人工智能 知识分享

--------以TensorFlow为例

景宁波

2016年11月07日

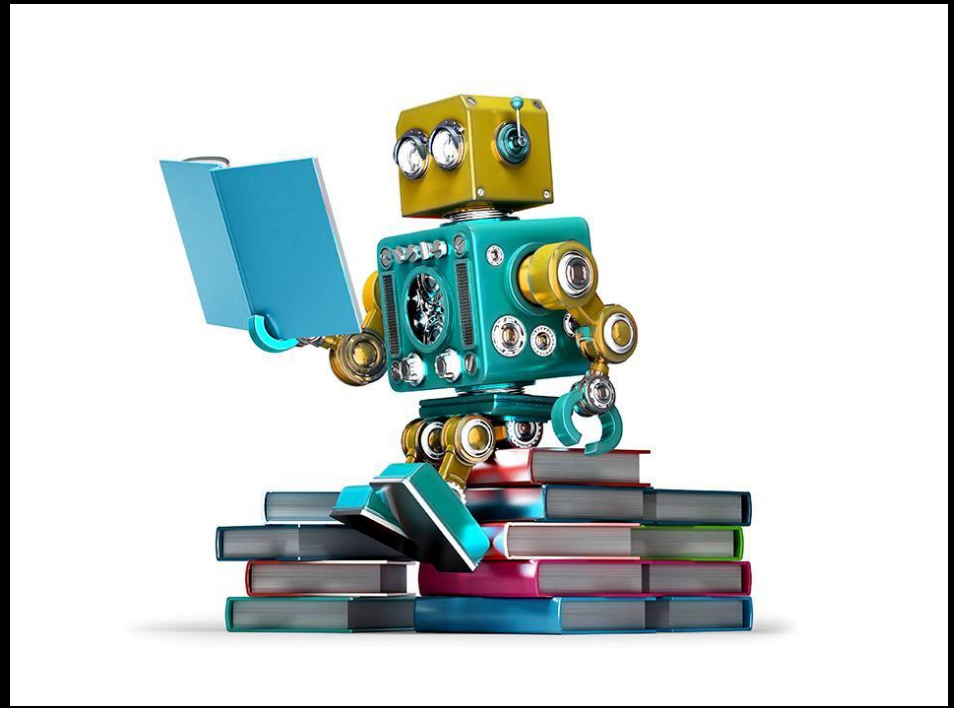
什么是人工智能（Artificial intelligence）

用**计算机**对人的意识、思维的**信息加工过程**的模拟。

人工智能不是人的智能，但可能**超过**人的智能

什么是机器学习（Machine Learning）

用计算机模拟人的**学习行为**以获取新的知识或技能，重新组织已有的知识结构使之不断改善自身的性能的学科。

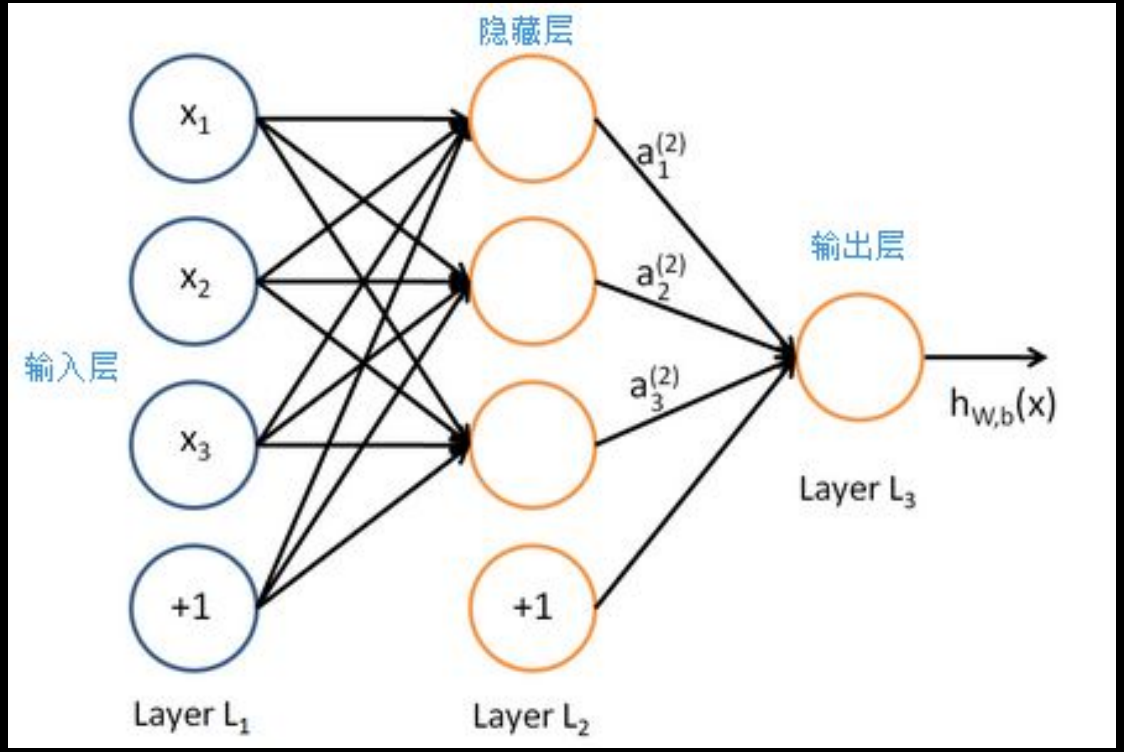


什么是深度学习（Deep Learning）

一种机器学习的方法，采用**多层 深度图 模型**对数据建模，形成数据的高级抽象。

程序员理解：

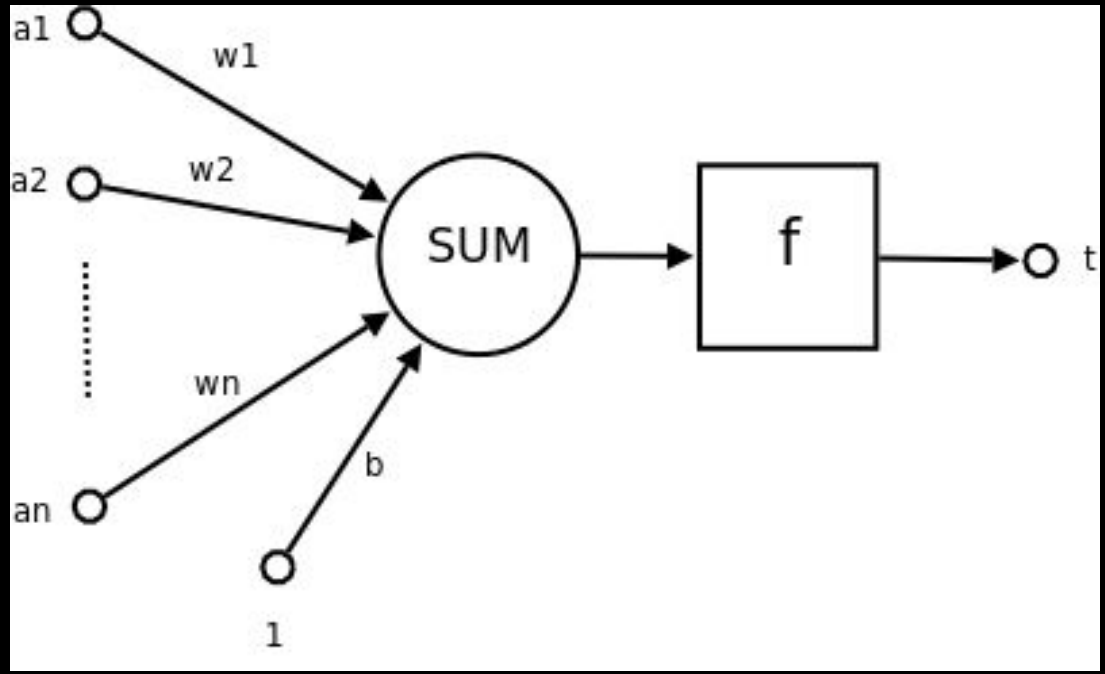
深度学习=多层神经网络



什么是神经网络(Artificial Neural Network)

