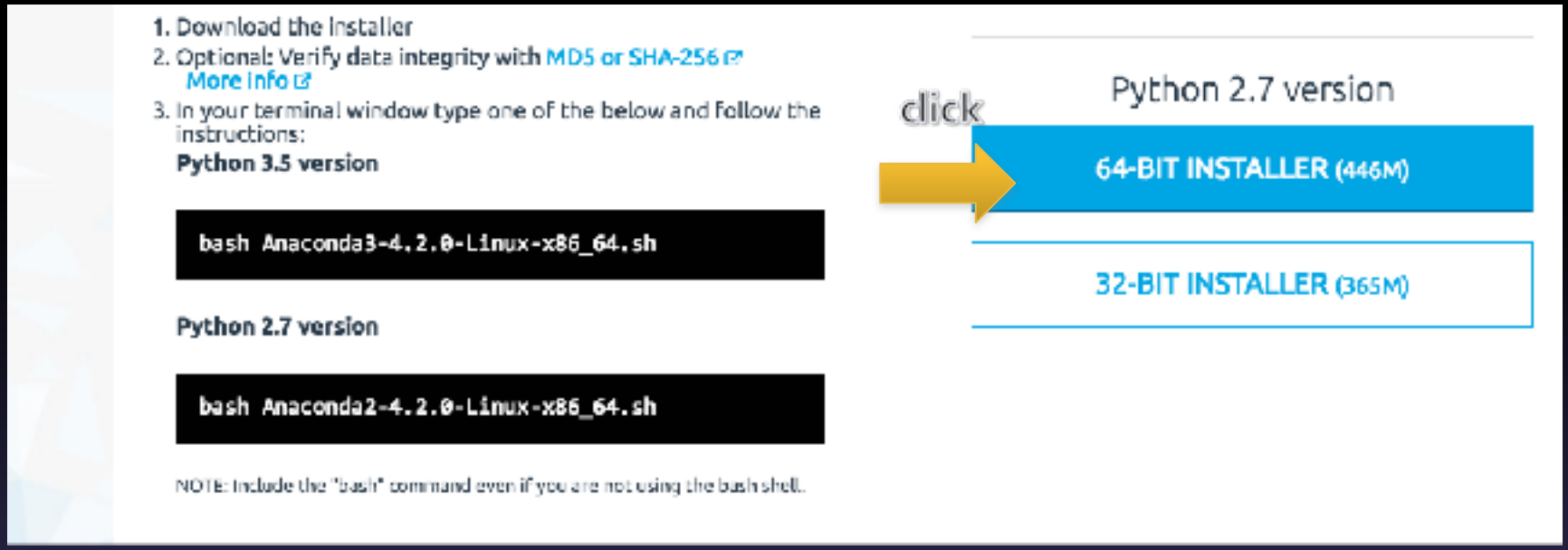
Install Ubuntu

1.All you need to do is just click “continue”

2.Tip:Allocating more space for ubuntu(default 8G)

Install Anaconda



Run command under terminal:

"bash ~/Downloads/Anaconda3-4.0.0-Linux-x86_64.sh"

NOTE: If you select the option to not add the Anaconda directory to your bash shell PATH environment variable, you may later add this line to the file .bashrc in your home directory: export PATH="/home/username/anaconda/bin:$PATH" Replace /home/username/anaconda with your actual path.

