Tensorflow 介绍

Outline

1)建立神经网络

-About Tensorflow

-Session

-Variables & Scope

-Placeholder

-Define add_layer

Outline

2)可视化

-结果可视化(matplotlib)

-Tensorboard

3)附录

-Activation functions & Optimizers

-参考

PART 1:建立神经网络

Q: What is Tensorflow?

A: A python-based toolkit for neural networks developed by Google.

Q: Advantages?

A: Open source, visualizable, and many metaframeworks for use!

- 从职位需求、谷歌搜索热度、Medium 文章数、arXiv 论文数和 GitHub 活跃度等方面审视不同的框架

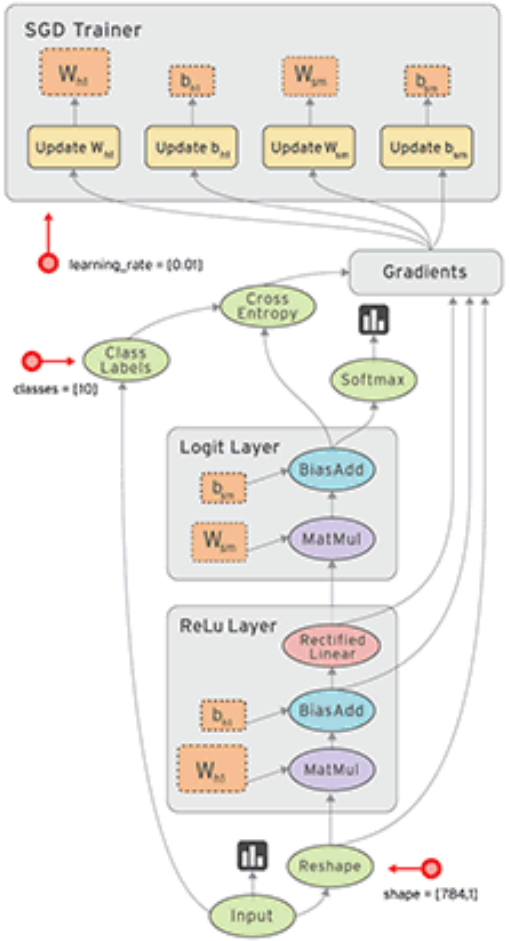
- https://mp.weixin.qq.com/s/9YAWr-ffZKR9KM8eIhbORw2018年最热门的深度学习框架？这份科学的排行榜可以告诉你

	Online Job Listings					KDnuggets	Google	Medium	Amazon	ArXiv	GitHub Activity			
Framework	Indeed	Monster	Simply Hired	LinkedIn	Angel List	Usage Survey	Search Volume	Articles	Books	Articles	Stars	Watchers	Forks	Contributors
TensorFlow	2,079	1,253	1,582	2,610	552	29.90%	78	45,300	202	201	109,576	8,334	67,551	1,642
Keras	684	364	449	695	177	22.20%	38	48,200	79	36	33,558	1,847	12,658	719
PyTorch	486	309	428	665	120	6.40%	30	6,410	18	42	18,716	952	4,474	760
Theano	356	316	279	508	95	4.90%	1	1,670	17	36	8,477	585	2,447	328
MXNET	266	154	200	298	29	1.50%	3	2,010	32	26	15,200	1,170	5,498	587
CNTK	126	96	97	160	12	3.00%	1	675	1	13	15,106	1,368	4,029	189
FastAI	0	0	0	0	0	0.00%	2	4,480	0	1	7,268	432	2,647	195

数据流图

- Tensorflow首先要定义神经网络结构，然后再把数据放入结构中去运算和training

- 因为TensorFlow是采用数据流图（data flow graphs）来计算, 所以首先我们得创建一个数据流图, 然后再将我们的数据（数据以张量(tensor)的形式存在）放在数据流图中计算.



Tensor张量意义

- 张量（Tensor):

张量有多种. 零阶张量为 纯量或标量 (scalar) 也就是一个数值. 比如 [1]

一阶张量为 向量 (vector), 比如 一维的 [1, 2, 3]

二阶张量为 矩阵 (matrix), 比如 二维的 [[1, 2, 3],[4, 5, 6],[7, 8, 9]]

以此类推, 还有 三阶 三维的 …

Operations

Category	Examples
Element-wise mathematical operations	Add, Sub, Mul, Div, Exp, Log, Greater, Less, Equal, ...
Array operations	Concat, Slice, Split, Constant, Rank, Shape, Shuffle, ...
Matrix operations	MatMul, MatrixInverse, MatrixDeterminant, ...
Stateful operations	Variable, Assign, AssignAdd, ...
Neural network building blocks	SoftMax, Sigmoid, ReLU, Convolution2D, MaxPool, ...
Checkpointing operations	Save, Restore
Queue and synchronization operations	Enqueue, Dequeue, MutexAcquire, MutexRelease, ...
Control flow operations	Merge, Switch, Enter, Leave, NextIteration

获取互斥锁

```
# Pretty standard, quite similar to numpy.
# See TensorFlow documentation
a=tf.constant[3,6]
b=tf.consrant[2,2]
tf.add(a,b) # >> [5 8]
tf.add_n([a, b, b]) # >> [7 10]. Equivalent to a + b + b
tf.mul(a, b) # >> [6 12] because mul is element wise
tf.matmul(a, b) # >> ValueError
tf.matmul(tf.reshape(a, [1, 2]), tf.reshape(b, [2, 1])) # >> [[18]]
tf.div(a, b) # >> [1 3]
tf.mod(a, b) # >> [1 0]
```

将矩阵a乘以矩阵b，生成a * b。

步骤：

- 1.导入模块(modules)
- 2.定义算法公式，也就是神经网络forward时的计算
- 3.定义loss，选定优化器，并指定优化器优化loss
- 4.迭代地对数据进行训练
- 5.在测试集或验证集上对准确率进行评测

会话(Session)

- Session 是 Tensorflow 为了控制,和输出文件的执行的语句.运行 session.run() 可以获得你要得知的运算结果, 或者是你所要运算的部分.
- 有两种方法来创建会话(Session)
- 例子：

```
import tensorflow as tf

# create two matrixes

matrix1 = tf.constant([[3,3]])
matrix2 = tf.constant([[2],
                       [2]])
product = tf.matmul(matrix1,matrix2)
```

会话(Session)