ANN或NN，模仿生物神经网络的结构和功能的计算模型，用于对函数进行估计或近似。
求得输入向量与权向量的内积后，经一个非线性激励函数得到一个标量结果。

a1~an为输入分量
w1~wn为权分量 b为偏置
f为激励函数 t为神经元输出



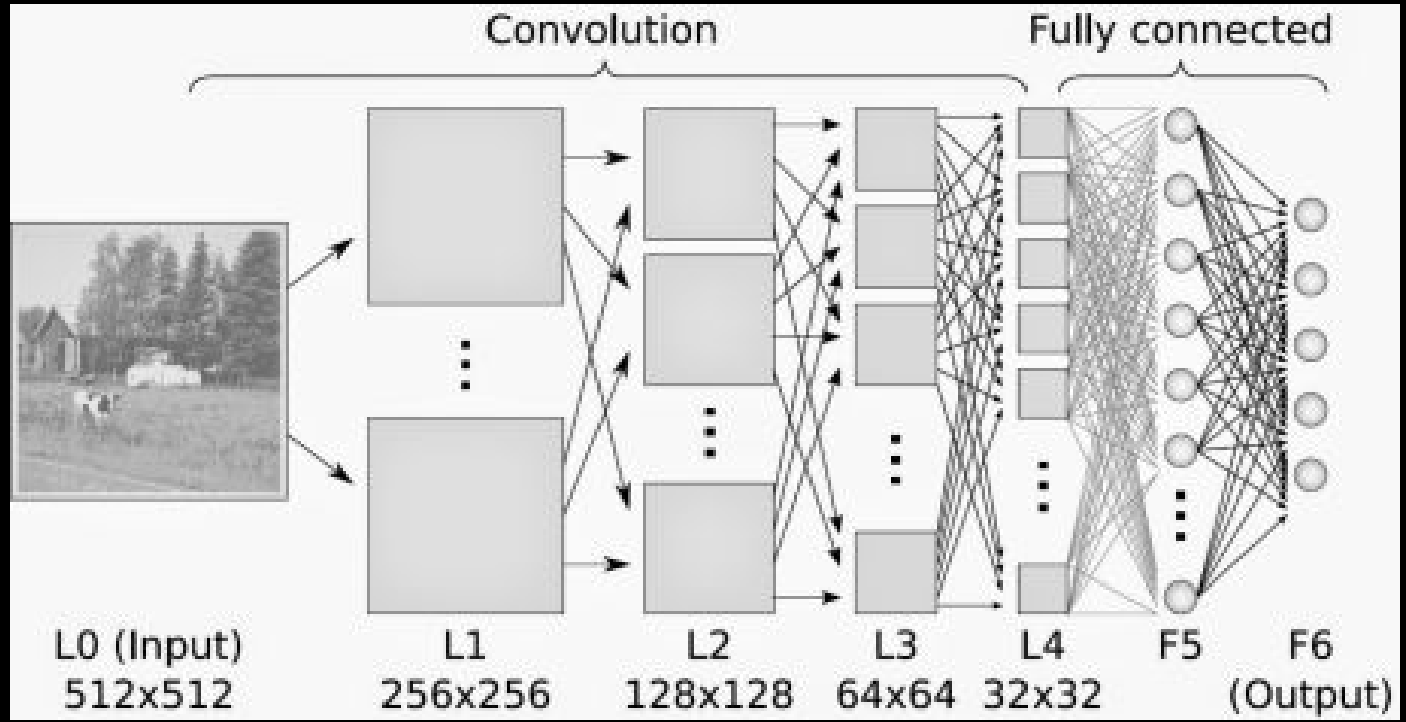
数学表示 $t = f(\vec{W'}\vec{A} + b)$

什么是卷积神经网络(CNN)