Install TensorFlow

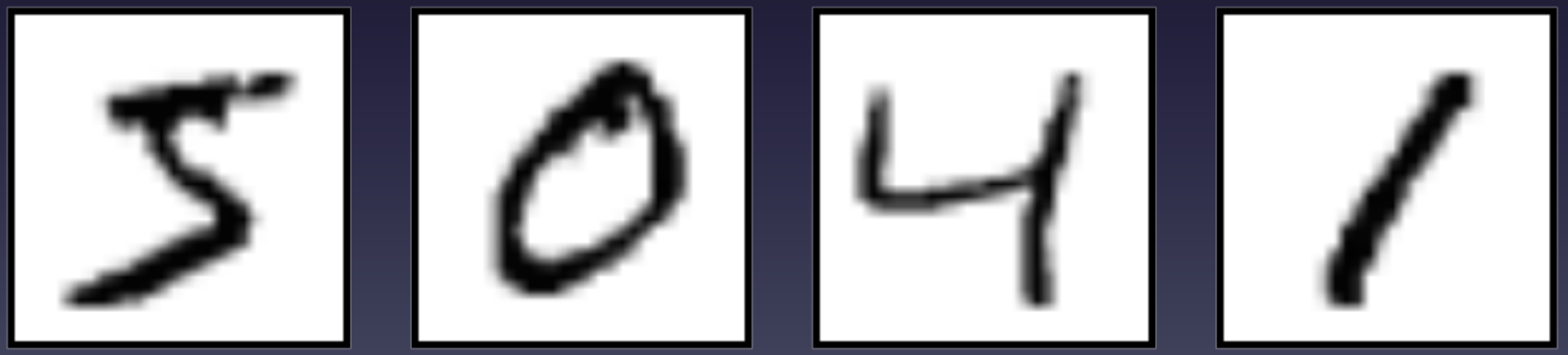
- 1.To install this package with conda run:
  - conda install -c jjhelmus tensorflow=0.10.0rc0
- 2.pip install
  - To install this package with pip run:
  - pip install -i https://pypi.anaconda.org/jjhelmus/simple tensorflow

What is MNIST?

The MNIST database (Mixed National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems.

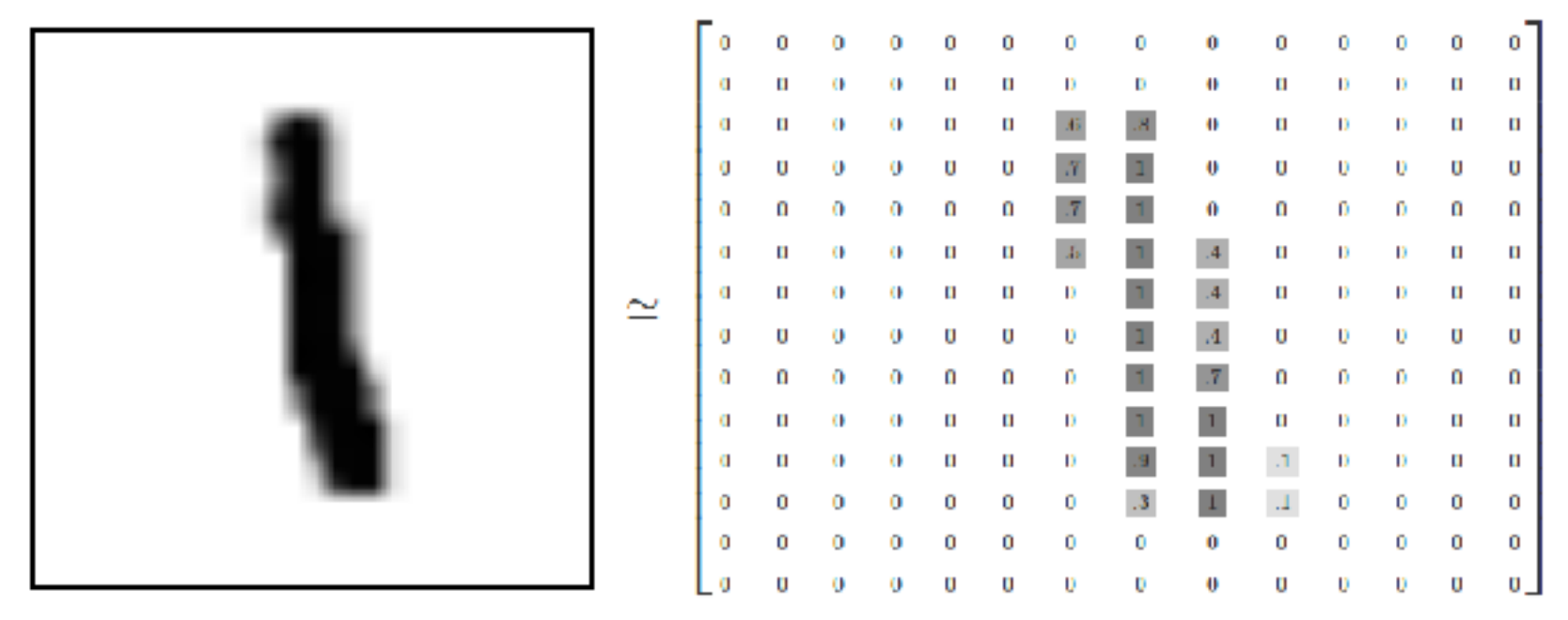
MNIST is look like..