- 方法一：为Session创建一个变量

```
# method 1
sess = tf.Session()
result = sess.run(product)
print(result)
sess.close()
# [[12]]
```

Session()注意大写

- 方法二：with...as...

```
# method 2
with tf.Session() as sess:
    result2 = sess.run(product)
    print(result2)
# [[12]]
```

Variables & Scope

1)定义语法

State=tf.Variable(initializer,name)

2)初始化

init = tf.global_variables_initializer()

3)激活init

sess.run(init)

直接print(state)不起作用

一定要把 sess 的指针指向 state 再进行 print 才能得到想要的结果！

```python
import tensorflow as tf

state = tf.Variable(0, name='counter')

# 定义常量 one
one = tf.constant(1)

# 定义加法步骤 (注: 此步并没有直接计算)
new_value = tf.add(state, one)

# 将 State 更新成 new_value
update = tf.assign(state, new_value)
```

```python
# 如果定义 Variable, 就一定要 initialize
# init = tf.initialize_all_variables() # tf 马上就要废弃这种写法
init = tf.global_variables_initializer()  # 替换成这样就好

# 使用 Session
with tf.Session() as sess:
    sess.run(init)
    for _ in range(3):
        sess.run(update)
        print(sess.run(state))
```

Variables & Scope

举例：创建几个变量。

```
# create variable a with scalar value
a = tf.Variable(2, name="scalar")
# create variable b as a vector
b = tf.Variable([2, 3], name="vector")
# create variable c as a 2x2 matrix
c = tf.Variable([[0, 1], [2, 3]], name="matrix")
# create variable W as 784 x 10 tensor, filled with zeros
W = tf.Variable(tf.zeros([784,10]))
```

Variables & Scope

- 生成tensor的一些方法。

1)生成常量(constant)

tf.zeros(shape,dtype,name)

```
#生成全0数组
#tf.zeros([2, 3], int32)     # ==> [[0, 0, 0], [0, 0, 0]]
```

tf.zeros_like(tensor, dtype, name)

```
#生成一个与给定tensor类型、形状一致的常量，其所有元素为0
#a = tf.constant([[1, 2, 3], [4, 5, 6]] )
#tf.zeros_like(a)          # ==> [[0, 0, 0], [0, 0, 0]]
```

Variables & Scope

tf.ones(shape, dtype, name)
tf.ones_like(tensor, dtype, name)

tf.constant(value, dtype, shape, name)
#生成一个给定值的常量
#a = tf.constant([[1, 2, 3], [4, 5, 6]], int32) ==> [[1, 2, 3], [4, 5, 6]]
tf.fill(dims, value, name)
#生成一个全部为给定数字的数组
#a = tf.fill([2, 3], 9) # ==> [[9, 9, 9], [9, 9, 9]]

Variables & Scope

2)生成序列(sequence)

tf.range(start, limit, delta=1, name='range')

tf.linspace(start, stop, num, name=None)

Variables & Scope

3)生成随机数(random number)

函数名	随机数分布	主要参数
tf.random_normal	正态分布	平均值、标准差、取值类型
tf.truncated_normal	正态分布，如果随机数偏离均值超过2个标准差，就重新随机	平均值、标准差、取值类型
tf.random_uniform	平均分布 均匀分布	最小值、最大值、取值类型
tf.random_gamma	gamma分布	形状参数alpha、尺度参数beta、取值类型

Variables & Scope

3)生成随机数(random number)

tf.random_normal(shape, mean=0.0, stddev=1.0, dtype=tf.float32, seed=None, name=None)

tf.truncated_normal(shape, mean=0.0, stddev=1.0, dtype=tf.float32, seed=None, name=None)

tf.random_uniform(shape, minval=0.0, maxval=1.0, dtype=tf.float32, seed=None, name=None)

tf.random_gamma(shape, alpha, beta=None, dtype=tf.float32, seed=None, name=None)

scope命名方法

- scope 能让你命名变量的时候轻松很多. 同时也会在 reusing variable 代码中常常见到.

- 两种定义scope方式：

(1)tf.name_scope()

(2)tf.variable_scope()

例子：scope.py

Placeholder

- placeholder 是 Tensorflow 的占位符，暂时储存变量。Tensorflow如果想要从外部传入data，那就需要用到tf.placeholder，然后以这种形式传输数据sess.run(***,feed_dict={input:**})

1.定义placeholder的type，一般为float32形式。

```
import tensorflow as tf

#在 Tensorflow 中需要定义 placeholder 的 type，一般为 float32 形式
input1 = tf.placeholder(tf.float32)
input2 = tf.placeholder(tf.float32)

# mul = multiply 是将input1和input2 做乘法运算，并输出为 output
ouput = tf.multiply(input1, input2)
```

Placeholder

- 2.使用sess.run()进行传值，需要传入的值放在feed_dict={}。Placeholder和feed_dict={}是绑定在一起出现的。

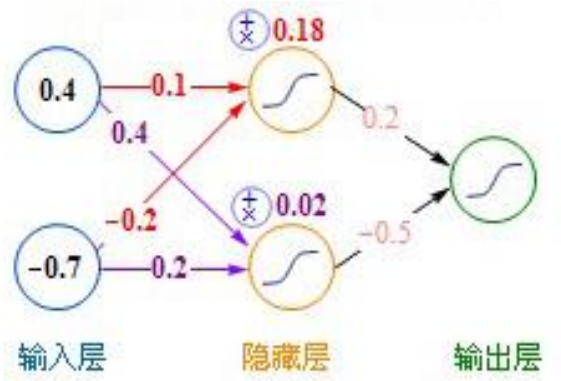
```
with tf.Session() as sess:
    print(sess.run(ouput, feed_dict={input1: [7.], input2: [2.]}))
# [ 14.]
```

TensorFlow Data Types

Data type	Python type	Description
DT_FLOAT	tf.float32	32 bits floating point.
DT_DOUBLE	tf.float64	64 bits floating point.
DT_INT8	tf.int8	8 bits signed integer.
DT_INT16	tf.int16	16 bits signed integer.
DT_INT32	tf.int32	32 bits signed integer.
DT_INT64	tf.int64	64 bits signed integer.
DT_UINT8	tf.uint8	8 bits unsigned integer.
DT_UINT16	tf.uint16	16 bits unsigned integer.
DT_STRING	tf.string	Variable length byte arrays. Each element of a Tensor is a byte array.
DT_BOOL	tf.bool	Boolean.
DT_COMPLEX64	tf.complex64	Complex number made of two 32 bits floating points: real and imaginary parts.
DT_COMPLEX128	tf.complex128	Complex number made of two 64 bits floating points: real and imaginary parts.
DT_QINT8	tf.qint8	8 bits signed integer used in quantized Ops.
DT_QINT32	tf.qint32	32 bits signed integer used in quantized Ops.
DT_QUINT8	tf.quint8	8 bits unsigned integer used in quantized Ops.