Convolutional neural networks

一种深度学习的实现方法

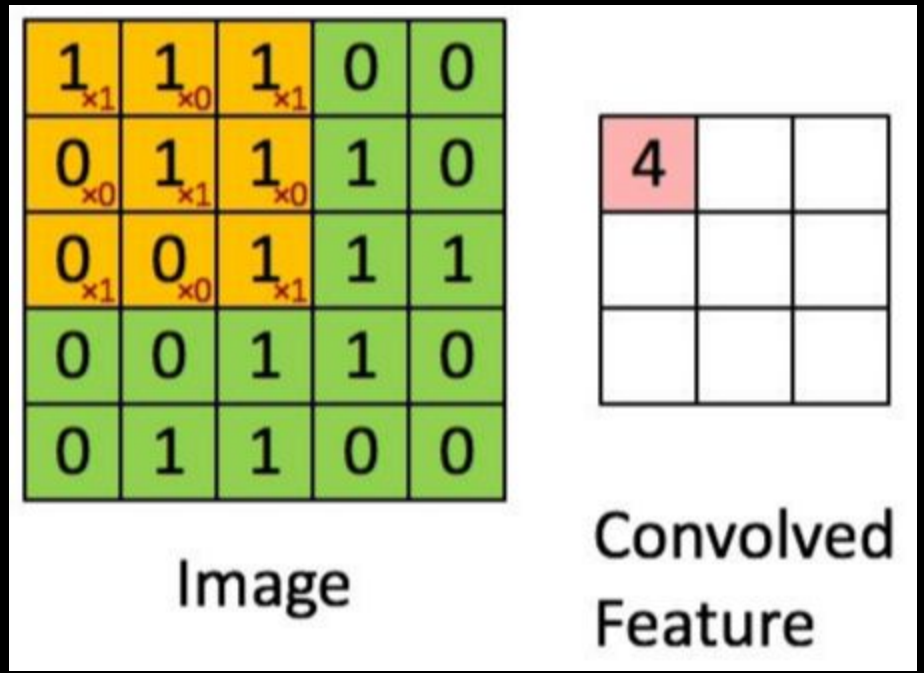
多层神经网络前增加多层卷积池化操作。



什么是卷积操作（Convolutio）

图像属性具有stationary

Feature extraction using convolution

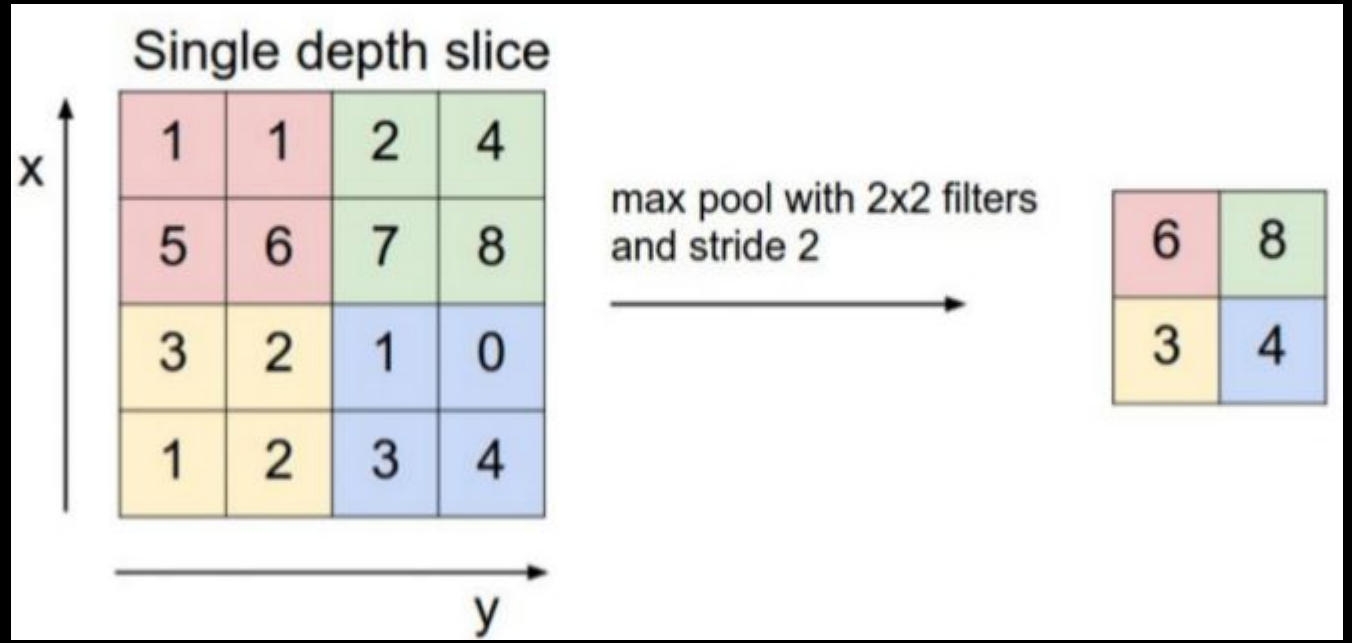


什么是池化操作(Pooling)

Convolutional neural networks

一种深度学习的实现方法

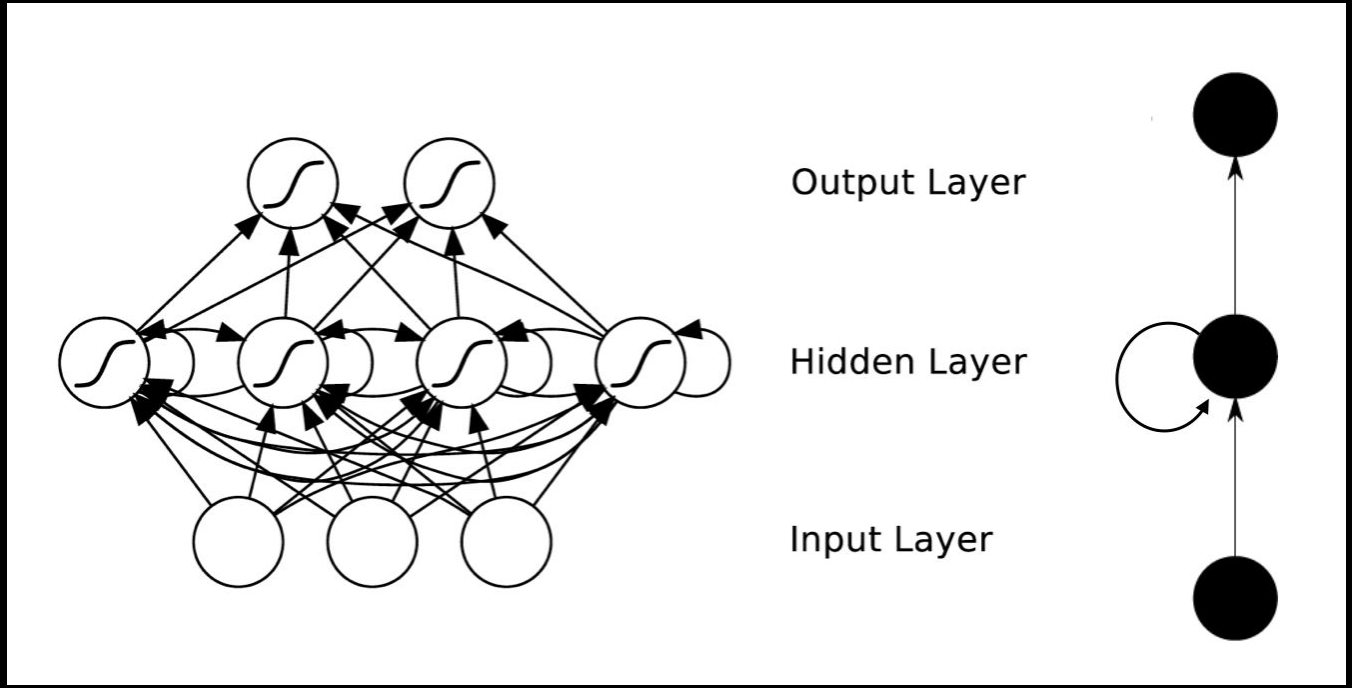
多层神经网络前增加多层卷积池化操作。



什么是循环神经网络(RNN)

