目　录　（4 号黑体，居中）

1 项目目的

学习神经网络在分类预测中的应用，理解 softmax 函数的相关知识，了解和使用 TensorFlow 开发环境。

2 项目环境和条件

硬件环境：win10 笔记本

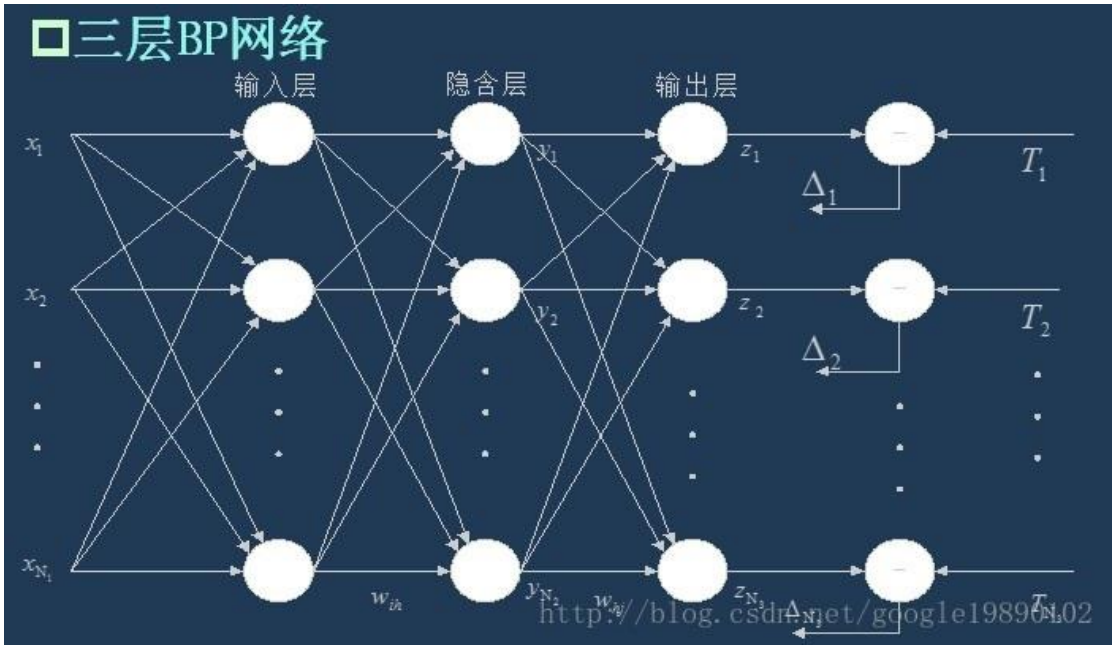
软件环境：ubuntu64 虚拟机，TensorFlow 环境，火狐浏览器

3 项目原理

主要介绍算法原理

(一) 三层 BP 神经网络

BP 神经网络是一种多层的前馈神经网络，其主要的特点是：信号是前向传播的，而误差是反向传播的。具体来说，如下图所示的就是只含一个隐层的神经网络模型：



(三层神经网络示意图)

BP 神经网络的过程主要分为两个阶段，第一阶段是信号的前向传播，从输入层经过隐含层，最后到达输出层；第二阶段是误差的反向传播，从输出层到隐含层，最后到输入层，依次调节隐含层到输出层的权重和偏置，输入层到隐含层的权重和偏置。

(二) 权重和偏置更新原理

（1）梯度下降

梯度下降算法是最普遍的优化算法，不过梯度下降算法需要用到全部的样本，训练速度

比较慢，但是迭代到一定的次数最终能够找到比较合适的最优解。在 tensorflow 中用一个函数即可实现梯度下降的功能，即“optimizer＝tf.train.GradientDescentOptimizer(learning_rate)”。

梯度下降算法的迭代原理如下图所示：

$$x_2 = x_1 - \eta \cdot \frac{\partial f}{\partial x_1}$$

$$x_3 = x_2 - \eta \cdot \frac{\partial f}{\partial x_2}$$

$$x_n = x_{n-1} - \eta \cdot \frac{\partial f}{\partial x_{n-1}}$$

（梯度下降算法的迭代原理）

（2）方向误差传播

反向传播算法（Backpropagation）是目前用来训练神经网络的最常用且最有效的算法。其主要思想是：

1）将训练集数据输入到 ANN 的输入层，经过隐藏层，最后达到输出层并输出结果，这是 ANN 的前向传播过程；

2）由于 ANN 的输出结果与实际结果有误差，则计算估计值与实际值之间的误差，并将该误差从输出层向隐藏层反向传播，直至传播到输入层；

3）在反向传播的过程中，根据误差调整各种参数的值；不断迭代上述过程，直至收敛。

反向传播算法的思想比较容易理解，但具体的公式则要一步步推导，因此下面主要介绍介绍权重和偏置更新公式的推导过程。

（3）权重和偏置更新公式

1）网络初始化

首先要对神经网络各层的参数进行设置，包括输入层的节点个数、隐含层的节点个数、输出层的节点个数、输入层到隐含层的权重、隐含层到输出层的权重、输入层到隐含层的偏

置、隐含层到输出层的偏置、梯度下降算法的学习率、以及训练过程中使用到的激励函数。参数设置如下表所示：

假设输入层的节点个数为n

隐含层的节点个数为l

输出层的节点个数为m

输入层到隐含层的权重为W_{ij}

隐含层到输出层的权重为W_{jk}

输入层到隐含层的偏置为a_j

隐含层到输出层的偏置为b_k

学习率为η

激励函数g(x)=Sigmoid(x)

（针对神经网络的参数设置）

2）输入层的数据经过权重、偏置以及激励函数的处理之后，从隐含层会输出一层数据，输出函数如下所示：

$$隐含层的输出H_j = g(\Sigma_{i=1}^{i=n} W_{ij}x_i + a_j)$$

3）隐含层的数据经过权重、偏置以及激励函数的处理之后，从输出层会输出最终结果数据，输出函数如下所示：

$$输出层的输出O_k = \Sigma_{j=1}^{j=l} W_{jk}H_j + b_k$$

4）在进行训练以及测试的过程中，会出现一定的误差，这是很重要的参数，因为后面要根据误差来调整偏置和权重，对神经网络进行优化，误差计算函数如下所示：

误差公式：$E = \frac{1}{2}\Sigma_{k=1}^{k=m}(Y_k - O_k)^2$

其中Y_k为期望输出，记$Y_k - O_k = e_k$，则E可表示为：

$$E = \frac{1}{2}\Sigma_{k=1}^{m} e_k^2$$

5）权重更新

A、Hidden 层→$Output$层的权重更新公式推导：首先对误差函数中的权重求偏导，再由梯度下降法即可得到权重更新公式。

$$\frac{\partial E}{\partial W_{jk}} = \Sigma_{k=1}^{m}(Y_k - O_k)(-\frac{\partial O_k}{\partial W_{jk}})$$
$$= -e_k.H_j$$