Define add_layer

- 在 Tensorflow 里定义一个添加层的函数可以很容易的添加神经层,为之后的添加省下不少时间。
- 神经层里常见的参数通常有weights、biases和激励函数。



$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = AF\begin{pmatrix} W_{1,1}x_1 + W_{1,2}x_2 + b_1 \\ W_{2,1}x_1 + W_{2,2}x_2 + b_2 \end{pmatrix}$$

- 定义添加神经层的函数def add_layer(),它有四个参数：输入值、输入的大小、输出的大小和激励函数，设定默认的激励函数是None。

Define add_layer

```
def add_layer(inputs, in_size, out_size, activation_function=None):
    Weights = tf.Variable(tf.random_normal([in_size, out_size]))
    biases = tf.Variable(tf.zeros([1, out_size]) + 0.1)
    Wx_plus_b = tf.matmul(inputs, Weights) + biases
    if activation_function is None:
        outputs = Wx_plus_b
    else:
        outputs = activation_function(Wx_plus_b)
return outputs
```

生成初始参数时，随机变量会比全部为0好很多。

在机器学习中，bias的推荐值不为0，所以在0向量的基础上又加了0.1

Part2：可视化

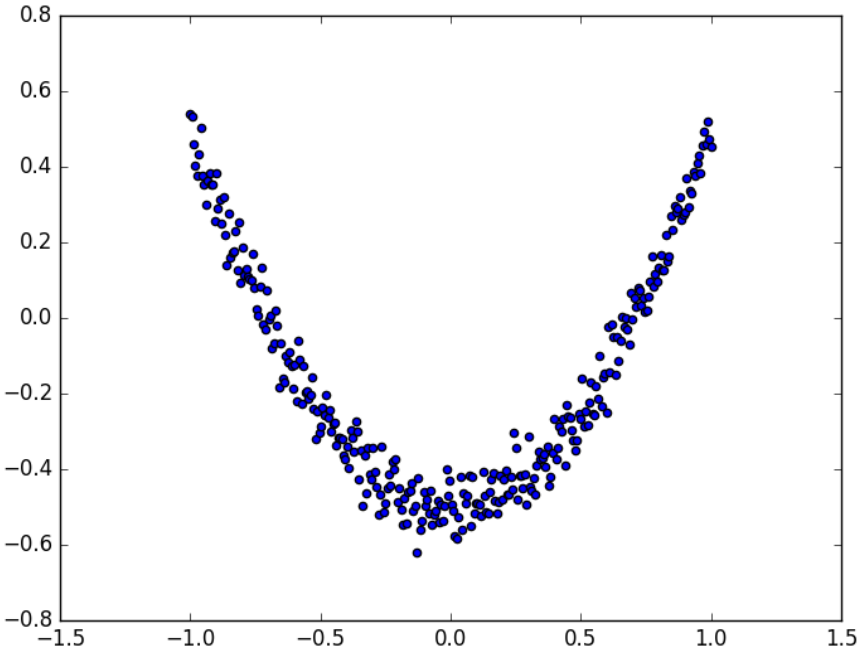
- Matplotlib模块

1)导入模块

```
import matplotlib.pyplot as plt
```

2) plt.figure()→add_subplot()→plt.show()

```
# plot the real data
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.scatter(x_data, y_data)
```



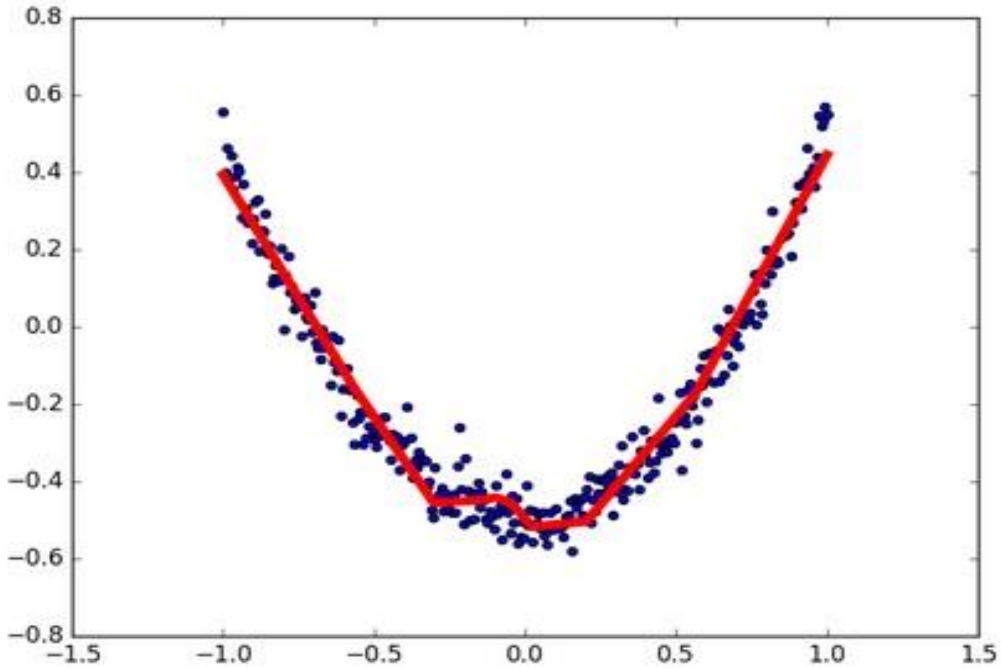
Part2：可视化

- Matplotlib模块

3)连续显示

plt.ion()

Ex:



- TensorBoard

1)TensorBoard是TensorFlow的一组Web应用，用来监控TensorFlow运行进程，或可视化computation。

目前支持5种可视化：

标量(scalar)、图片(images)、音频(audio)、直方图(histograms)和计算图(Computation Graph)