Recurrent Neural Networks
是一种深度学习的实现方法

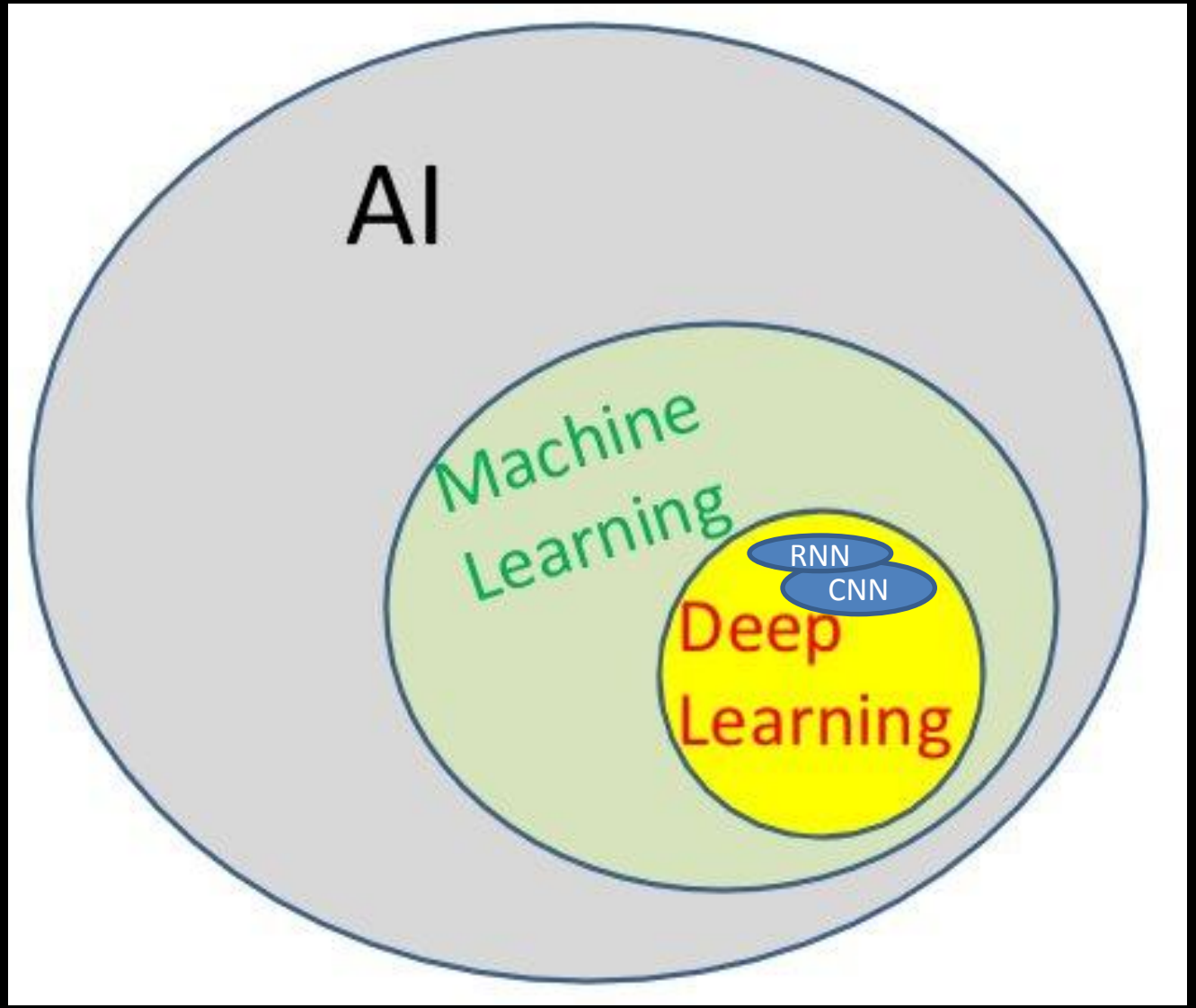
处理那些输入之间前后关联的问题。



关系

AI＞ML＞DL＞CNN

AI　人工智能 1950
ML 机器学习　1980
DL　深度学习　2010
CNN　卷积神经网络
RNN　循环神经网络

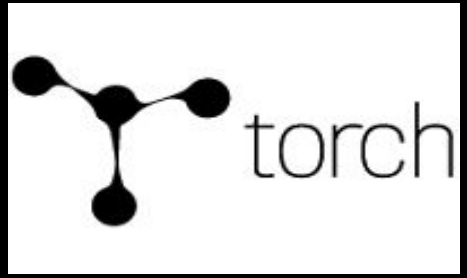


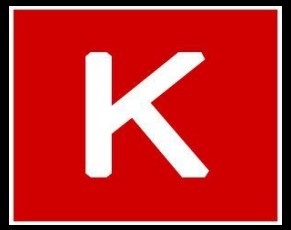
开源人工智能系统

Why open-source?

智能时代已经到来

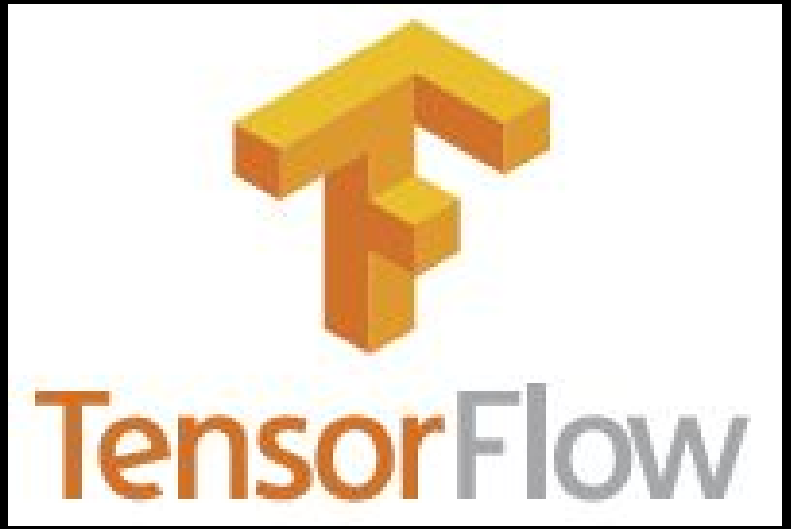
Data ＞＞ Software





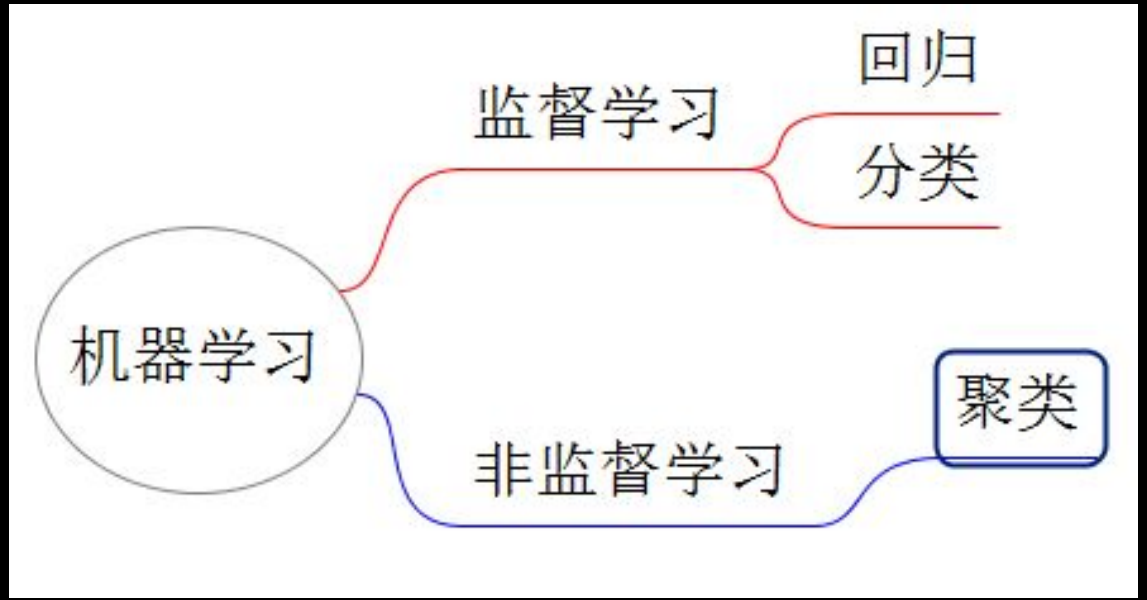
什么是Tensor Flow

- 神经网络结构的数值计算库
- Google开发
- Tensor + Flow



Tensor Flow能干啥

- **Any** machine neural network problem
  - 分类
  - 回归
  - 聚类



回归-code 1/5

第一步：生成或者导入 训练数据

```
x_data = np.linspace(-1, 1, 300)[:, np.newaxis]

noise = np.random.normal(0, 0.05, x_data.shape)

y_data = np.square(x_data) - 0.5 + noise
```

回归-code 2/5

第二步：搭建神经网络

```
#添加隐含层,
l1 = add_layer(xs, 1, 10, activation_function=tf.nn.relu)

# 添加输出层
prediction = add_layer(l1, 10, 1, activation_function=None)
```

回归-code 3/5

第三步 : loss函数 优化算法

```
loss =
tf.reduce_mean(tf.reduce_sum(tf.square(ys-prediction)
, reduction_indices=[1]))


train_step =
tf.train.GradientDescentOptimizer(0.1).minimize(loss)
```