- MNIST is a simple computer vision dataset. It consists of images of handwritten digits like these:



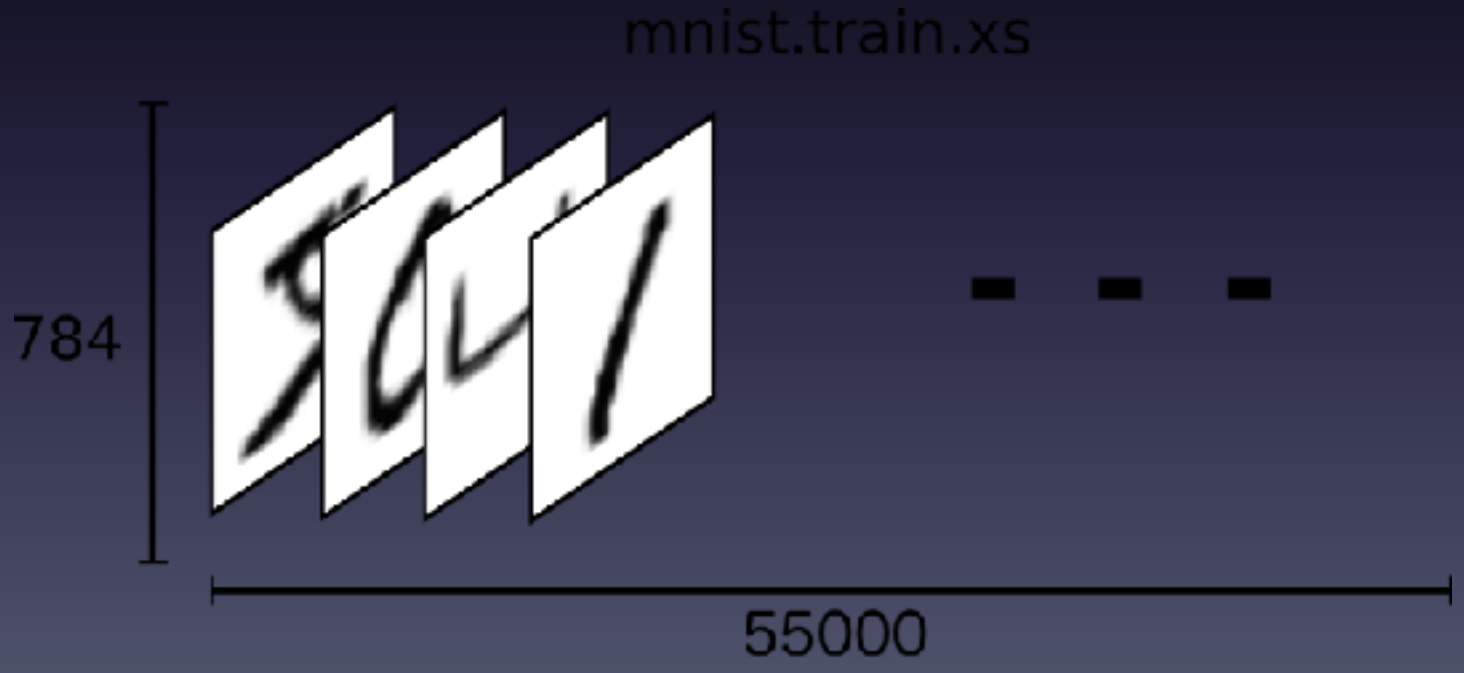
MNIST in file..