由梯度下降法可知权重更新公式为：

$$\boldsymbol{W'_{jk} = W_{jk} + \eta.H_j e_k}$$

B、Input 层→Hidden 层的权重更新公式推导：先对误差函数中的权重求偏导，再对误差函数中的隐含层输出求偏导，最后由梯度下降法得到权重更新公式。

$$\frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial H_j}\frac{\partial H_j}{\partial W_{ij}} \quad ,\text{其中}$$

$$\frac{\partial E}{\partial H_j} = \Sigma_{k=1}^{m}(Y_k - O_k)\left(-\frac{\partial O_k}{\partial H_j}\right) = -\Sigma_{k=1}^{m} e_k.W_{jk}$$

$$\frac{\partial H_j}{\partial W_{ij}} = \frac{\partial g(\Sigma_{i=1}^{n} W_{ij}x_i + a_j)}{\partial W_{ij}} = H_j(1-H_j)x_i$$

由梯度下降法可知权重更新公式为：

$$\boldsymbol{W'_{ij} = W_{ij} + \eta.H_j(1-H_j)x_i\Sigma_{k=1}^{m} e_k.W_{jk}}$$

6）偏置更新

A、Hidden 层→$Output$层的偏置更新公式推导：首先对误差函数中的偏置求偏导，再由梯度下降法即可得到偏置更新公式。

$$\frac{\partial E}{\partial b_k} = (Y_k - O_k)\left(-\frac{\partial O_k}{\partial b_k}\right) = -e_k$$

由梯度下降法可知偏置更新公式为：

$$\boldsymbol{b_k' = b_k + \eta e_k}$$

B、Input 层→*H*idden 层的偏置更新公式推导：先对误差函数中的偏置求偏导，再对误差函数中的隐含层输出求偏导，最后由梯度下降法得到偏置更新公式。

$$\frac{\partial E}{\partial a_j} = \frac{\partial E}{\partial H_j}\frac{\partial H_j}{\partial a_j} \quad ，其中$$

$$\frac{\partial E}{\partial H_j} = \Sigma_{k=1}^{m}(Y_k - O_k)(-\frac{\partial O_k}{\partial H_j}) = -\Sigma_{k=1}^{m} e_k . W_{jk}$$

$$\frac{\partial H_j}{\partial a_j} = \frac{\partial g(\Sigma_{i=1}^{n} W_{ij}x_i + a_j)}{\partial a_j} = H_j(1 - H_j)$$

由梯度下降法可知偏置更新公式为：

$$\boldsymbol{a_j' = a_j + \eta H_j(1 - H_j)\Sigma_{k=1}^{m} e_k . W_{jk}}$$

4 项目内容

（1）理清思路，弄明白一个机器学习模型是如何建立并训练的。

（2）softmax 回归模型

我们的分类预测模型的输出结果是当前所预测的图片代表每个数字的概率，取最大的概率作为预测结果。于是我们有两个步骤：

第一步，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。我们还需要加入一个额外的偏置量（bias），因为输入往往会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：

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

其中 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。该线性函数的输出是一个十维向量。

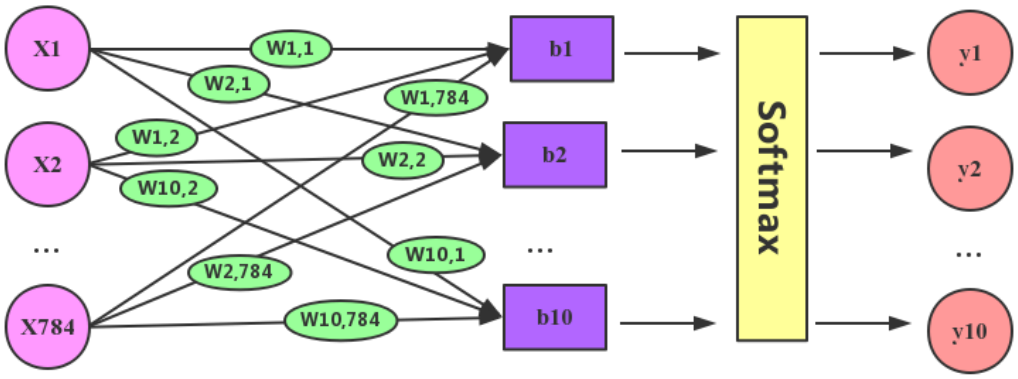
第二步，用 softmax 函数把这些证据转换成概率 y：

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

Softmax 函数的定义：

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

所以本次实验的模型如下图所示：



所得的 y 就是我们想要的它是每个数字的概率。

（3）损失函数在模型评估中的作用

为了训练我们的模型，我们需要定义一个指标来评估这个模型的好坏。在机器学习中通常定义一个指标来表示这个模型是坏的，这个指标称为成本（cost）或损失（loss），然后尽量最小化这个指标。

本次实验中我们采用的损失函数是“交叉熵”（cross-entropy）。交叉熵产生于信息论里面的信息压缩编码技术，但是它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。它的定义如下：

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

y 是我们预测的概率分布，**y'** 是实际的分布。交叉熵表示的是我们的预测有多不符合正确

标签，然后接下来会使用梯度下降算法来调整模型的参数（全重 W 和偏置值 b），尽量使交叉熵变到最小。

（4）TensorFlow 和 TensorBoard 的使用及理解

TensorFlow 是一个采用数据流图（data flow graphs），用于数值计算的开源软件库。节点（Nodes）在图中表示数学操作，图中的线（edges）则表示在节点间相互联系的多维数据数组，即张量（tensor）。它可以在多种平台上展开计算。使用 TensorFlow, 你必须明白以下几个要点：

- 在 tensorflow 库在被加载的时候,它会自动创建一个 Graph 对象,并把它作为默认的数据流图. TensorFlow 的计算都是基于数据流图的。如果不特殊指定，会使用系统默认图。
- Operation 就是 Tensorflow Graph 中的一个计算节点。其接收零个或者多个 Tensor 对象作为输入，然后产生零个或者多个 Tensor 对象作为输出。
- 张量，就是 n 维矩阵的抽象。因此一维的张量等价于一维的矩阵，二维的张量等价于二维矩阵。
- Session 会话是 tensorflow 里面的重要机制，tensorflow 构建的计算图必须通过 Session 会话才能执行

TensorFlow 的几个重要概念：

- 使用图 (graph) 来表示计算任务
- 在会话(seesion)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 可以为任意的操作赋值或者从其中获取数据

使用 Summary 和 TensorBoard 合作完成 TensorFlow 的可视化，TensorBoard 用于跟踪神经网络的整个训练过程中的信息，比如迭代的过程中每一层参数是如何变化与分布的，比如每次循环参数更新后模型在测试集与训练集上的准确率是如何的，等等，它将模型训练过程中的各种数据汇总起来存在自定义的路径与日志文件中，然后在指定的 web 端可视化地展现这些信息。

Tensorboard 可以记录与展示这些数据形式：

标量 Scalars；

图片 Images；

音频 Audio；

计算图 Graph；

数据分布 Distribution；

直方图 Histograms；

嵌入向量 Embeddings

5 项目过程与内容（例如：包括任务分析、数据分析、开发步骤、关键问题、实验结果分析等）

5.1 任务分析

MNIST 是一个入门级的计算机视觉数据集，它包含各种手写数字图片，范围是 0 到 9。我们需要建立一个机器学习模型，通过大量的训练，使得该模型能够预测 MNIST 图片里的数字。