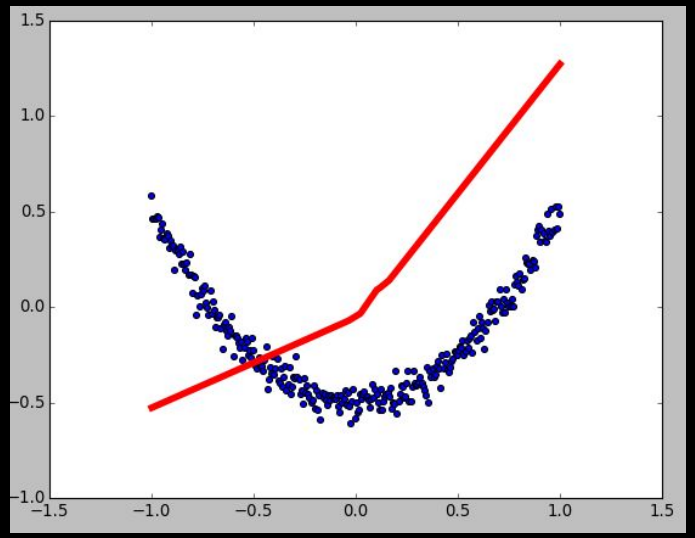
回归-code 4/5

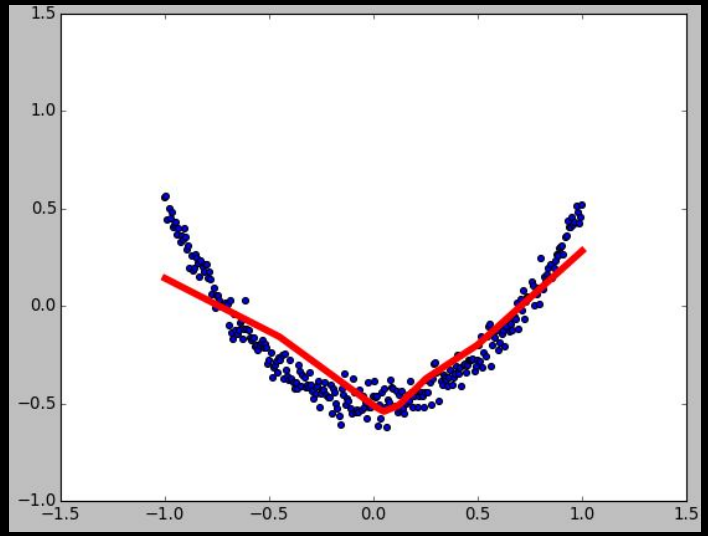
第四步：开始训练

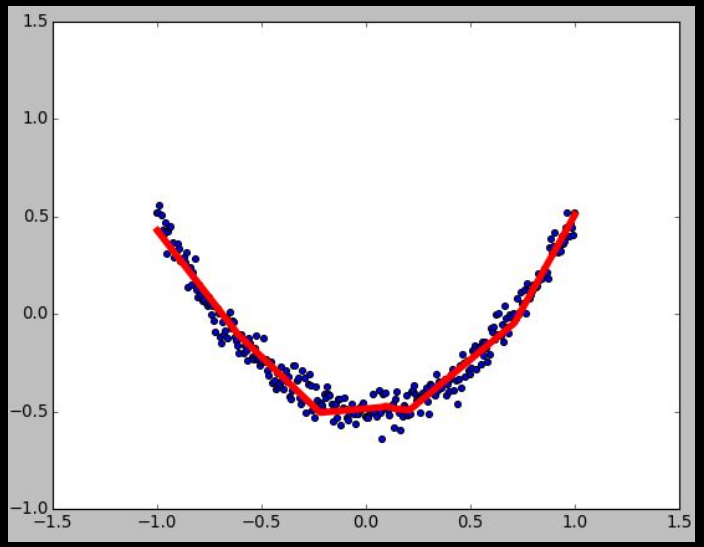
```
for i in range(1000):
    sess.run(train_step, feed_dict={xs: x_data, ys: y_data})
```