- Each image is 28 pixels by 28 pixels. We can interpret this as a big array of numbers:



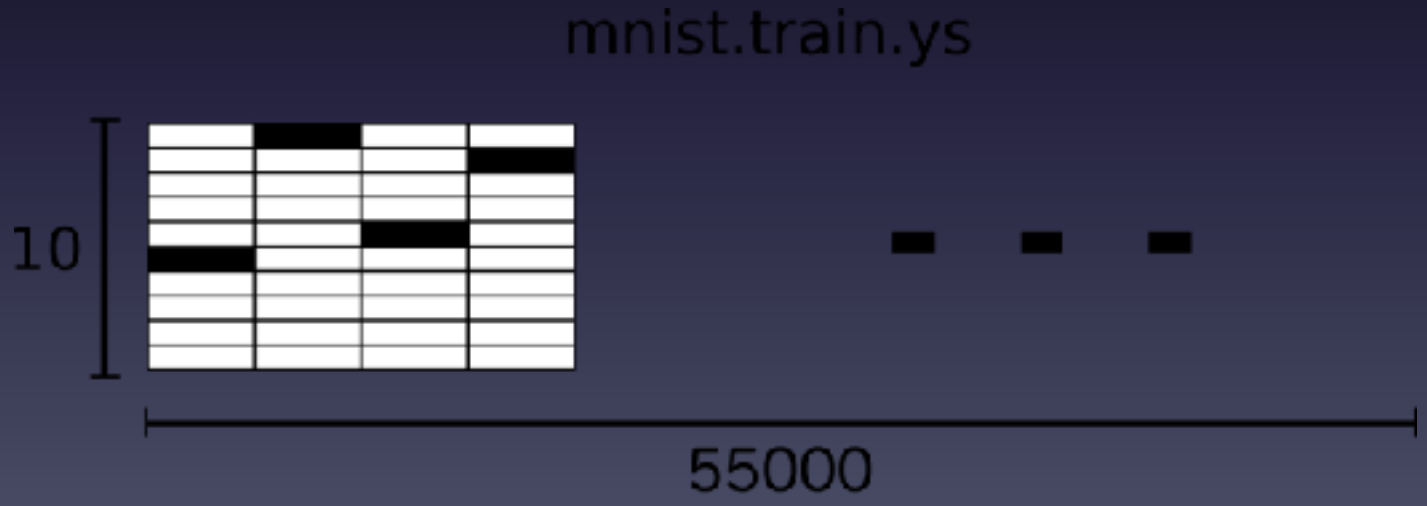
Training set's input is look like

- The result is that mnist.train.images is a tensor (an n-dimensional array) with a shape of [55000, 784].The first dimension is an index into the list of images and the second dimension is the index for each pixel in each image.

