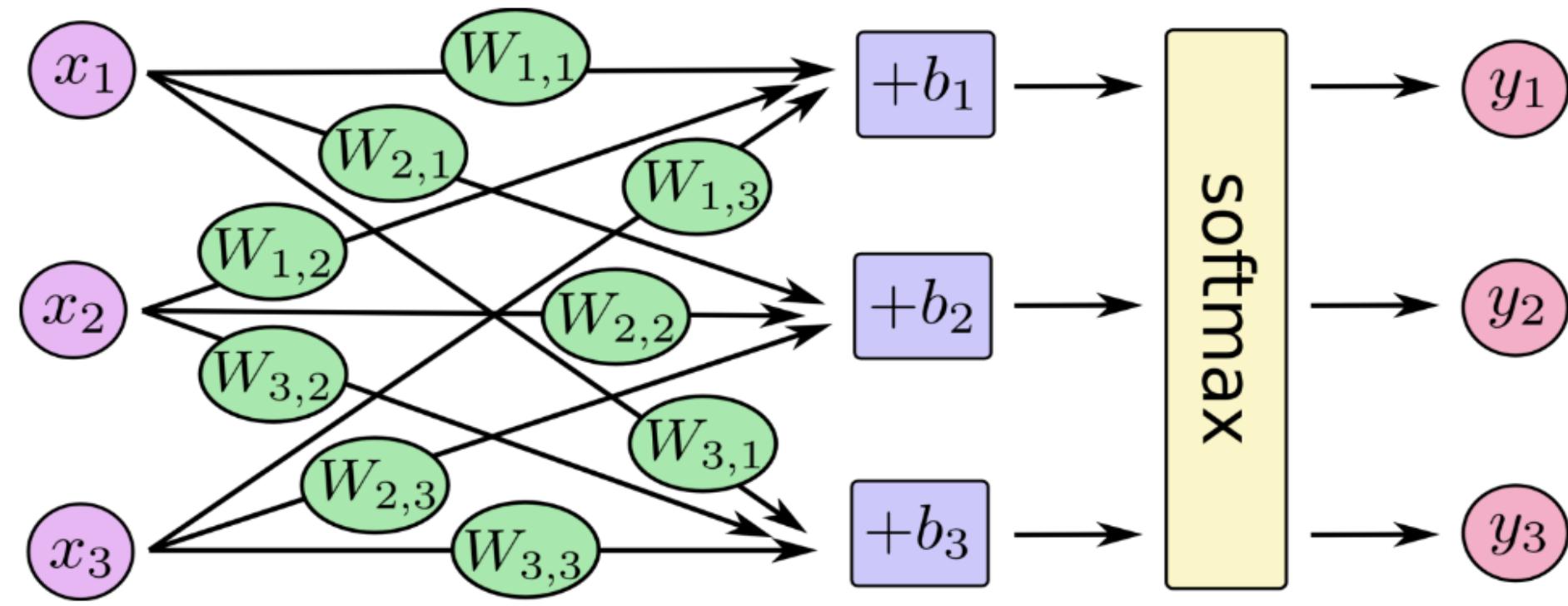
$$\text{evidence}_i = \sum_j W_{i,j} x_j + b_i$$

W_i 代表权重 , b_i 代表数字 i 类的偏置量 , j 代表给定图片 x 的像素索引用于像素求和。

2. 把证据转化为概率

$$y = \text{softmax}(\text{evidence})$$

用一张图来表示：



写成等式：

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax} \begin{pmatrix} W_{1,1}x_1 + W_{1,2}x_2 + W_{1,3}x_3 + b_1 \\ W_{2,1}x_1 + W_{2,2}x_2 + W_{2,3}x_3 + b_2 \\ W_{3,1}x_1 + W_{3,2}x_2 + W_{3,3}x_3 + b_3 \end{pmatrix}$$

用矩阵表示：

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax} \left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

$$\bullet Y = \text{softmax}(Wx + b)$$

实现回归模型

```
4 x = tf.placeholder("float", [None, 784])  
5 w = tf.Variable(tf.zeros([784, 10]))  
6 b = tf.Variable(tf.zeros([10]))  
7 y = tf.nn.softmax(tf.matmul(x, w) + b)
```

第4行代码中，x是一个“占位符”，在TensorFlow计算时输入这个值。其中None代表图片的数量，表示可以是任意张；784表示每张图的像素。

第5、6行代码中，用Variable表示权重值和偏置量，Variable可以在计算中被修改。这里我们都用全为零的张量来初始化W和b。因为我们要学习W和b的值，它们的初值可以随意设置。