回归-code 5/5

第五步：输出训练结果

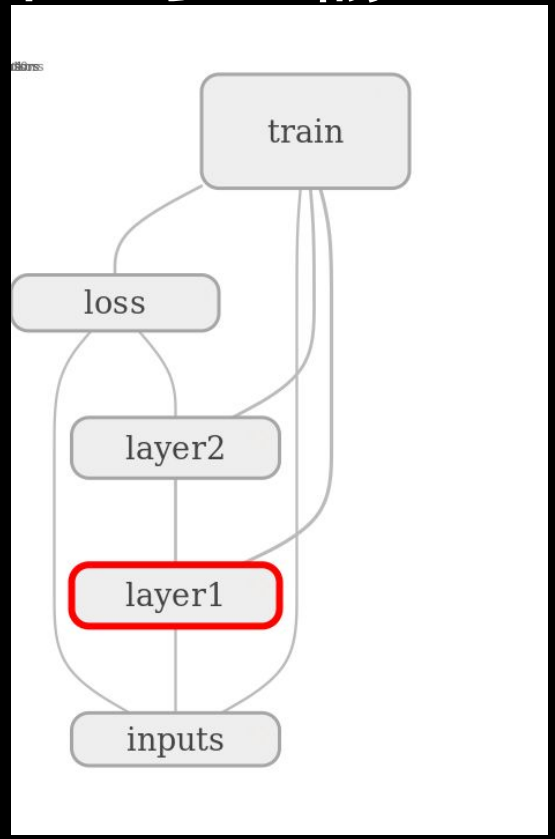


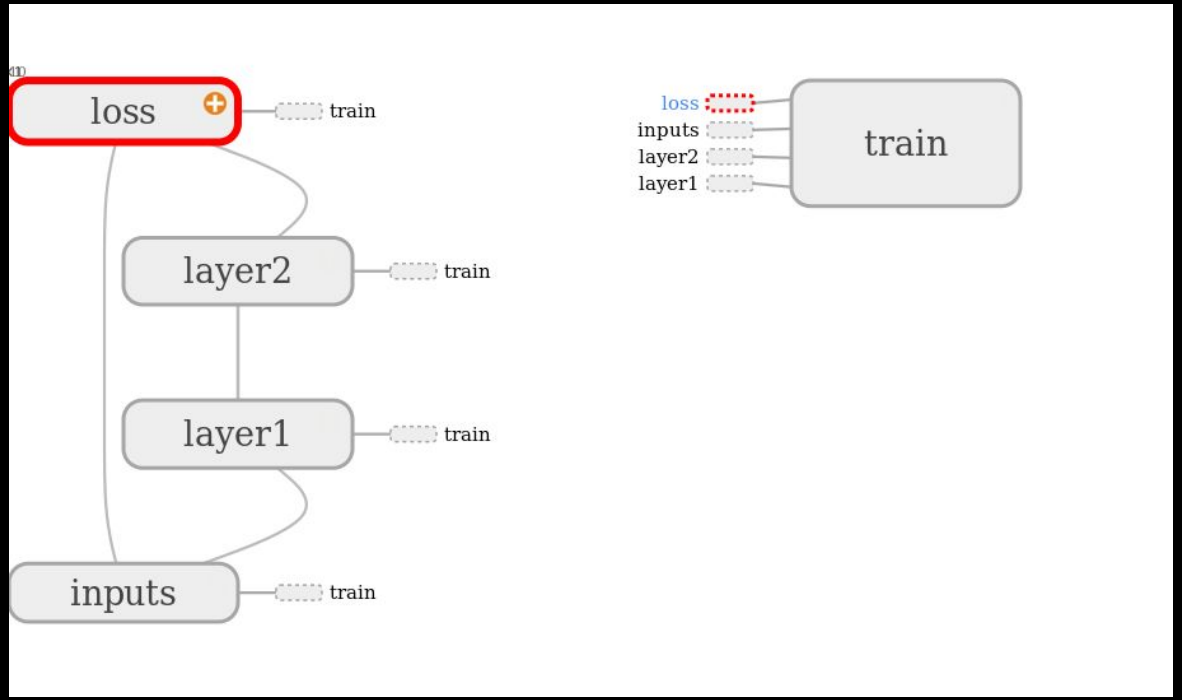




回归-code 5/5

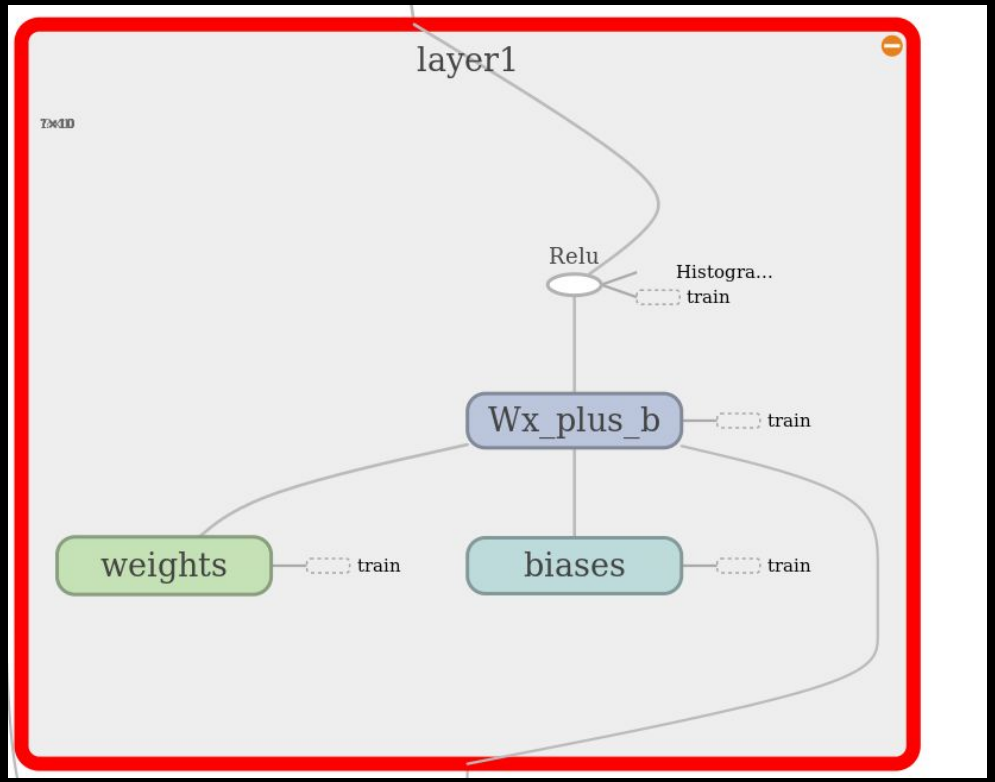
第五步：输出训练结果

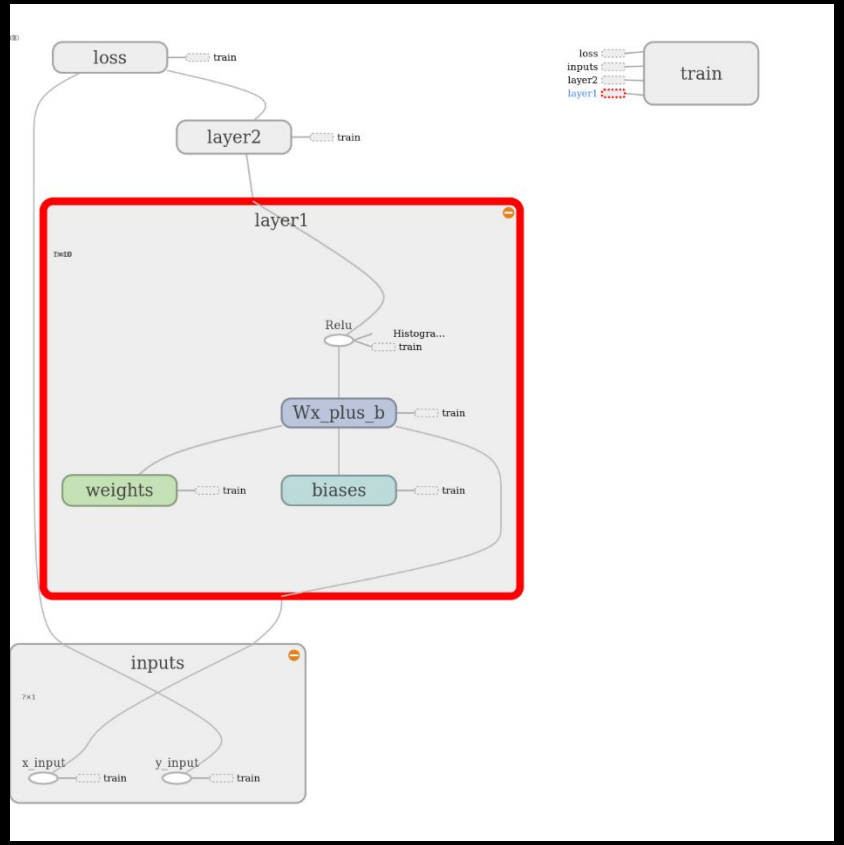




回归-code 5/5

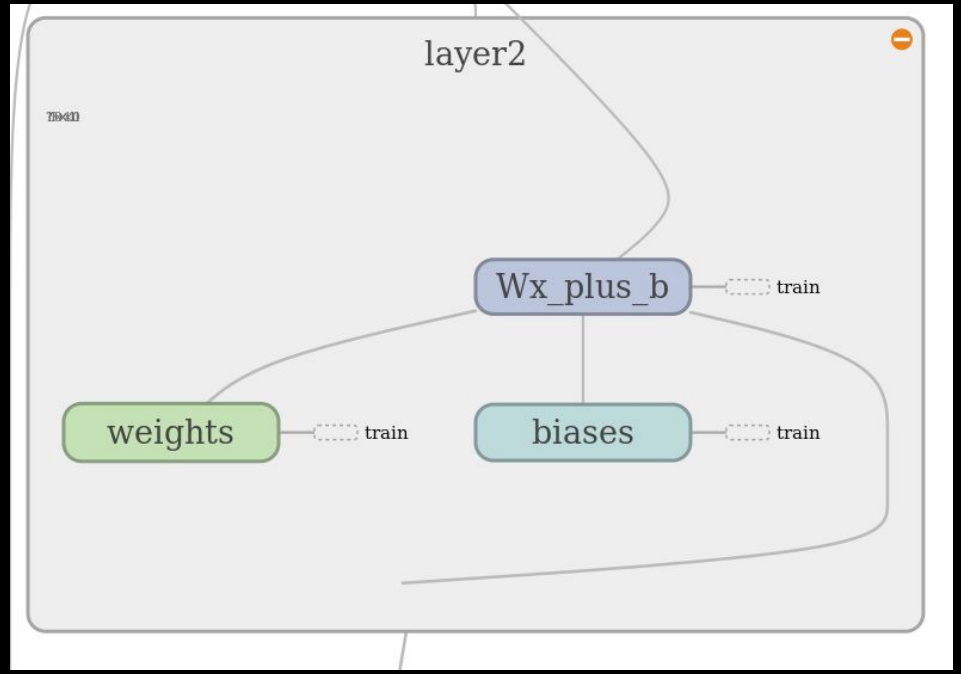
第五步：输出训练结果

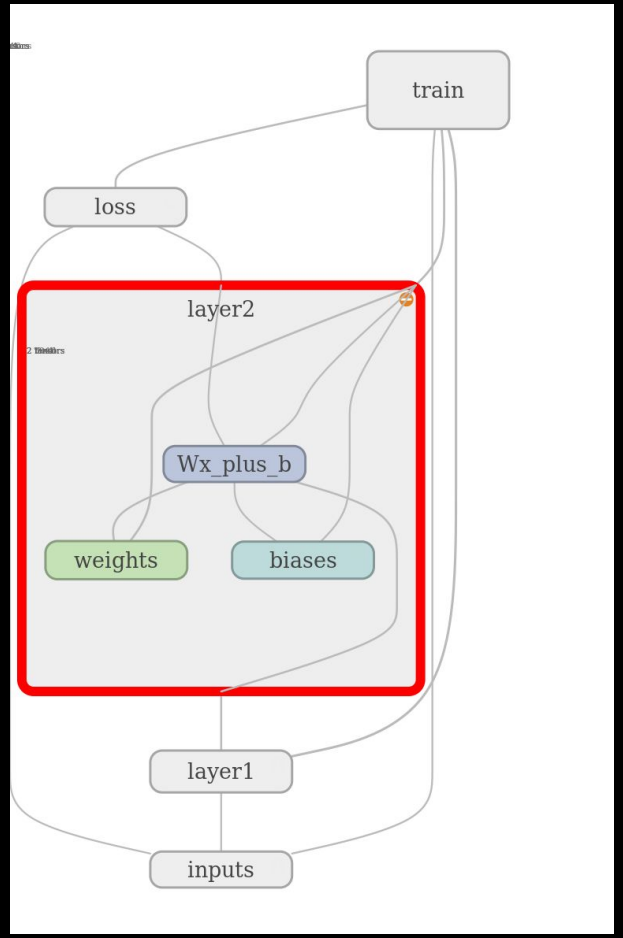




回归-code 5/5

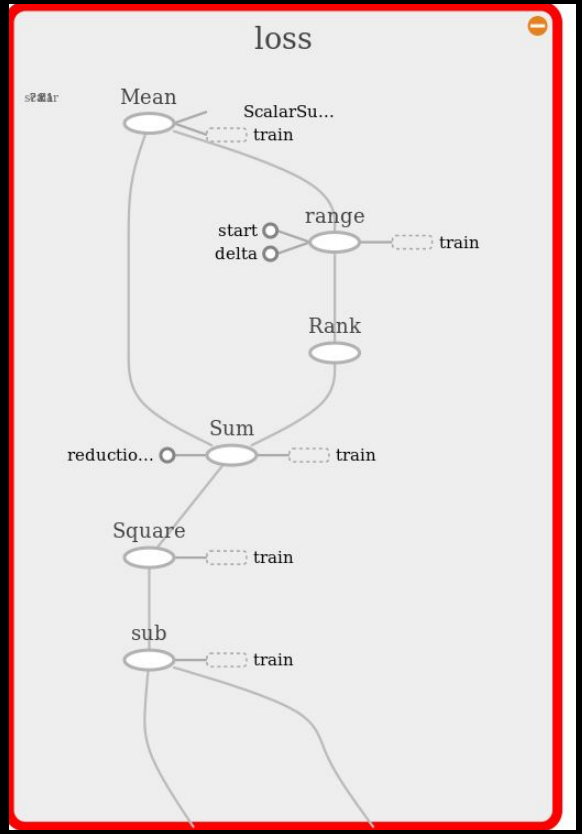
第五步：输出训练结果

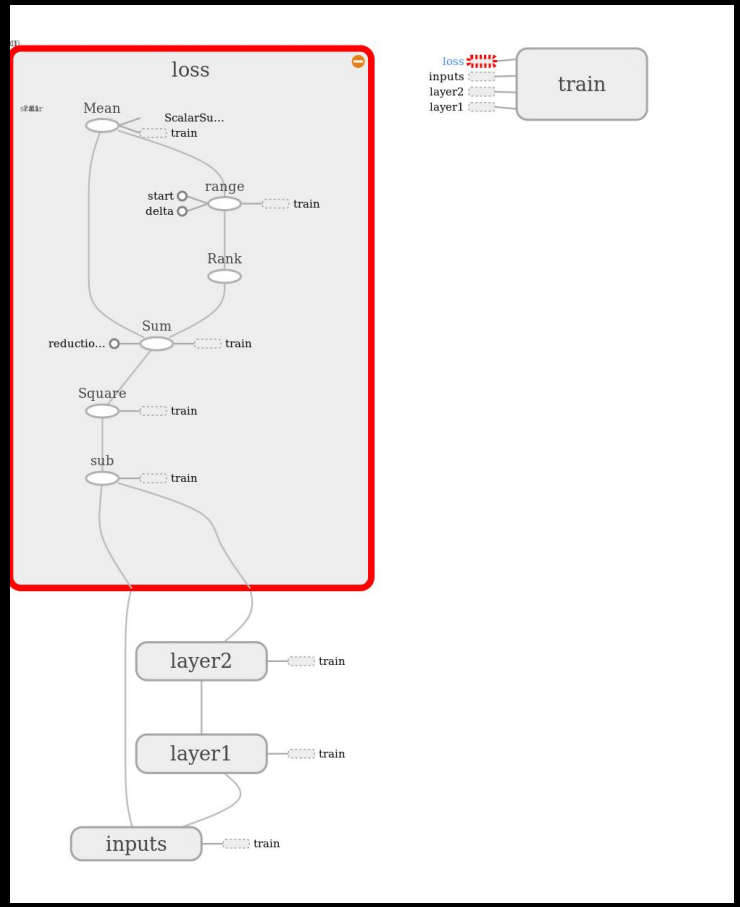




回归-code 5/5

第五步：输出训练结果





回归-code 5/5

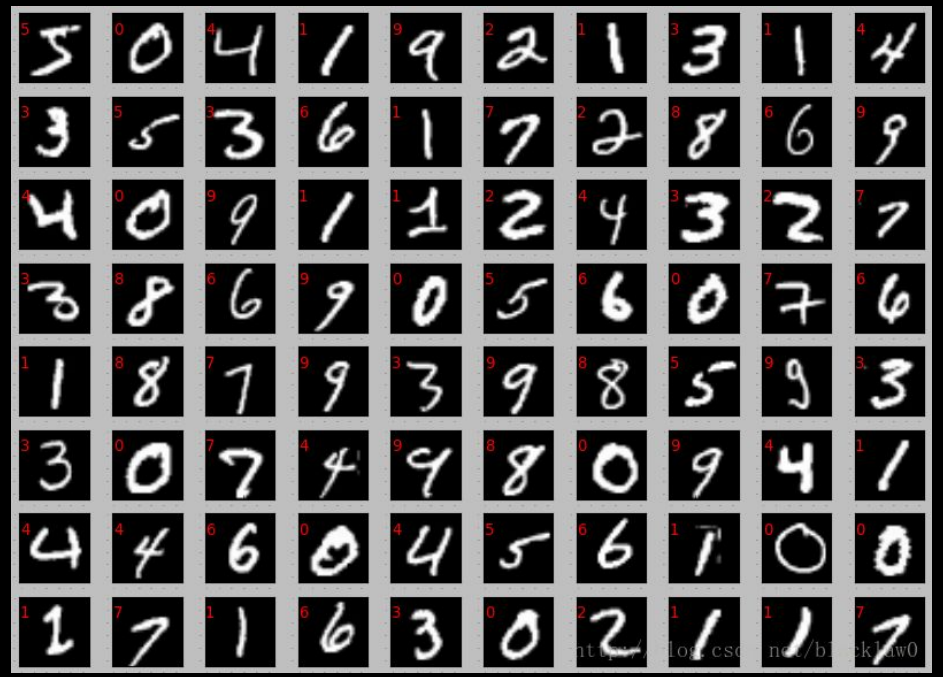
第五步：输出训练结果

```
Weights1=
 [[-0.11733268 -0.48034236 -0.03083559  0.87120265 -0.95855314 -0.31740719
  -0.46013582  0.90629649 -0.81020981 -0.98052007]]
biases1=
 [[ 0.07653242 -0.2624011   0.1820188  -0.26913378  0.19843166  0.3172363
  -0.12975664 -0.00327107 -0.01271619 -0.13476808]]
Weights2=
 [[ 0.21200591]
 [ 1.35447502]
 [-0.798397  ]