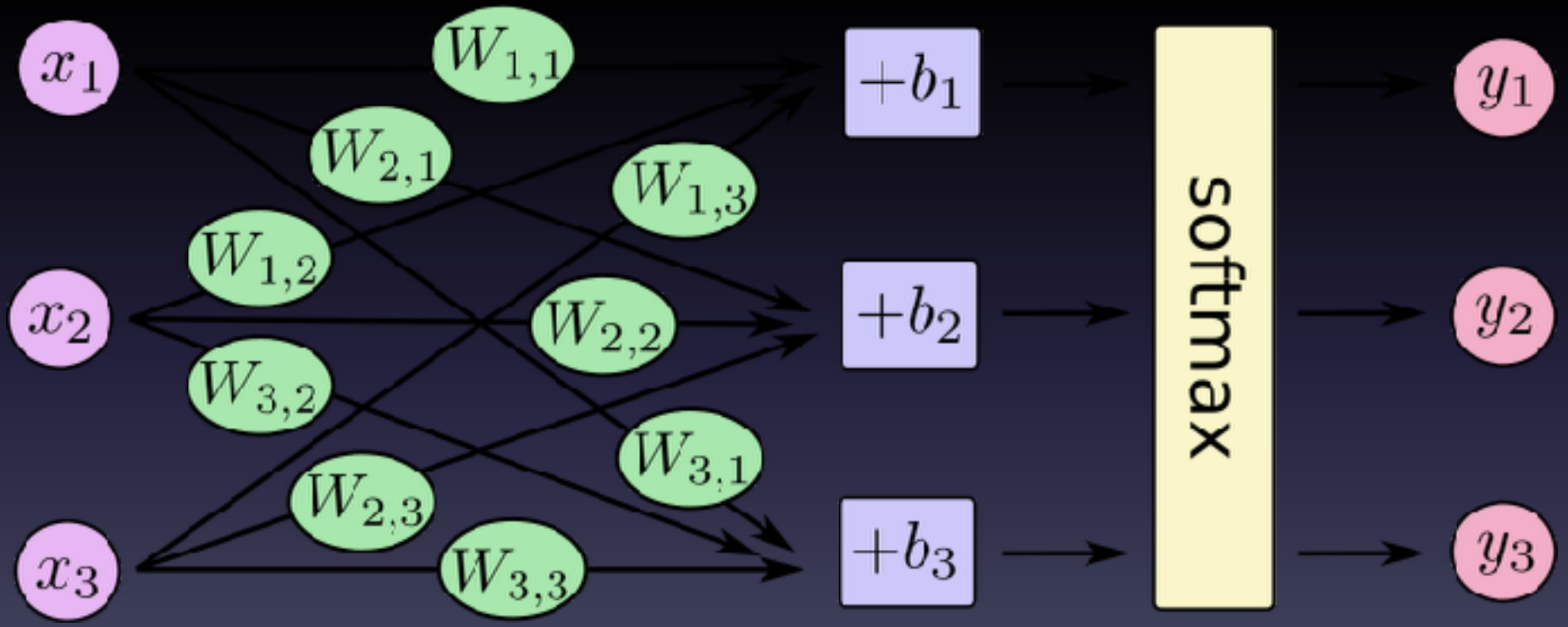


Training set's output is look like

- For example, 3 would be [0,0,0,1,0,0,0,0,0,0]. Consequently, mnist.train.labels is a [55000, 10] array of floats.



Model is look like



What is softmax?

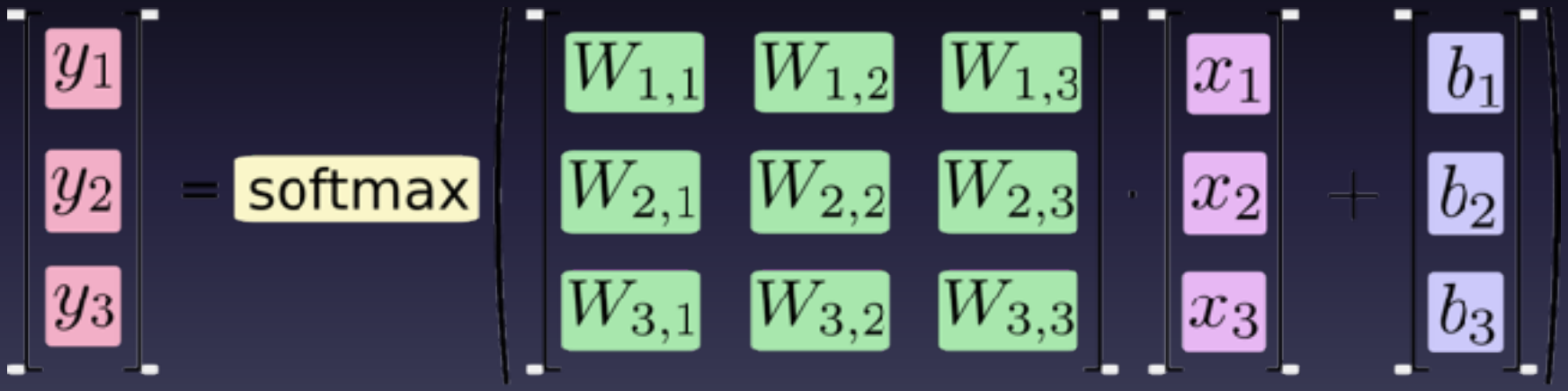
- there are only ten possible things that a given image can be. We want to be able to look at an image and give the probabilities for it being each digit.
- A softmax regression has two steps: first we add up the evidence of our input being in certain classes, and then we convert that evidence into probabilities.