- Tensorboard

2)通过tf.summary.FileWriter()方式将绘出的图保存在一个目录里。

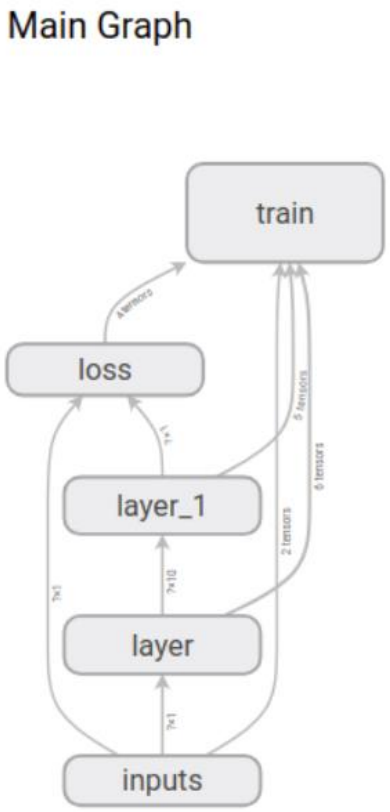
```
sess = tf.Session() # get session
# tf.train.SummaryWriter soon be deprecated, use following
writer = tf.summary.FileWriter("logs/", sess.graph)
```

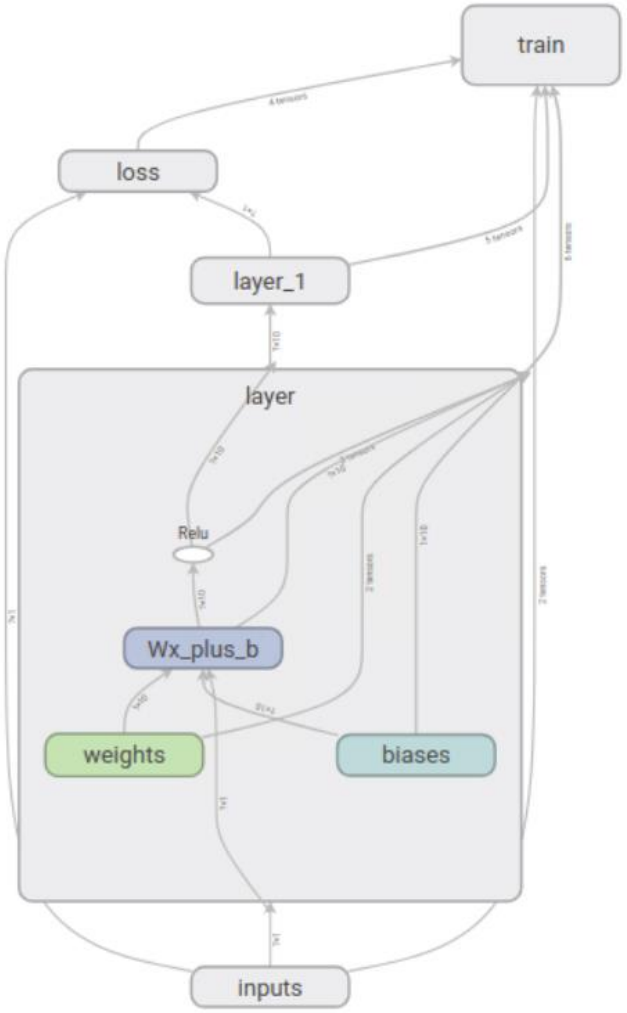
在终端输入：

```
tensorboard --logdir logs
```