最后实现模型只需要一行代码（第7行）。

训练模型

- 机器学习中，通常定义一个指标衡量模型好坏，称为成本（cost）或损失（loss），然后尽量最小化这个指标。
- 交叉熵、反向传播算法、梯度下降算法、随机训练

交叉熵是一个常见又好用的成本函数。

它产生于信息论里面的信息压缩编码技术，后来演变成为从博弈论到机器学习等其他领域里的重要技术手段。

定义如下：

$$H_{y'}(y) = -\sum_i y'_i \log(y_i)$$

y 是我们预测的概率分布, y' 是实际的分布 (我们输入的one-hot vector)。

如第8行代码所示，为了计算交叉熵，我们首先需要添加一个新的占位符用于输入正确值。

然后我们可以用

$$-\sum y' \log(y)$$

计算交叉熵：第九行代码，用 `tf.log` 计算 y 的每个元素的对数。接下来，我们把 `y_` 的每一个元素和 `tf.log(y)` 的对应元素相乘。最后，用 `tf.reduce_sum` 计算张量的所有元素的总和。

```
8 | y_ = tf.placeholder("float", [None, 10])
9 | cross_entropy = -tf.reduce_sum(y_*tf.log(y))
```

反向传播算法

- 一种与最优化方法结合使用的，用来训练人工神经网络的常见方法。
- TensorFlow可以自动地使用反向传播算法来有效地确定你的变量是如何影响你想要最小化的那个成本值的。然后TensorFlow会用你选择的优化算法来不断地修改变量以降低成本。

梯度下降算法

- 一种最优化算法

```
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
```

第10行代码中，我们要求TensorFlow用**梯度下降算法**（gradient descent algorithm）（一种最优化算法）以0.01的学习速率最小化交叉熵。

梯度下降算法是一个简单的学习过程，TensorFlow只需将每个变量一点点地往使成本不断降低的方向移动。当然TensorFlow也提供了其他许多优化算法：只要简单地调整一行代码就可以使用其他的算法。

```
11 |     init = tf.initialize_all_variables()  
12 |     sess = tf.Session()  
13 |     sess.run(init)
```

随机训练

```
14 for i in range(1000):  
15     batch_xs, batch_ys = mnist.train.next_batch(100)  
16     sess.run(train_step, feed_dict = {x:batch_xs, y_:batch_y}
```

14~16行是训练模型1000次。

每次循环随机提取100个训练数据作为一批 (*batch*) ,来替换之前的占位符来运行 *train_step* 。

使用一小部分的随机数据来进行训练被称为**随机训练 (stochastic training)** - , 在这里更确切的说是随机梯度下降训练。

在理想情况下，我们希望用我们所有的数据来进行每一步的训练，因为这能给我们更好的训练结果，但显然这需要很大的计算开销。所以，每一次训练我们可以使用不同的数据子集，这样做既可以减少计算开销，又可以最大化地学习到数据集的总体特性。

评估模型

```
18 correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))  
19 accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
```

首先我们找出那些预测正确的标签。

第18行代码中，`tf.argmax` 是一个非常有用的函数，它能给出某个`tensor`对象在某一维上的其数据最大值所在的索引值。由于标签向量是由0,1组成，因此最大值1所在的索引位置就是类别标签，比如 `tf.argmax(y,1)` 返回的是模型对于任一输入`x`预测到的标签值，而 `tf.argmax(y_,1)` 代表正确的标签，我们可以用 `tf.equal` 来检测我们的预测是否真实标签匹配(索引位置一样表示匹配)。

第19行代码会给我们一组布尔值。为了确定正确预测项的比例，我们可以把布尔值转换成浮点数，然后取平均值。例如，`[True, False, True, True]` 会变成 `[1,0,1,1]`，取平均值后得到 0.75。

最后(第21行代码),我们计算所学习到的模型在测试数据集上面的正确率。结果大约是91%~92%。

```
(tensorflow) lizhao@lab603-1:~/tensorflow/MNIST$ python a.py
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
0.9147
(tensorflow) lizhao@lab603-1:~/tensorflow/MNIST$ python a.py
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
0.9137
(tensorflow) lizhao@lab603-1:~/tensorflow/MNIST$ python a.py
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
0.9185
```

- 事实上，这个结果是很差的。
- 这是因为我们仅仅使用了一个非常简单的模型。
- 做一些小小的改进，我们就可以得到97%的正确率。
- 最好的模型甚至可以获得超过99.7%的准确率。
- 体会TensorFlow的设计思想。



完整版代码：

```
1 import tensorflow as tf
2 import input_data
3 mnist = input_data.read_data_sets("MNIST_data/", one_hot = True)
4 x = tf.placeholder("float", [None, 784])
5 W = tf.Variable(tf.zeros([784, 10]))
6 b = tf.Variable(tf.zeros([10]))
7 y = tf.nn.softmax(tf.matmul(x, W) + b)
8 y_ = tf.placeholder("float", [None, 10])
9 cross_entropy = -tf.reduce_sum(y_*tf.log(y))
10 train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_
11 init = tf.initialize_all_variables()
12 sess = tf.Session()
13 sess.run(init)
14 for i in range(1000):
15     batch_xs, batch_ys = mnist.train.next_batch(100)
16     sess.run(train_step, feed_dict = {x:batch_xs, y_:batch_ys})
17
18 correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
19 accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
20
21 print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.
```