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

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

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

Predictive in Mathmatical Way.



Coding

Import Library & Data

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import argparse

# Import data
from tensorflow.examples.tutorials.mnist import input_data

import tensorflow as tf
```

Import所需的Library

Import所需的Training data和Testing data

Define Model and cost function

```
# Create the model
x = tf.placeholder(tf.float32, [None, 784])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
y = tf.matmul(x, W) + b

# Define loss and optimizer
y_ = tf.placeholder(tf.float32, [None, 10])

# The raw formulation of cross-entropy,
#
#   tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(tf.softmax(y)),
#                                 reduction_indices=[1]))
#
# can be numerically unstable.
#
# So here we use tf.nn.softmax_cross_entropy_with_logits on the raw
# outputs of 'y', and then average across the batch.
cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(y, y_))
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
```

初始化模型，輸入為784維(圖大小28X28)的向量，輸出為10維 的向量(0-9)

定義Cost Function，採用Cross Entropy函數，並在輸出端加入SoftMax，使每個輸出值介於0-1 間

Training Model

```
# Train
tf.initialize_all_variables().run()
for _ in range(1000):
  batch_xs, batch_ys = mnist.train.next_batch(100)
  sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
```

開始訓練模型，更新Weight時跑1000個epoch，並且每次隨機取100個Data

Predictive

```
# Test trained model
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print(sess.run(accuracy, feed_dict={x: mnist.test.images,
                                    y_: mnist.test.labels}))
```

拿Testing data測試ModelTraining結果，
並印出其精準度

Predictive Result(ipython notrbook)

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

# Test trained model
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print(accuracy.eval({x: mnist.test.images, y_: mnist.test.labels}))
```

```
Extracting /tmp/data/train-images-idx3-ubyte.gz
Extracting /tmp/data/train-labels-idx1-ubyte.gz
Extracting /tmp/data/t10k-images-idx3-ubyte.gz
Extracting /tmp/data/t10k-labels-idx1-ubyte.gz
0.9172
```

With 91.72% accuracy!