在Google Chrome： http://0.0.0.0:6006 or http://localhost:6006

3)例子：tensorboard.py

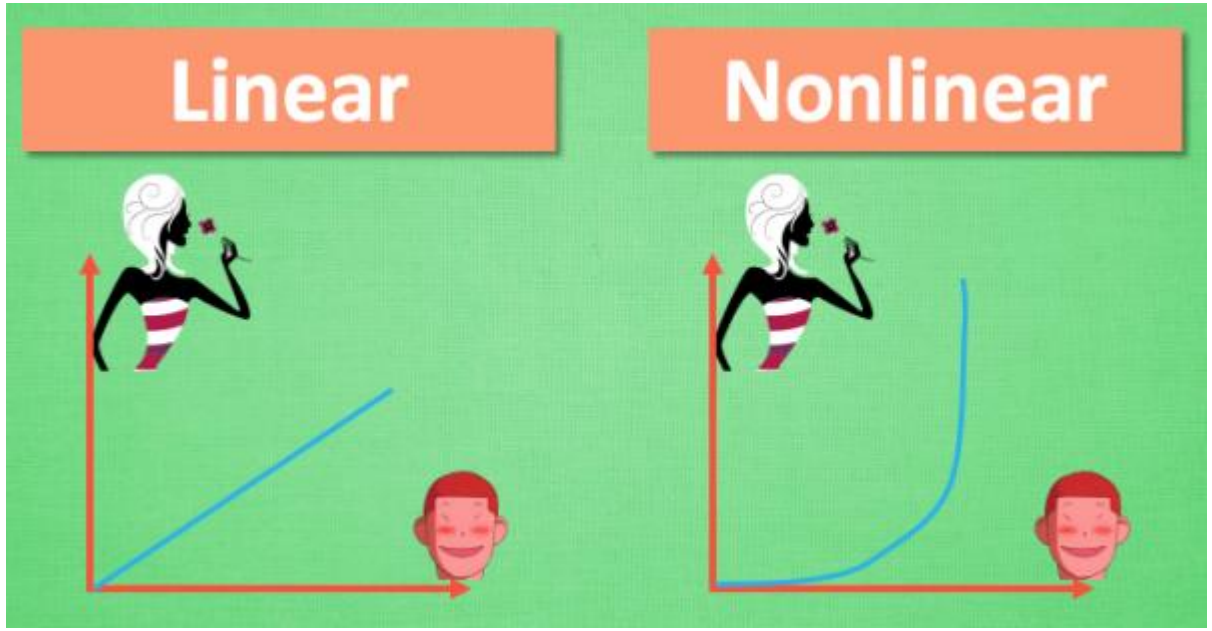


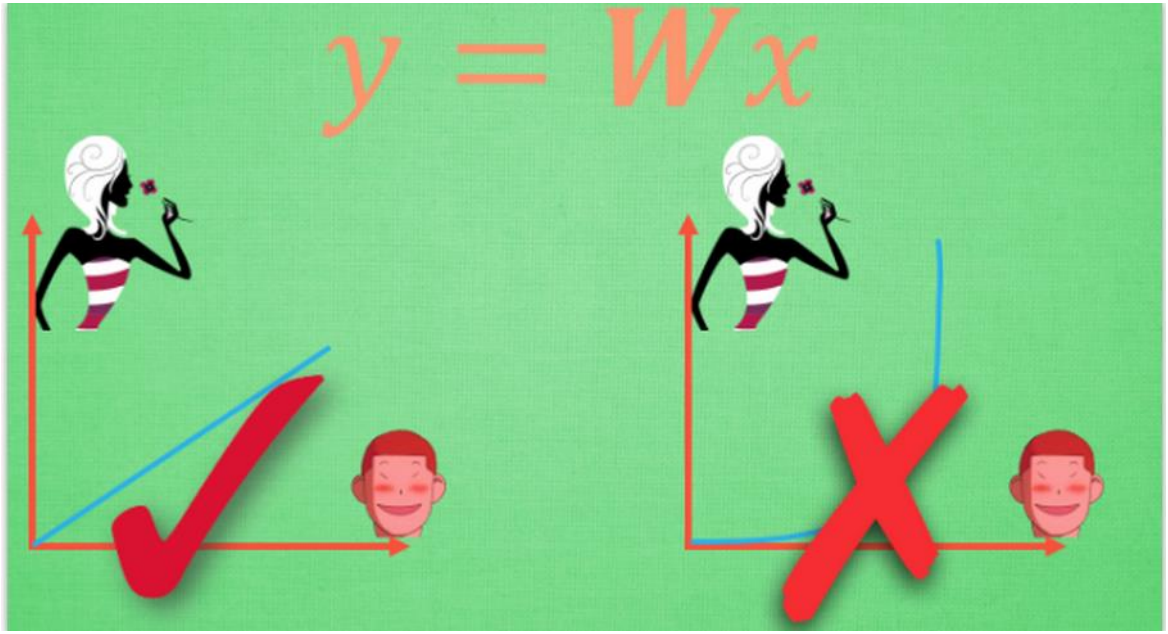


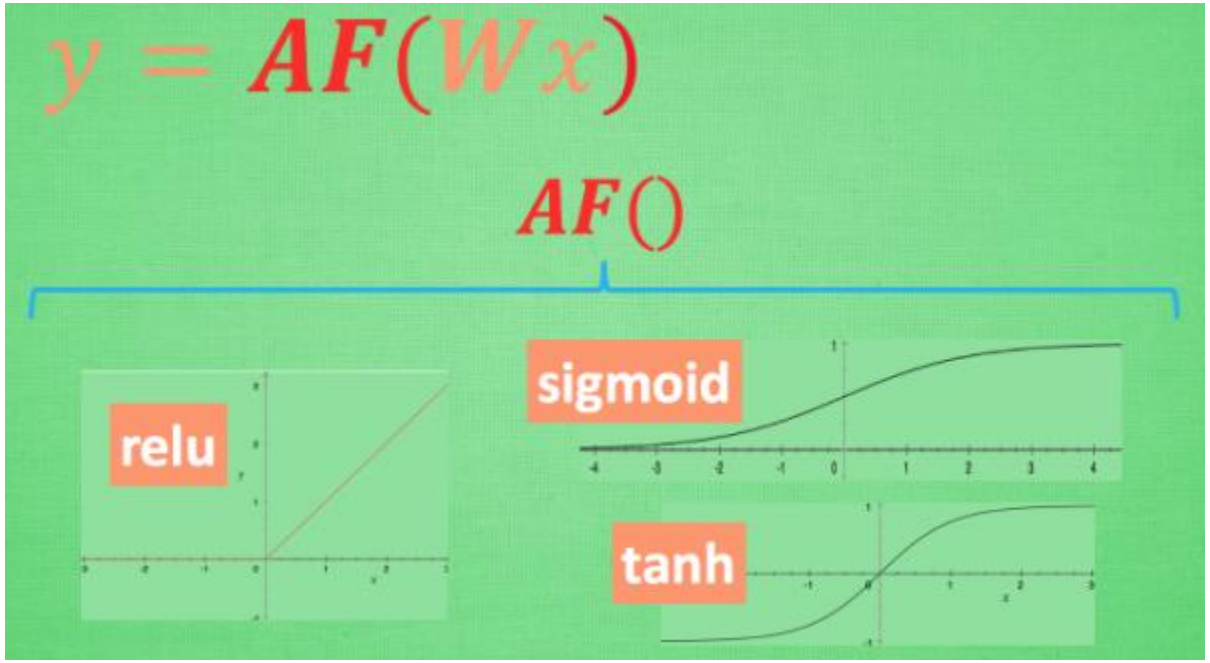
Part3：附录

- 激活函数(Activation functions)

  为什么使用激励函数？激励函数也就是为了解决我们日常生活中不能用线性方程所概括的问题。







要确保的是这些激励函数必须是可以微分的, 因为在 backpropagation 误差反向传递的时候, 只有这些可微分的激励函数才能把误差传递回去。

如何选择:

1)当你的神经网络层只有两三层，不是很多的时候，对于隐藏层，使用任意的激励函数,，不会有特别大的影响。

2)当你使用特别多层的神经网络，不能随意使用激励函数，因为这会涉及到梯度爆炸, 梯度消失的问题。

Sigmoid函数在反向传播中梯度值会逐渐减小，经过多层传递后呈指数级急剧减小，ReLU函数比较完美地解决了梯度弥散问题。

ReLU对比Sigmoid的主要变化有如下3点。

(1)单侧抑制。

(2)相对宽阔的兴奋边界。

(3)稀疏激活性。

MLP和CNN大部分情况下使用ReLU及其变种(EIU,PReLU,RReLU)。

RNN内部主要还是使用Sigmoid、Tanh、Hard Sigmoid。

神经网络的输出层一般还是使用Sigmoid函数，因为它最接近概率输出分布。

- 优化器(Optimizers)