 [ 1.61474979]
 [-0.18205701]
 [-0.47955099]
 [ 0.56182081]
 [-0.29411045]
 [ 0.22289196]
 [ 0.7005806 ]]
biases2=
 [[-0.16945662]]
Model saved in file:  /tmp/jnb.ckpt
```

分类-code 1/5

第一步：生成或者导入 训练数据

```
mnist = input_data.read_data_sets('MNIST_data',
        one_hot=True)
```



分类-code 2/5

第二步：搭建神经网络

```
#添加输出层
prediction = add_layer(xs, 784, 10,
activation_function=tf.nn.softmax)
```

分类-code 3/5

第三步 : loss函数 优化算法

```
loss = tf.reduce_mean(-tf.reduce_sum(ys * tf.log(prediction),
                    reduction_indices=[1]))


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

分类-code 4/5

第四步：开始训练

```
for i in range(1000):
    sess.run(train_step, feed_dict={xs: batch_xs, ys: batch_ys})
```

分类-code 5/5

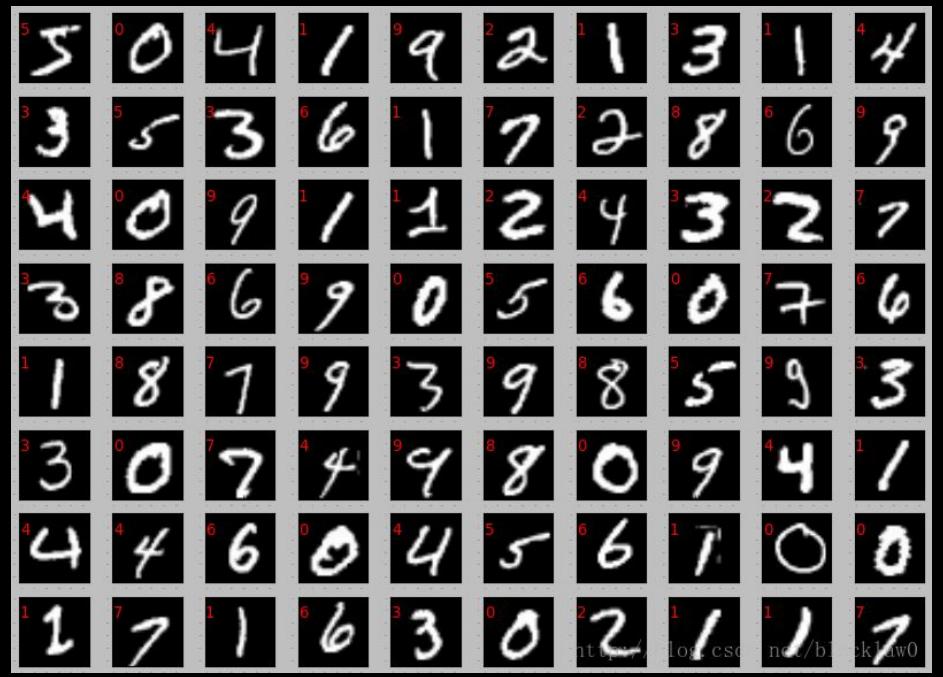
第五步：输出训练结果

```
>>>
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
0.0691
0.6277
0.7385
0.7827
0.8003