Reproduce the weight images

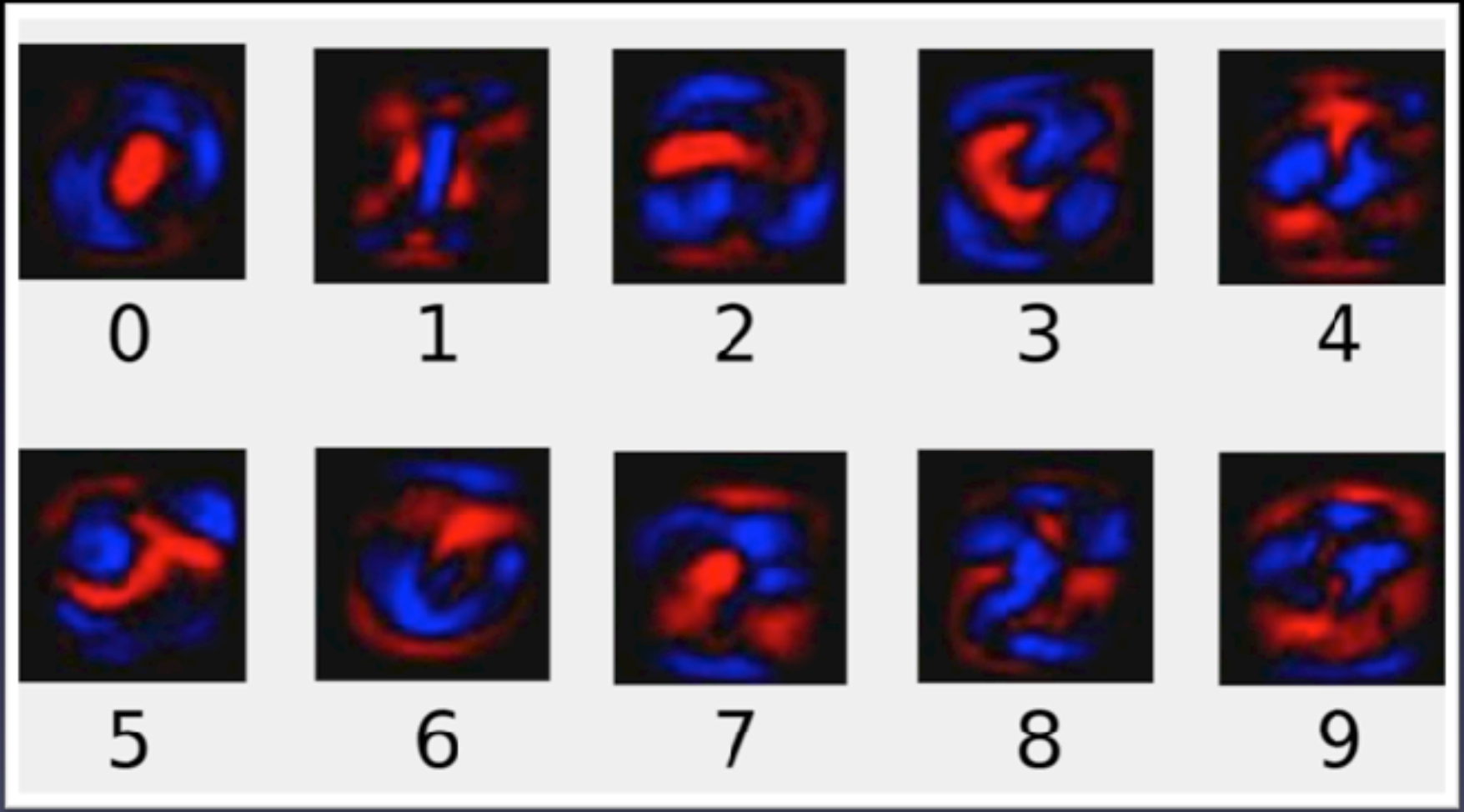
```
# A red/black/blue colormap
cdict = {'red':   [(0.0,  1.0, 1.0),
                   (0.25,  1.0, 1.0),
                   (0.5,  0.0, 0.0),
                   (1.0,  0.0, 0.0)],
         'green': [(0.0,  0.0, 0.0),
                   (1.0,  0.0, 0.0)],
         'blue':  [(0.0,  0.0, 0.0),
                   (0.5,  0.0, 0.0),
                   (0.75, 1.0, 1.0),
                   (1.0,  1.0, 1.0)]}
redblue = matplotlib.colors.LinearSegmentedColormap('red_black_blue',cdict,256)
```

初始matplotlib所需的參數，並採用LinearSegmentedColormap函數繪出所需的圖

```
wts = W.eval(sess)
for i in range(0,10):
    im = wts.flatten()[i::10].reshape((28,-1))
    plt.imshow(im, cmap = redblue, clim=(-1.0, 1.0))
    plt.colorbar()
    print "Digit %d" % i
    plt.show()
```

提取並繪出從0-9的Weight值，其中wts包含weight值

Result of reproduce the weight images



END