Optimizers

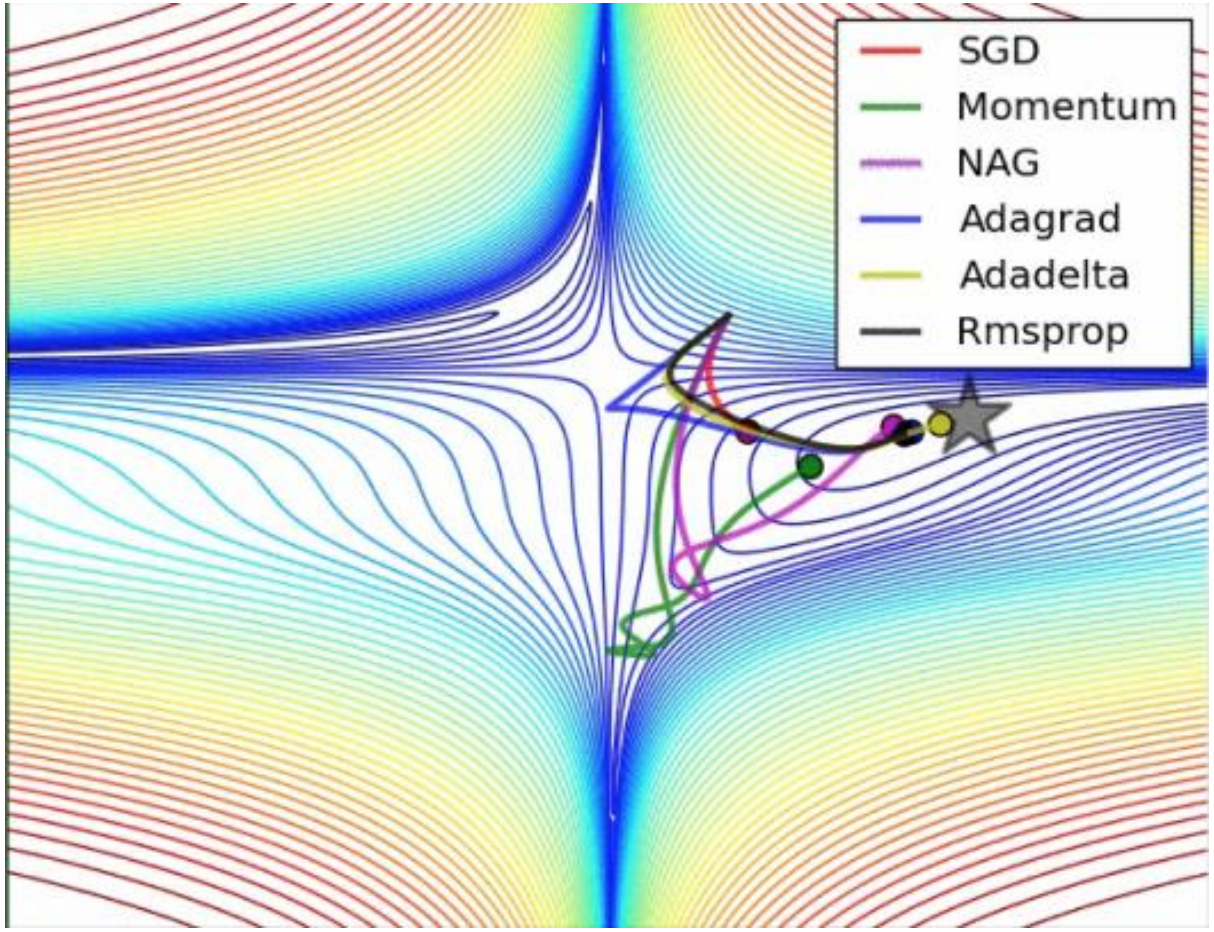
The Optimizer base class provides methods to compute gradients for a loss and apply gradients to variables. A collection of subclasses implement classic optimization algorithms such as GradientDescent and Adagrad.

You never instantiate the Optimizer class itself, but instead instantiate one of the subclasses.

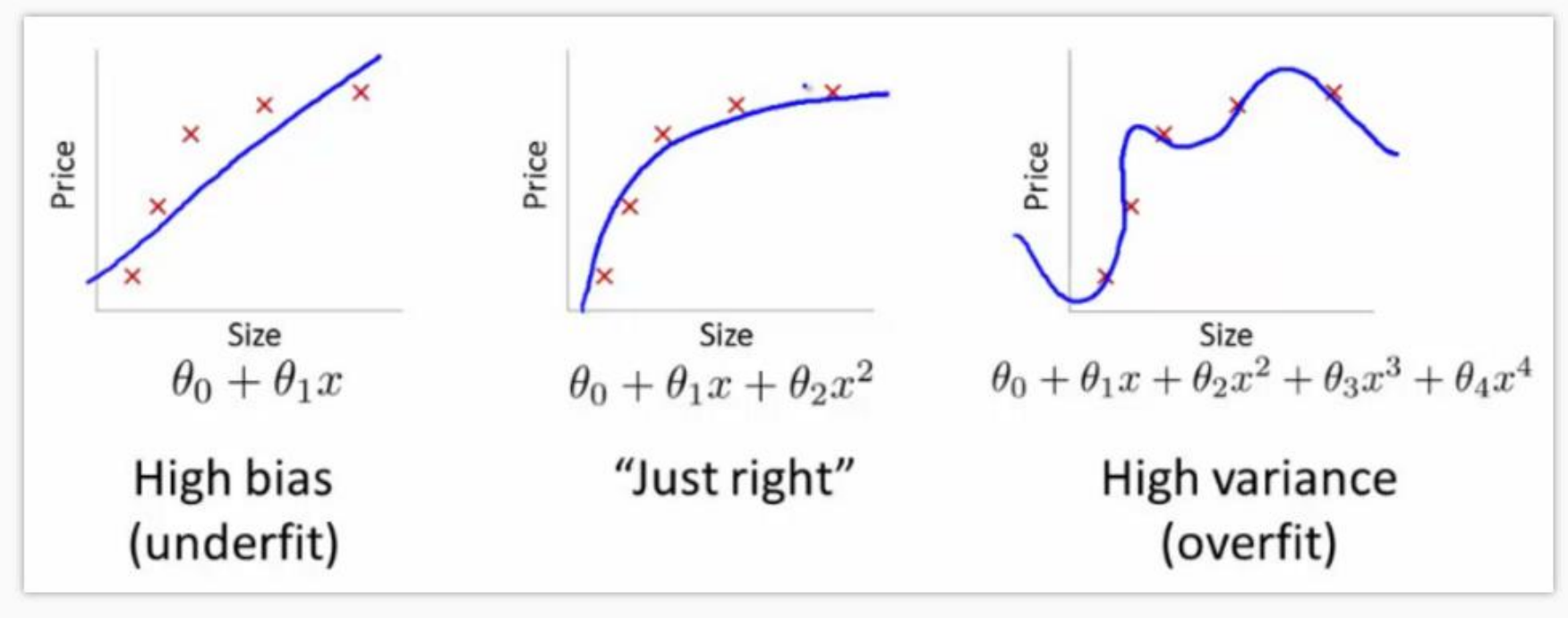
- `tf.train.Optimizer`
- `tf.train.GradientDescentOptimizer`
- `tf.train.AdadeltaOptimizer`
- `tf.train.AdagradOptimizer`
- `tf.train.AdagradDAOptimizer`
- `tf.train.MomentumOptimizer`
- `tf.train.AdamOptimizer`
- `tf.train.FtrlOptimizer`
- `tf.train.ProximalGradientDescentOptimizer`
- `tf.train.ProximalAdagradOptimizer`
- `tf.train.RMSPropOptimizer`