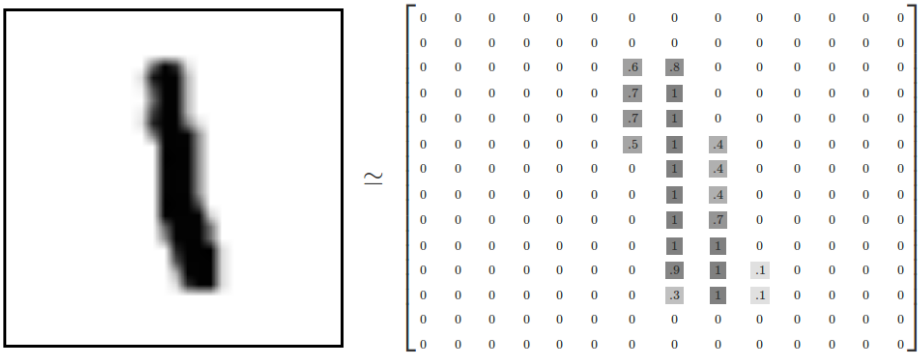
5.2 数据分析

数据集的获取：数据集被分成两部分，第一部分是 60000 行的训练数据集（mnist.train），另一部分是 10000 行的测试数据集（mnist.test），前者用于训练模型，后者用于测试，也就是评估这个模型的性能。MNIST 中的每一个数据单元由两部分组成：一张包含手写数字的图片和一个对应的标签，标签标示该图片所写的是哪个数字。我们从 MNIST 官网下载下来的数据集有 4 个文件，分别为：训练图片集、训练标签集、测试图片集、测试标签集。如下图：

```
train-images-idx3-ubyte.gz:  training set images (9912422 bytes)
train-labels-idx1-ubyte.gz:  training set labels (28881 bytes)
t10k-images-idx3-ubyte.gz:   test set images (1648877 bytes)
t10k-labels-idx1-ubyte.gz:   test set labels (4542 bytes)
```

这四个 gz 文件我们无法直接打开，需要在实现模型的代码中使用官方提供的一份 python 源代码来安装。

数据单元的格式分析：每一张图片包含 28 像素×28 像素，可以用一个数字数组来表示一张图片：



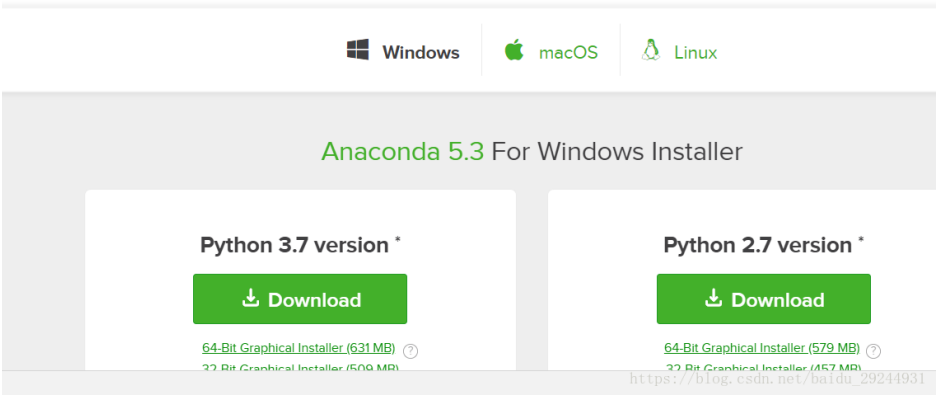
这个数组可以展开成一个向量，长度是 28×28＝784。可以理解成：MNIST 数据的图片就是在 784 维向量空间里面的点。所以在 MNIST 训练数据集中，mnist.train.images 是一个形状为 [60000, 784] 的张量，第一个维度数字用来索引图片，第二个维度数字用来索引每张图片中的像素点。MNIST 数据的标签是 0 到 9 的数字，使用“one-hot vectors”表示，一个 one-hot 向量除了某一位的数字是 1 以外其余各维度数字都是 0，数字 n 将表示成一个只有在第 n 维度（从 0 开始）数字为 1 的 10 维向量。比如，标签 0 将表示成([1,0,0,0,0,0,0,0,0,0])。因此， mnist.train.labels 是一个 [60000, 10] 的数字矩阵，与图片集相类似，第一个维度数字用来索引是哪张图片的标签，第二个维度数组表示该图片数字是几。

5.3 开发步骤

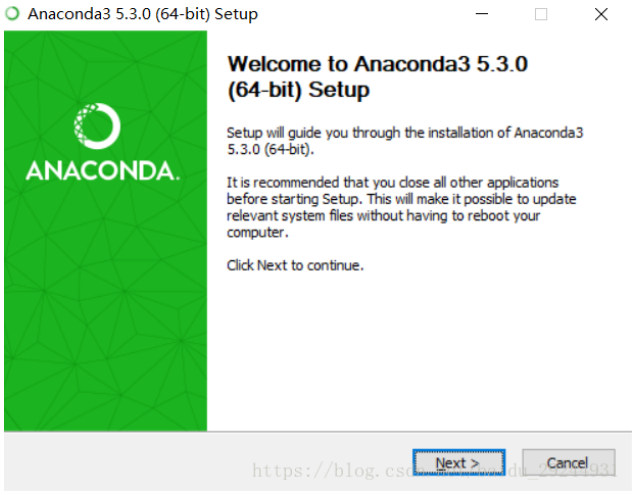
（一）Windows 下用 Anaconda 进行安装 Tensorflow:

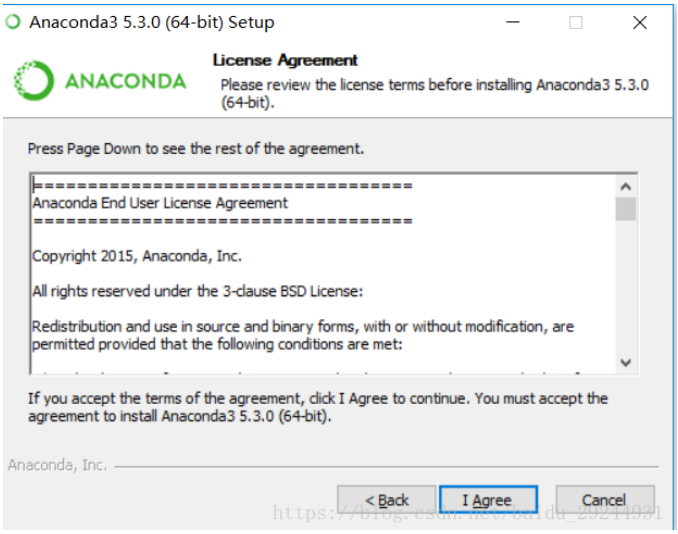
(1)安装 anaconda python：

下载：https://www.anaconda.com/download/#windows

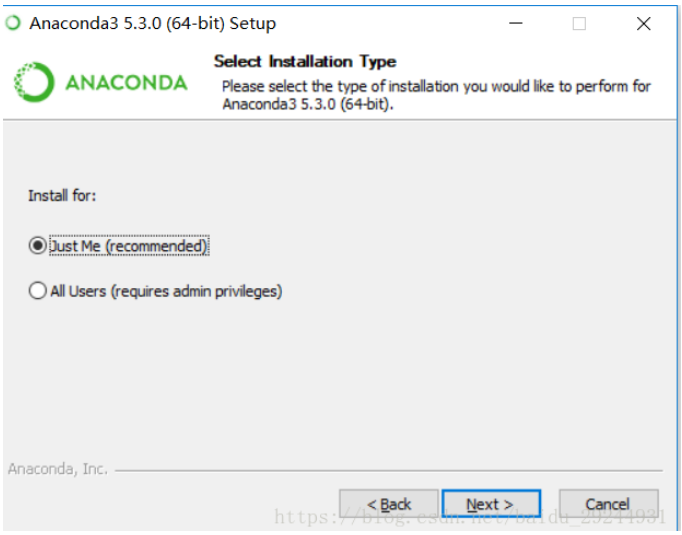


安装 anoconda（必须是 3.5 版本）：

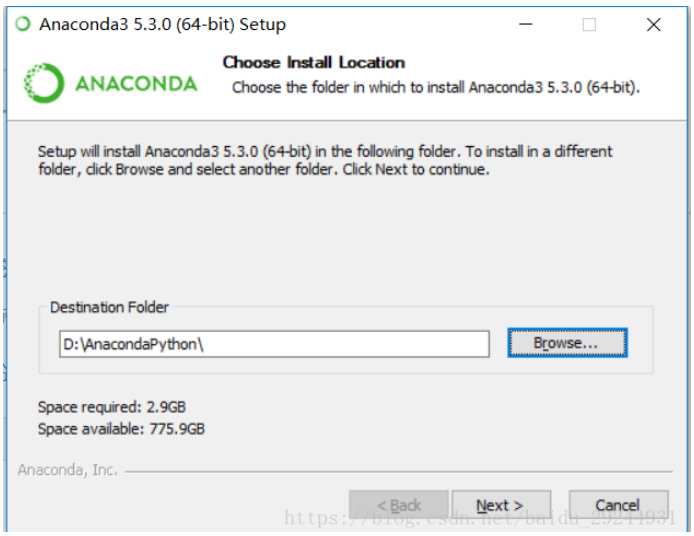




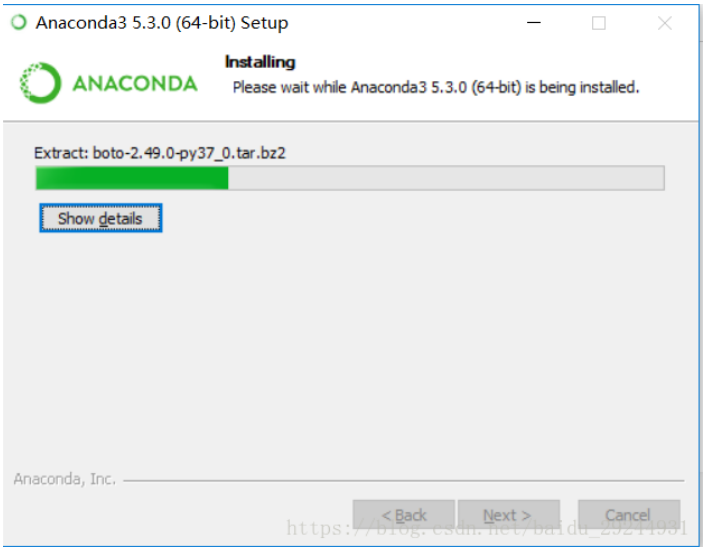
如果系统只有一个用户选择默认的第一个即可，如果有多个用户而且都要用到 Anaconda ，则选择第二个选项：



选择安装目录：



等待安装完成：



（2）环境变量配置：

在我的电脑右键-属性-高级系统设置-高级-环境变量中在 path 中加入你系统安装 anaconda 的目录下的 scripts 中

（3）安装 Tensorflow:

进入到 Anaconda3-envs 文件夹中通过调用以下命令创建名为 tensorflow 的 conda 环境：

```
conda create -n tensorflow pip python=3.5
```

通过发出以下命令激活 conda 环境：

```
activate tensorflow
```

发出相应命令以在 conda 环境中安装 TensorFlow。要安装仅支持 CPU 的 TensorFlow 版本，请输入以下命令：

```
(tensorflow)C:> pip install --ignore-installed --upgrade tensorflow
```

要安装 GPU 版本的 TensorFlow，请输入以下命令（在同一行）：

```
(tensorflow)C:>pip install --ignore-installed --upgrade tensorflow-gpu
```

（二）Linux 下用 Anaconda 进行安装 Tensorflow:

首先通过如下命令安装 pip 和 virtualenv：

```
$ sudo apt-get install python-pip python-dev python-virtualenv # for Python 2.7
$ sudo apt-get install python3-pip python3-dev python-virtualenv # for Python 3.n
```

使用如下命令创建 virtualenv 环境：

```
$ virtualenv --system-site-packages targetDirectory # for Python 2.7
$ virtualenv --system-site-packages -p python3 targetDirectory # for Python 3.n
```

激活 virtualenv 环境：

```
$ source ~/tensorflow/bin/activate # bash, sh, ksh, or zsh
$ source ~/tensorflow/bin/activate.csh  # csh or tcsh
```

在 virtualenv 环境下安装 TensorFlow ，如果已经将 TensorFlow 的源添加到系统则可以直接执行如下命令：

```
(tensorflow)$ pip install --upgrade tensorflow      # for Python 2.7
(tensorflow)$ pip3 install --upgrade tensorflow     # for Python 3.n
(tensorflow)$ pip install --upgrade tensorflow-gpu  # for Python 2.7 and GPU
(tensorflow)$ pip3 install --upgrade tensorflow-gpu # for Python 3.n and GPU
```