0.818
0.8291
0.8319
0.8378
0.8505
0.8547
0.8582
0.8603
0.867
0.8648
0.8702
0.8679
0.8742
0.8735
0.8746
```

分类2-code 1/5

第一步：生成或者导入 训练数据

```
mnist = input_data.read_data_sets('MNIST_data',
        one_hot=True)
```



分类2-code 2/5

第二步：搭建神经网络

#添加第一卷积池化层 Add_h_conv1_pool1

28*28*1→28*28*32→14*14*32

#添加第二卷积池化层 Add_h_conv1_pool2

14*14*32→14*14*64→7*7*64

#添加2层普通神经网络

```
mid = add_layer(xs, 7*7*64, 1024)
prediction = add_layer(mid, 1024, 10)
```

分类2-code 3/5

第三步 : loss函数 优化算法

```
loss= tf.reduce_mean(-tf.reduce_sum(ys * tf.log(prediction),
                    reduction_indices=[1]))

train_step = tf.train.AdamOptimizer(1e-4).minimize(loss)
```

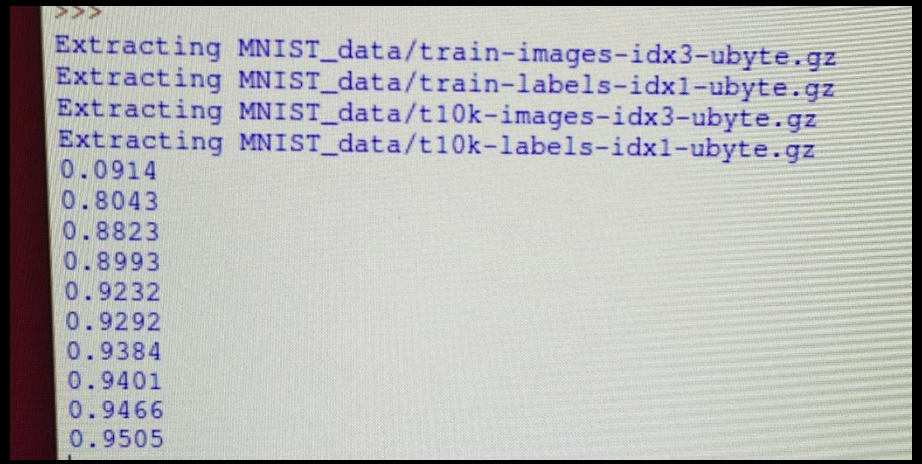
分类2-code 4/5

第四步：开始训练

```
for i in range(500):
    sess.run(train_step, feed_dict={xs: batch_xs, ys: batch_ys, keep_prob: 0.5})
```

分类2-code 5/5

第五步：输出训练结果



Any Questions?