See `tf.contrib.opt` for more optimizers.

- 优化器(Optimizers)



- 过拟合：预测准确率在训练集上提升，但是在测试集上反而下降了



- 使用dropout方法来解决过拟合问题。
  - 在训练时，将神经网络某一层的输出节点数据随机丢弃一部分。这种做法实质上等于创造出了很多新的随机样本，通过增大样本量、减少特征数量来防止过拟合。

- 建立dropout层

```
keep_prob = tf.placeholder(tf.float32)
...
...
Wx_plus_b = tf.nn.dropout(Wx_plus_b, keep_prob)
```

```
sess.run(train_step, feed_dict={xs: X_train, ys: y_train, keep_prob: 0.5})
#sess.run(train_step, feed_dict={xs: X_train, ys: y_train, keep_prob: 1})
```

例子1：建立神经网络(回归)

- 第一步:导入模块

```
import tensorflow as tf
import numpy as np
```

- 第二步：构造添加一个神经层的函数

```
def add_layer(inputs, in_size, out_size, activation_function=None):
    Weights = tf.Variable(tf.random_normal([in_size, out_size]))
    biases = tf.Variable(tf.zeros([1, out_size]) + 0.1)
    Wx_plus_b = tf.matmul(inputs, Weights) + biases
    if activation_function is None:
        outputs = Wx_plus_b
    else:
        outputs = activation_function(Wx_plus_b)
    return outputs
```

第三步：导入数据

```
x_data = np.linspace(-1,1,300, dtype=np.float32)[:, np.newaxis]
noise = np.random.normal(0, 0.05, x_data.shape).astype(np.float32)
y_data = np.square(x_data) - 0.5 + noise
```

利用占位符定义我们所需的神经网络的输入。 tf.placeholder()就是代表占位符。

```
xs = tf.placeholder(tf.float32, [None, 1])
ys = tf.placeholder(tf.float32, [None, 1])
```

- 第四步：搭建网络

定义隐藏层：

```
l1 = add_layer(xs, 1, 10, activation_function=tf.nn.relu)
```

定义输出层：

```
prediction = add_layer(l1, 10, 1, activation_function=None)
```

计算预测值和真实值误差：

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

- 第四步：搭建网络

让机器学习提高它的准确率：

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

通常学习率小于1，这里取0.1，代表以0.1的效率来最小化误差loss。

- 第五步：对变量进行初始化

```
init = tf.global_variables_initializer()
```

定义会话(Session)，并用Session来执行init初始化步骤。

```
sess = tf.Session()
sess.run(init)
```

- 第六步：训练

机器学习的内容是train_step，用Session来run每一次训练的数据，逐步提升神经网络的预测准确性。

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

每50步输出一下机器学习的误差：

```
if i % 50 == 0:
    # to see the step improvement
    print(sess.run(loss, feed_dict={xs: x_data, ys: y_data}))
```

例子2：手写体识别MNIST

- MNIST数据集
  - MNIST 数据集来自美国国家标准与技术研究所, **National Institute of Standards and Technology (NIST)**. 训练集 (training set) 由来自 250 个不同人手写的数字构成, 其中 50% 是高中学生, 50% 来自人口普查局 (the Census Bureau) 的工作人员. 测试集(test set) 也是同样比例的手写数字数据.

  - MNIST 数据集可在 http://yann.lecun.com/exdb/mnist/ 获取, 它包含了四个部分:
    - Training set images: train-images-idx3-ubyte.gz (9.9 MB, 解压后 47 MB, 包含 60,000 个样本)
    - Training set labels: train-labels-idx1-ubyte.gz (29 KB, 解压后 60 KB, 包含 60,000 个标签)
    - Test set images: t10k-images-idx3-ubyte.gz (1.6 MB, 解压后 7.8 MB, 包含 10,000 个样本)
    - Test set labels: t10k-labels-idx1-ubyte.gz (5KB, 解压后 10 KB, 包含 10,000 个标签)

TRAINING SET LABEL FILE (train-labels-idx1-ubyte):

```
[offset] [type]          [value]          [description]
0000     32 bit integer  0x00000801(2049) magic number (MSB first)
0004     32 bit integer  60000            number of items
0008     unsigned byte   ??               label
0009     unsigned byte   ??               label
........
xxxx     unsigned byte   ??               label

The labels values are 0 to 9.
```