（三）Tensorboard 安装与使用

(1)tensoroard 安装：

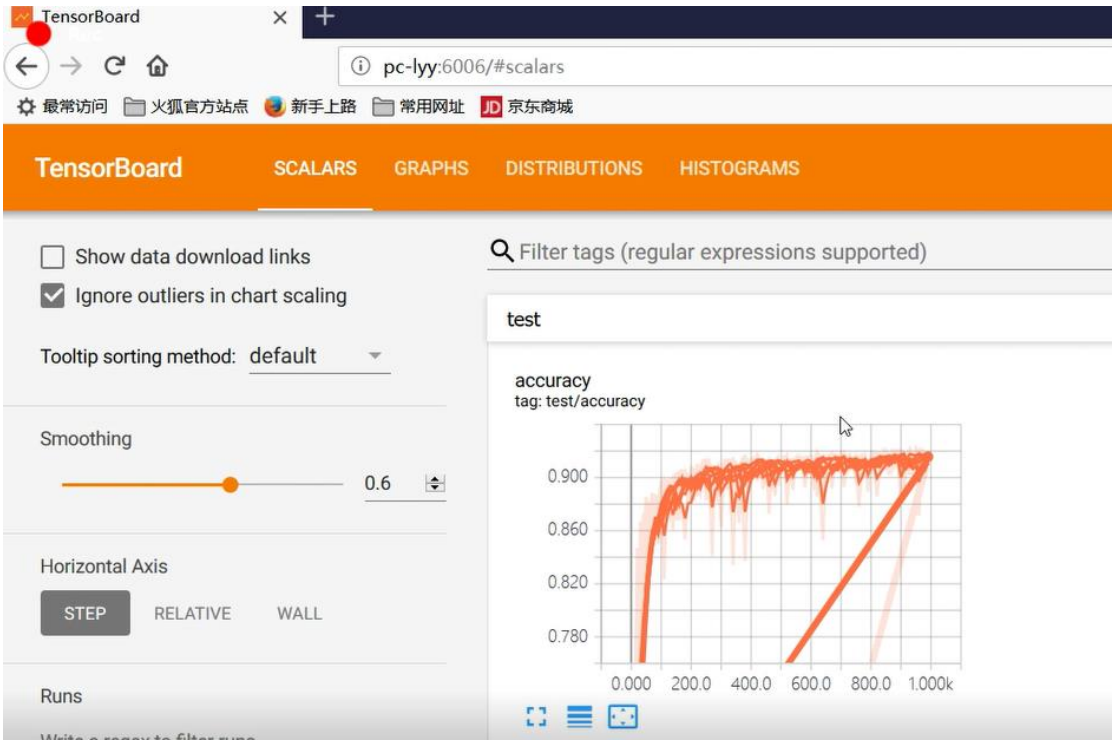
```
Pip install tensorboard
```

(2)tensorboard 使用：

logdir 为存放 log 文件的文件夹：

```
(tensorflow) d:\tmp>tensorboard --logdir=mnist_logs
TensorBoard 1.11.0 at http://PC-LYY:6006 (Press CTRL+C to quit)
W1018 23:47:23.753580 Reloader tf_logging.py:120] Found more than one graph event per run, or there was a metagraph containing a gr
aph_def, as well as one or more graph events.  Overwriting the graph with the newest event.
W1018 23:47:23.753580 15760 tf_logging.py:120] Found more than one graph event per run, or there was a metagraph containing a graph
 def, as well as one or more graph events.  Overwriting the graph with the newest event.
```

打开出现的 ip,就看见 tensorboard 界面:



5.4 关键代码及说明

1.导入数据并创建模型：

x（图片的特征值）：这里使用了一个 28*28=784 列的数据来表示一个图片的构成，784 个像素点对应 784 个数，因此输入层是 784 个神经元输出层是 10 个神经元；

W（特征值对应的权重）；

b（偏置量）；

使用 tf.Variable 创建一个变量，然后使用 tf.zeros 将变量 W 和 b 设为值全为 0 的张量（就是将张量中的向量维度值设定为 0）。由于我们在后续的过程中要使用大量的数据来训练 W 和 b 的值，因此他们的初始值是什么并不重要。

W 的形状是一个[784,10]的张量，第一个向量列表表示每个图片都有 784 个像素点，第二个向量列表表示从“0”到“9”一共有 10 类图片。所以 w 用一个 784 维度的向量表示像素值，用 10 维度的向量表示分类，而 2 个向量进行乘法运算（或者说 2 个向量的笛卡尔集）就表示“某个像素在某个分类下的证据”。

b 的形状是[10]，他仅表示 10 个分类的偏移值。

```
mnist = input_data.read_data_sets('Mnist_data/', one_hot=True,
fake_data=FLAGS.fake_data)

sess = tf.InteractiveSession()

x = tf.placeholder(tf.float32, [None, 784], name='x-input')
W = tf.Variable(tf.zeros([784, 10]), name='weights')
b = tf.Variable(tf.zeros([10], name='bias'))
```

2. tf.name_scope 主要是给表达式命名，给节点命名：

预测值 y，softmax 相当于 sigmoid 函数

```
with tf.name_scope('Wx_b'):
    y = tf.nn.softmax(tf.matmul(x, W) + b)
```

3. 在 tensorboard 中 histogram 模块中将变量可视化：分布图、直方图：

```
_ = tf.summary.histogram('weights', W)#可视化观看变量：添加任意 shape 的 Tensor，统计这个 Tensor 的取值分布。
_ = tf.summary.histogram('biases', b)#可视化观看变量：添加任意 shape 的 Tensor，统计这个 Tensor 的取值分布。
_ = tf.summary.histogram('y', y)#可视化观看变量：添加任意 shape 的 Tensor，统计这个 Tensor 的取值分布。
```

4. 定义偏差率和优化器：

y_（真实结果）：来自 MNIST 的训练集，每一个图片所对应的真实值，如果是 6，则表示为：[0 0 0 0 0 1 0 0 0]

```
y_ = tf.placeholder(tf.float32, [None, 10], name='y-input')
```

5. 交叉熵评估代价以及梯度下降算法：

```
with tf.name_scope('xent'):
    #交叉熵评估代价
    cross_entropy = -tf.reduce_sum(y_ * tf.log(y))
    _ = tf.summary.scalar('cross entropy', cross_entropy)
  with tf.name_scope('train'):
     #梯度下降算法的一种
    train_step = tf.train.GradientDescentOptimizer(
        FLAGS.learning_rate).minimize(cross_entropy)

  with tf.name_scope('test'):
    correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
    _ = tf.summary.scalar('accuracy', accuracy)

  # Merge all the summaries and write them out to /tmp/mnist_logs 合并所有的摘要并将它们写到/tMP/Mistist 日志中
  merged = tf.summary.merge_all()
  writer = tf.summary.FileWriter('C:/Users/YS/Documents/course/mnist-master/mnist-master/mnist-master/mnist_logs', sess.graph_def)
  tf.initialize_all_variables().run()
```

6. 训练模型，并输入试验数据，每 10 步记录总结，初始化变量，开始迭代执行。每隔 10 次，使用测试集验证一次，sess.run()的输入，merged 合并的 Log 信息，accuracy 计算图，feed 数据，将信息写入 test_writer。每隔 99 步，将运行时间与内存信息，存入 Log 中，其余步骤正常秩序，添加存储信息。

```
for i in range(FLAGS.max_steps):        #训练阶段，迭代最大次数
    if i % 10 == 0:  #每训练 100 次，测试一次
        #损失函数（交叉熵）和梯度下降算法，通过不断的调整权重和偏置量的值，
        #来逐步减小根据计算的预测结果和提供的真实结果之间的差异，以达到训练
模型的目的。
        if FLAGS.fake_data:
          #fill_feed_dict 函数会查询给定的 DataSet，索要下一批次 batch_size 的图像和
标签，
          #与占位符相匹配的 Tensor 则会包含下一批次的图像和标签。
          batch_xs, batch_ys = mnist.train.next_batch(
              100, fake_data=FLAGS.fake_data)      #按批次训练，每批 100 行数据
          feed = {x: batch_xs, y_: batch_ys}  # 用训练数据替代占位符来执行训练
        else:
          feed = {x: mnist.test.images, y_: mnist.test.labels} # 用测试数据替代占位符来
执行训练
        result = sess.run([merged, accuracy], feed_dict=feed)
        summary_str = result[0]
        acc = result[1]
        writer.add_summary(summary_str, i)
        print('Accuracy at step %s: %s' % (i, acc))
    else:
        batch_xs, batch_ys = mnist.train.next_batch(
            100, fake_data=FLAGS.fake_data)
        feed = {x: batch_xs, y_: batch_ys}
        sess.run(train_step, feed_dict=feed)

if __name__ == '__main__':
  tf.app.run()
```

5.5 实验结果分析

本部分通过运行过程进行实验结果分析

首先运行代码进行训练和测试，即下面一行代码

```
guyun@guyun-virtual-machine:~/桌面/tensorflow/mnist-master/mnist-master$ python
 mnist_with_summaries.py
```

得到运行结果如下：

```
Accuracy at step 40: 0.8415
Accuracy at step 50: 0.8673
Accuracy at step 60: 0.8689
Accuracy at step 70: 0.8655
Accuracy at step 80: 0.8759
Accuracy at step 90: 0.89
Accuracy at step 100: 0.879
Accuracy at step 110: 0.8752
Accuracy at step 120: 0.8899
Accuracy at step 130: 0.8995
```

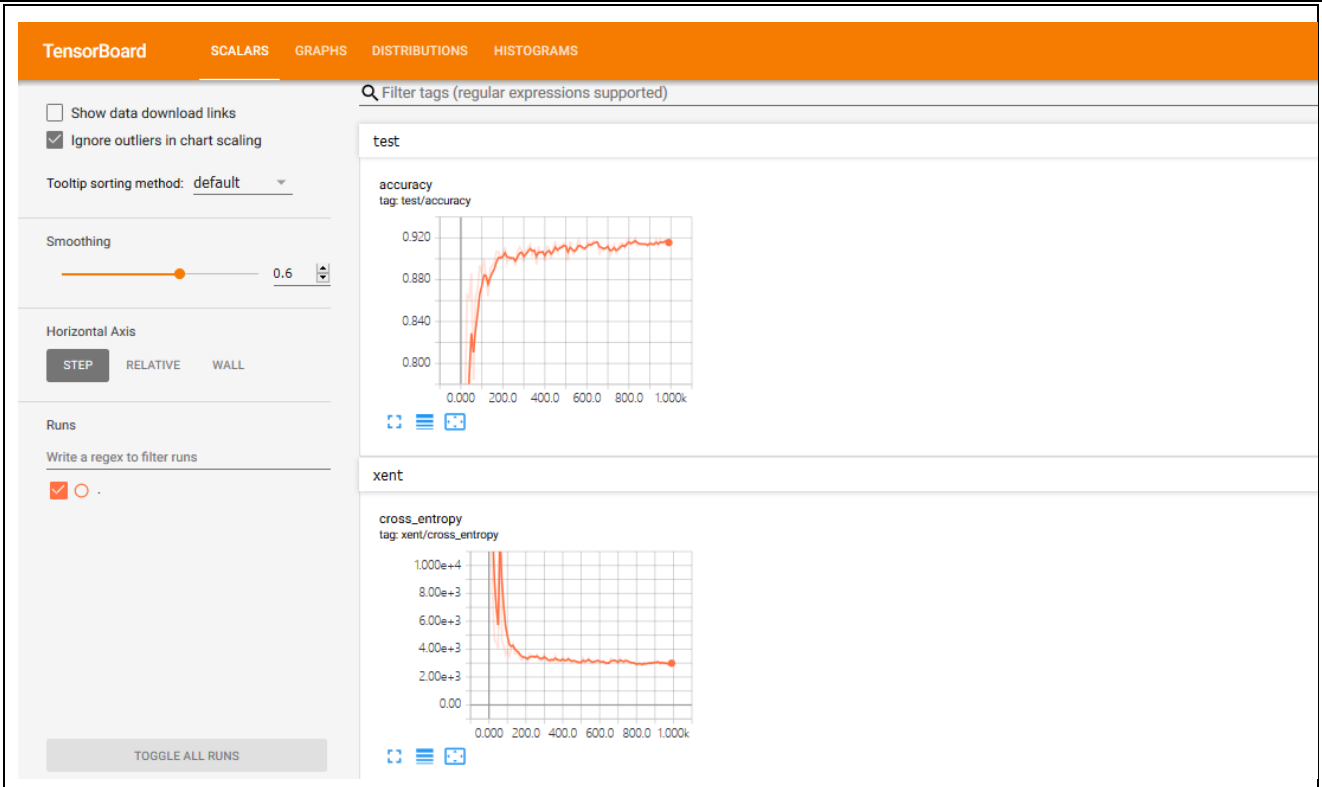
```
Accuracy at step 790: 0.9172
Accuracy at step 800: 0.8928
Accuracy at step 810: 0.9109
Accuracy at step 820: 0.9045
Accuracy at step 830: 0.9149
Accuracy at step 840: 0.8958
Accuracy at step 850: 0.9165
Accuracy at step 860: 0.9131
Accuracy at step 870: 0.8833
Accuracy at step 880: 0.9172
Accuracy at step 890: 0.9199
Accuracy at step 900: 0.9187
Accuracy at step 910: 0.9153
Accuracy at step 920: 0.9166
Accuracy at step 930: 0.9184
Accuracy at step 940: 0.92
Accuracy at step 950: 0.92
Accuracy at step 960: 0.9159
Accuracy at step 970: 0.8968
Accuracy at step 980: 0.9187
Accuracy at step 990: 0.9183
```

上述实验结果表述一下信息，初始化变量，开始迭代执行。每训练 10 次，使用测试集验证一次并输出测试信息。根据实验结果我们得到结论，训练后测试集的准确率逐渐增加，但是在准确率增加的过程中也会存在震荡，这说明测试的准确率受到训练集好坏的影响。

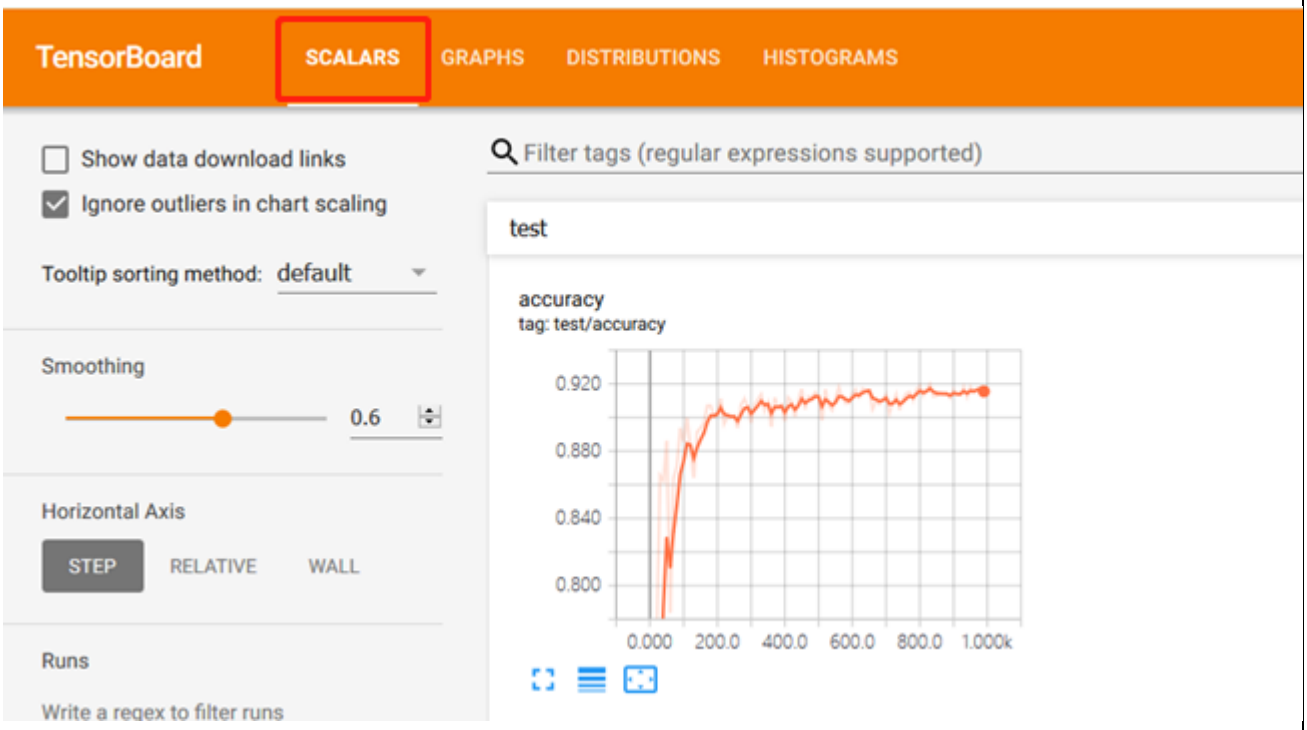
本实验中我们使用 TensorBoard 来展现我们的 TensorFlow 图像，绘制图像生成的定量指标图以及附加数据。这里的参数 logdir 指向 SummaryWriter 序列化数据的存储路径。通过代码 tensorboard –logdir=mnist_logs

```
guyun@guyun-virtual-machine:/$ cd tmp
guyun@guyun-virtual-machine:/tmp$ tensorboard --logdir=mnist_logs
TensorBoard 1.11.0 at http://guyun-virtual-machine:6006 (Press CTRL+C to quit)
^Cguyun@guyun-virtual-machine:/tmp$
```