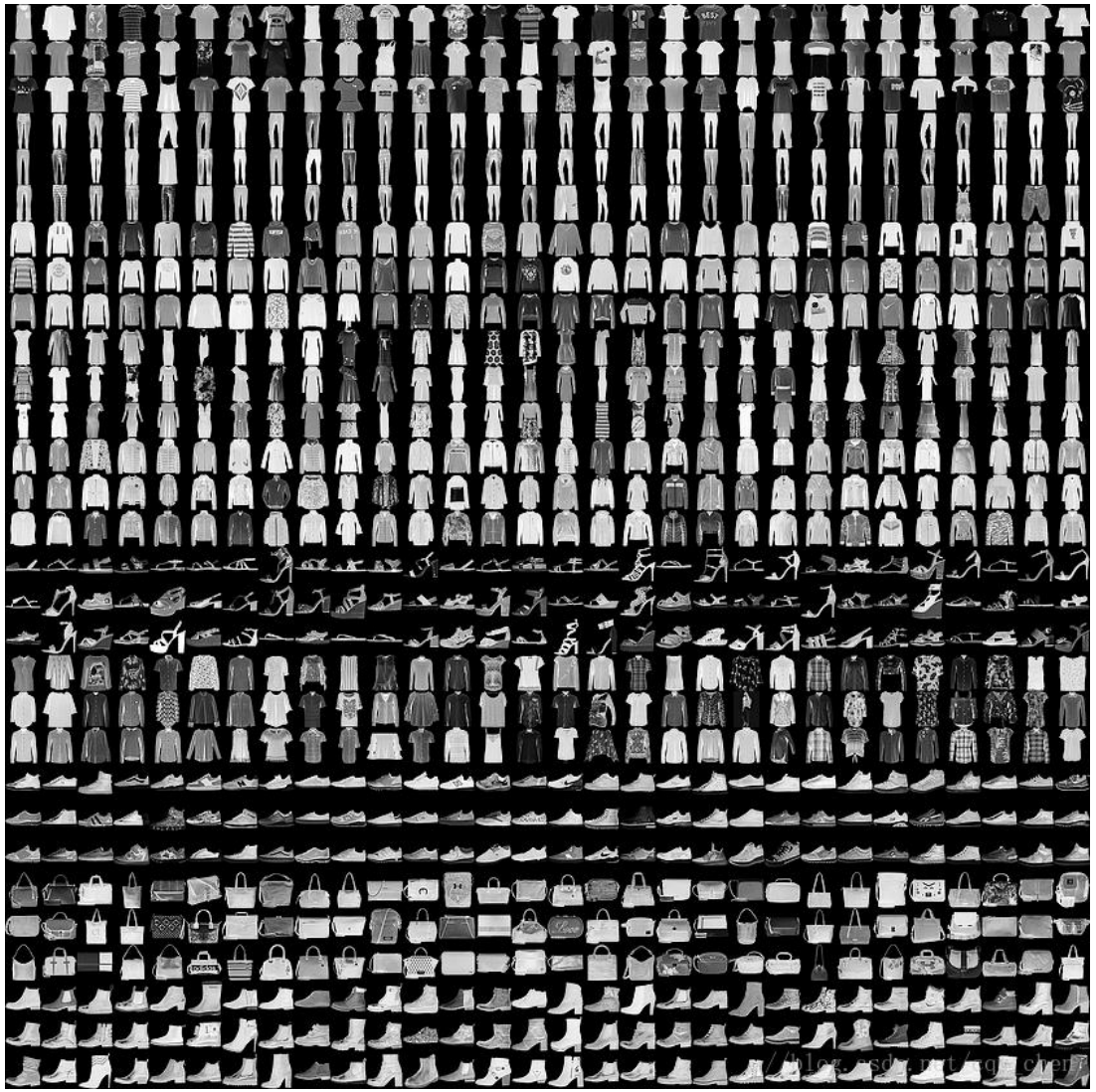
TRAINING SET IMAGE FILE (train-images-idx3-ubyte):

```
[offset] [type]          [value]          [description]
0000     32 bit integer  0x00000803(2051) magic number
0004     32 bit integer  60000            number of images
0008     32 bit integer  28               number of rows
0012     32 bit integer  28               number of columns
0016     unsigned byte   ??               pixel
0017     unsigned byte   ??               pixel
........
xxxx     unsigned byte   ??               pixel
```

Pixels are organized row-wise. Pixel values are 0 to 255. 0 means background (white), 255 means foreground (black).

MNIST Database

FashionMNIST Database



例子2：手写体识别MNIST

目标：

X: image of a handwri1en digit

Y: the digit value

Recognize the digit in the image

第一步：导入模块

```
from tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
```

第二步：导入数据

```
from tensorflow.examples.tutorials.mnist import input_data
MNIST = input_data.read_data_sets(  "/mnist", one_hot=True)
x = tf.placeholder(  tf.float32,[None,784],name='image')
y_ = tf.placeholder(  tf.float32,[None,10],name='label')
```

第三步：搭建网络

(1)创建Weight和bias

tf.Variable(initial_value=None, trainable=True, collections=None, name=None, dtype=None, …)

###Code

```
W=tf.Variable()
b=tf.Variable()
```

第三步：搭建网络

(2)建立模型

处理多分类任务时，通常需要使用Softmax Regression模型。

工作原理:将可以判定为某类的特征相加，然后将这些特征转化为判定是这一类的概率。

$$feature_i = \sum_j W_{i,j} x_j + b_i$$

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

第三步：搭建网络

(2)建立模型

y=softmax(Wx+b)

###Code

y=tf.nn.sotfmax()

Category	Examples
Element-wise mathematical operations	Add, Sub, Mul, Div, Exp, Log, Greater, Less, Equal, ...
Array operations	Concat, Slice, Split, Constant, Rank, Shape, Shuffle, ...
Matrix operations	MatMul, MatrixInverse, MatrixDeterminant, ...
Stateful operations	Variable, Assign, AssignAdd, ...
Neural network building blocks	SoftMax, Sigmoid, ReLU, Convolution2D, MaxPool, ...
Checkpointing operations	Save, Restore
Queue and synchronization operations	Enqueue, Dequeue, MutexAcquire, MutexRelease, ...
Control flow operations	Merge, Switch, Enter, Leave, NextIteration

第三步：搭建网络

(3)定义loss function
对于多分类问题，通常使用交叉熵(cross-entropy)作为loss function

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

y'真实概率分布，y是预测概率分布

reduce_mean求均值，reduce_sum求和

###Code

cross_entropy=
tf.reduce_mean(-tf.reduce_sum(_____,reduction_indices=[1]))

第三步：搭建网络

(4)定义优化算法

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

第四步：训练

(1)全局参数初始化

(2)迭代训练

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

作业：

- 1.完成例子2：MNIST手写体识别分类（不含隐藏层）
- 2.加分题：
  - （1）增加隐藏层　　(隐藏层使用relu激活函数，tf.nn.relu())
  - （2）使用Dropout防止过拟合　　(tf.nn.dropout())
  - （3）使用自适应优化器Adagrad　　(tf.train.AdagradOptimizer)