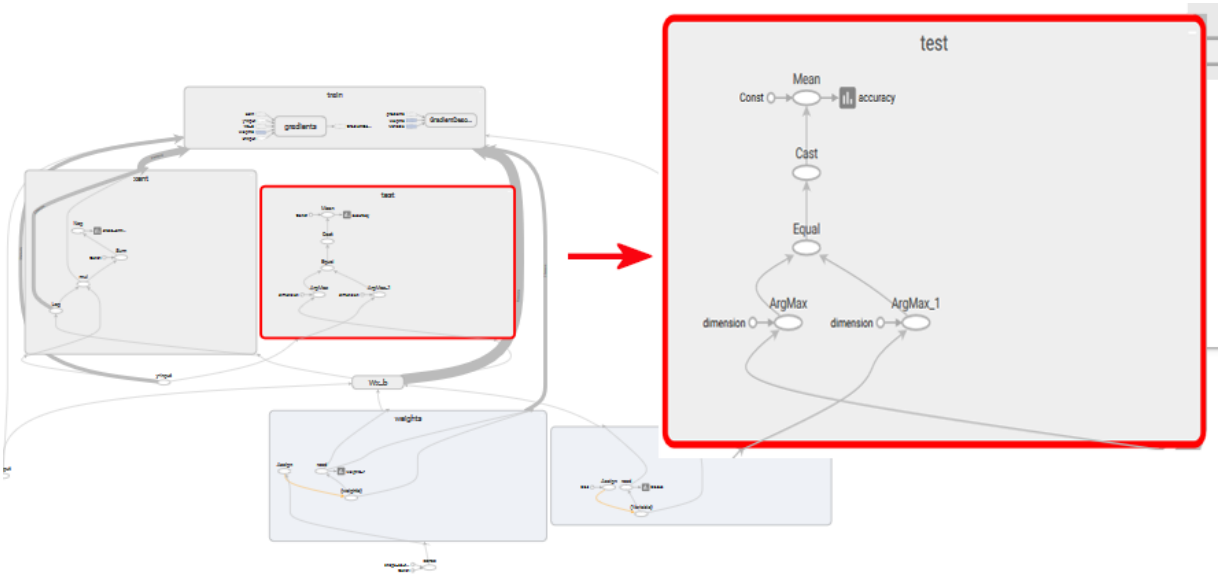
打开火狐浏览器，输入得到的网址即可打开 tensorboard 界面



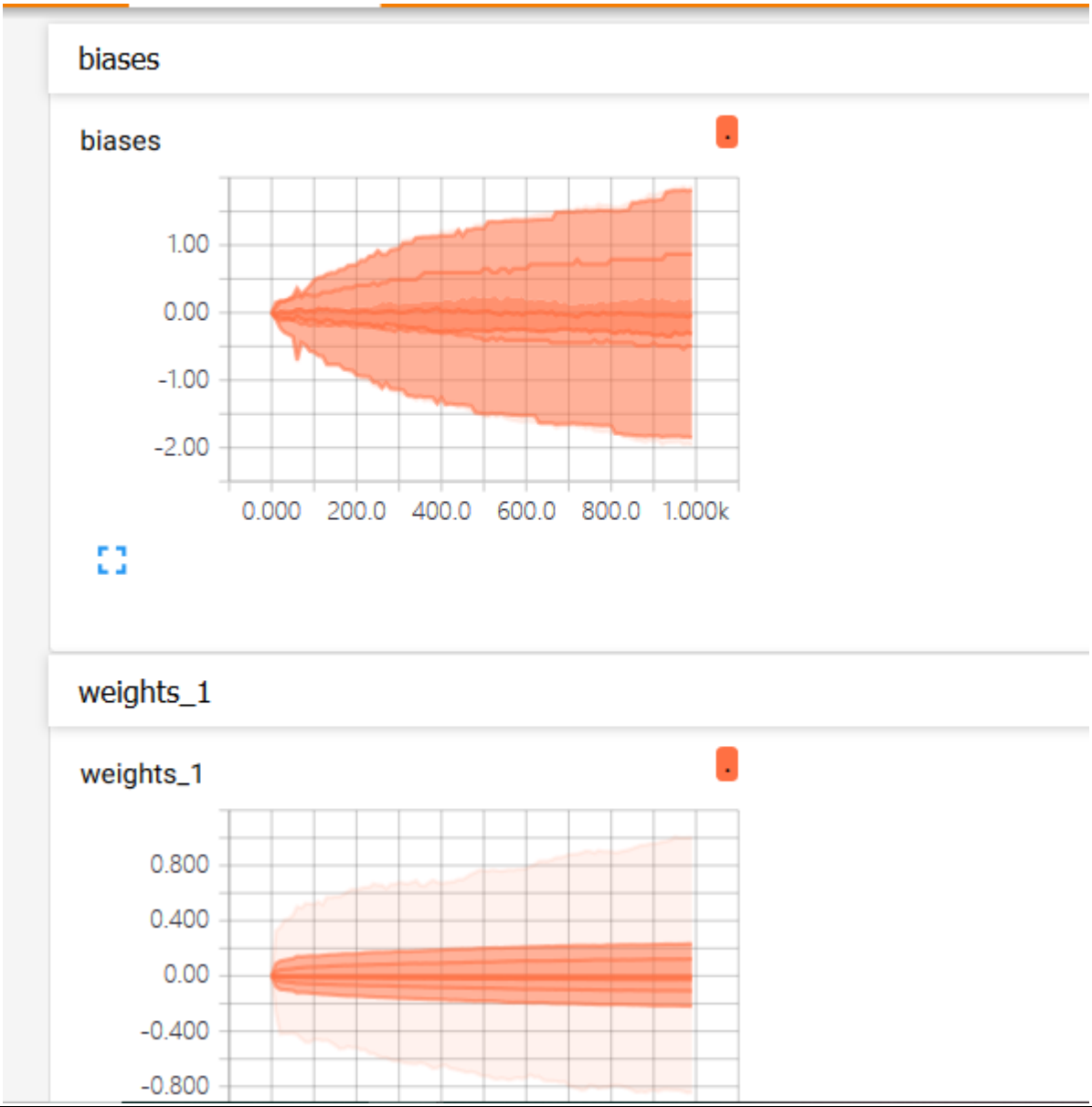
首先选择 SCALARS 查看标量的变化情况，如下图，测试集的准确率变化状况



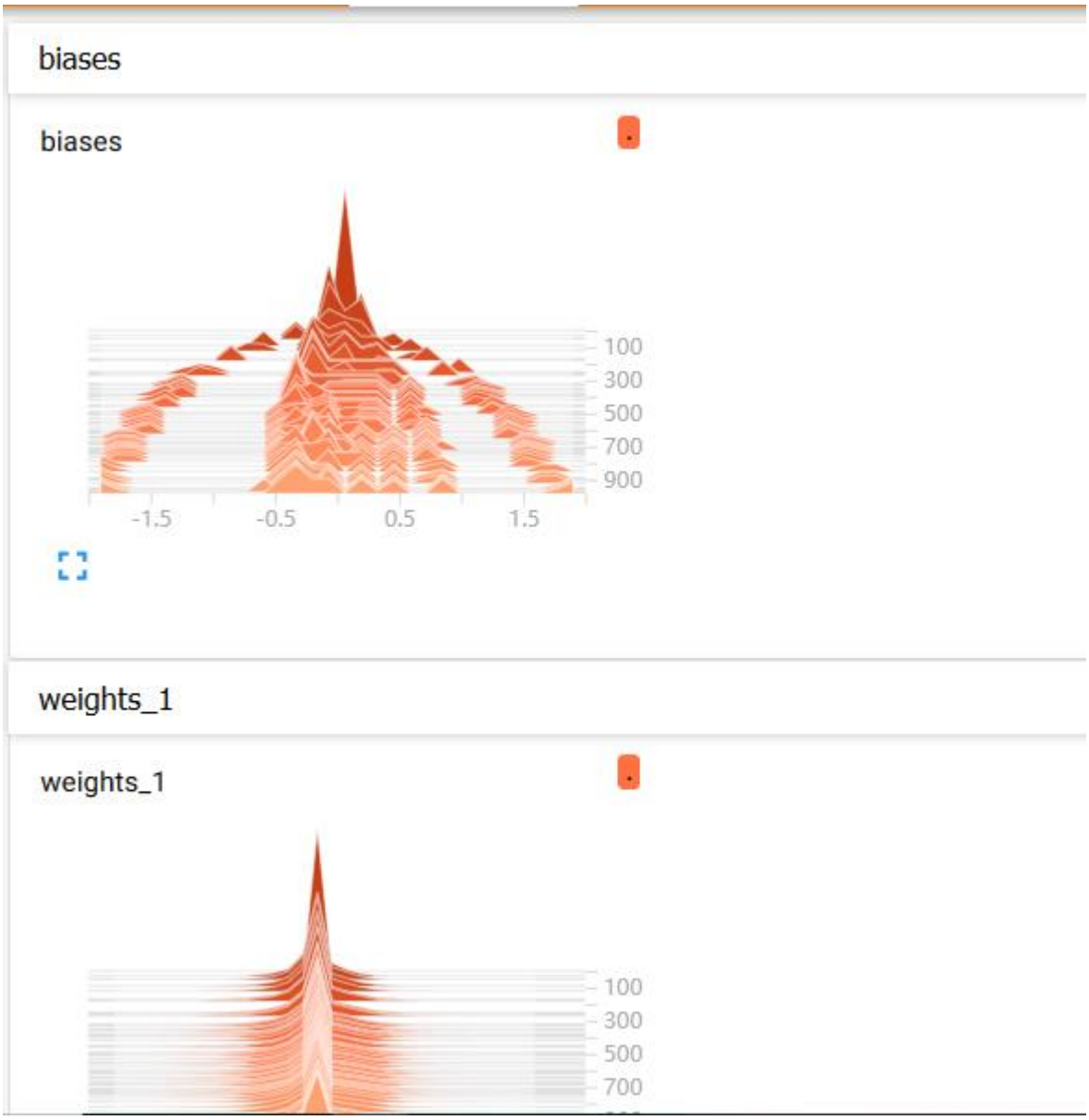
选择 GRAPHS 查看数据流图如下：



打开 DISTRIBUTIONS 查看 biases 和 weights_1 的变化状况



打开 HISTOGRAMS 查看 biases 和 weights_1 的直方图变化状况



6 项目总结与心得体会

这次实验让我踏入了机器学习的大门，我们的题目是“深度学习+TensorFlow+分类预测”，起初我们对深度学习完全不了解，通过查阅资料、阅读各种博客和教程，弄清了何为神经网络，深度学习与机器学习的关系、和神经网络的关系，深度学习是机器学习的一个子领域，通常和大的深层神经网络联系在一起。尽管最后的代码实现并没有采用深度神经网络，但是本次实验为后续学习 CNN 算法奠定了基础，搜索的过程中也积累了宝贵的经验。对我而言，本次实验的难点除了模型原理、数学原理，还有对 TensorFlow 的安装、学习和 python 中其他深度学习库的安装和使用。一开始，我根据 TensorFlow 官网的教程，在 ubuntu 虚拟机中

进行尝试，但是在安装一些依赖运行库的时候出现问题，解决之后又在安装深度学习库时失败，经过排查和搜索教程之后得以解决，非常锻炼动手能力。

本次实验还让我体会到团队协作的重要性，多个人的力量总是比单打独斗要强许多，职责分工使我们高效搜索资料，在理解知识的过程中，多人讨论使我们思维碰撞，得到意想不到的收获。

7 参考文献

[1] 百度百科 https://baike.baidu.com/item/TensorFlow/18828108?fr=aladdin

[2] 百度百科：https://baike.baidu.com/item/BP%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C/4581827?fr=aladdin

[3] 刘居贤.深度学习与人工神经网络[J].科技经济导刊,2018(31):7-9.

[4] 张荣,李伟平,莫同.深度学习研究综述[J].信息与控制,2018,47(04):385-397+410.

[5] 范铭豪.TensorFlow 中不同神经网络模型对 MNIST 数据集影响研究[J].现代信息科技,2018,2(11):75-77.

[6] MNIST 机器学习入门：http://www.tensorfly.cn/tfdoc/tutorials/mnist_beginners.html

[7] TensorFlow 教程（MNIST 手写体数字识别）：https://www.jianshu.com/p/db2afc0b0334

[8] #TensorFlow 基本使用 https://blog.csdn.net/yhl_leo/article/details/50619029

[9] #Tensorflow MNIST 数据集测试代码入门 https://blog.csdn.net/yhl_leo/article/details/50614444

[10]#Tensorflow 之 MNIST 解析 https://blog.csdn.net/qq546542233/article/details/77836328?utm_source=blogxgwz1

[11]#TensorFlow 学习笔记（十二）TensorFLow tensorBoard 总结 https://blog.csdn.net/qq_36330643/article/details/76709531#commentBox

[12]#Tensorflow 的应用（四）https://blog.csdn.net/yyxyyx10/article/details/78695531