数据准备

读取数据

```
%matplotlib notebook

import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.utils import shuffle

# 读取数据文件
df = pd.read_csv("data/boston.csv",

#显示数据摘要描述信息
print (df.describe())
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	\
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	
75%	3.677082	12.500000	18.100000	0.000000	0.624000	6.623500	
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	

	AGE	DIS	RAD	TAX	PTRATIO	LSTAT	\
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	
mean	68.574901	3.795043	9.549407	408.237154	18.455534	12.653063	
std	28.148861	2.105710	8.707259	168.537116	2.164946	7.141062	
min	2.900000	1.129600	1.000000	187.000000	12.600000	1.730000	
25%	45.025000	2.100175	4.000000	279.000000	17.400000	6.950000	
50%	77.500000	3.207450	5.000000	330.000000	19.050000	11.360000	
75%	94.075000	5.188425	24.000000	666.000000	20.200000	16.955000	
max	100.000000	12.126500	24.000000	711.000000	22.000000	37.970000	

	MEDV
count	506.000000
mean	22.532806
std	9.197104
min	5.000000
25%	17.025000
50%	21.200000
75%	25.000000
max	50.000000

通过pandas读取数据文件，列出统计概述

数据准备

载入本示例所需数据

```
# 获取df的值

df = df.values
```

```
# 把 df 转换为 np 的数组格式

df = np.array(df)
```

```
# x_data 为 前12列特征数据
x_data = df[:,:12]

# y_data 为最后1列标签数据
y_data = df[:,12]
```

```
print(df)
```

```
[[  6.32000000e-03   1.80000000e+01   2.31000000e+00 ...,   1.53000000e+01
    4.98000000e+00   2.40000000e+01]
 [  2.73100000e-02   0.00000000e+00   7.07000000e+00 ...,   1.78000000e+01
    9.14000000e+00   2.16000000e+01]
 [  2.72900000e-02   0.00000000e+00   7.07000000e+00 ...,   1.78000000e+01
    4.03000000e+00   3.47000000e+01]
...,
 [  6.07600000e-02   0.00000000e+00   1.19300000e+01 ...,   2.10000000e+01
    5.64000000e+00   2.39000000e+01]
 [  1.09590000e-01   0.00000000e+00   1.19300000e+01 ...,   2.10000000e+01
    6.48000000e+00   2.20000000e+01]
 [  4.74100000e-02   0.00000000e+00   1.19300000e+01 ...,   2.10000000e+01
    7.88000000e+00   1.19000000e+01]]
```

模型定义

定义训练数据占位符

定义**特征数据和标签数据**的占位符

```
x = tf.placeholder(tf.float32, [None, 12], name = "X") # 12个特征数据（12列）
y = tf.placeholder(tf.float32, [None, 1], name = "Y") # 1个标签数据（1列）
```

shape中 None 表示行的数量未知，在实际训练时决定一次代入多少行样本，从一个样本的随机SDG到批量SDG都可以

定义模型结构

定义模型函数

```
# 定义了一个命名空间

with tf.name_scope("Model"):

    # w 初始化值为shape=(12,1)的随机数
    w = tf.Variable(tf.random_normal([12,1], stddev=0.01), name="W")

    # b 初始化值为 1.0
    b = tf.Variable(1.0, name="b")

    # w和x是矩阵相乘，用matmul,不能用mutiply或者*
    def model(x, w, b):
        return tf.matmul(x, w) + b

    # 预测计算操作，前向计算节点
    pred= model(x, w, b)
```

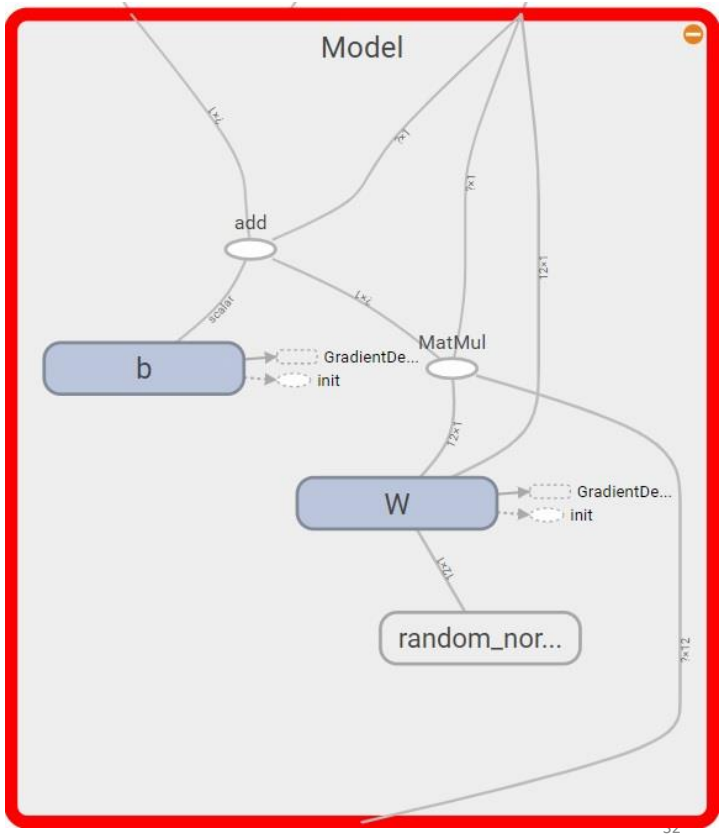
$$Y = x_1 \times w_1 + x_2 \times w_2 + \ldots + x_n \times w_n + b$$

$$[x_1 \quad x_2 \quad x_n] * \begin{bmatrix} w_1 \\ w_2 \\ w_n \end{bmatrix} + b = [x_1 * w_1 + x_2 * w_2 + x_n * w_n] + b$$

计算图和命名空间

命名空间 **name_scope**

Tensorflow计算图模型中常有数以千计节点，在可视化过程中很难一下子全部展示出来，因此可用name_scope为变量划分范围，在可视化中，这表示在计算图中的一个层级。



模型训练

模型训练

设置训练超参数

```
# 迭代轮次
train_epochs = 50

# 学习率
learning_rate = 0.01
```

模型训练

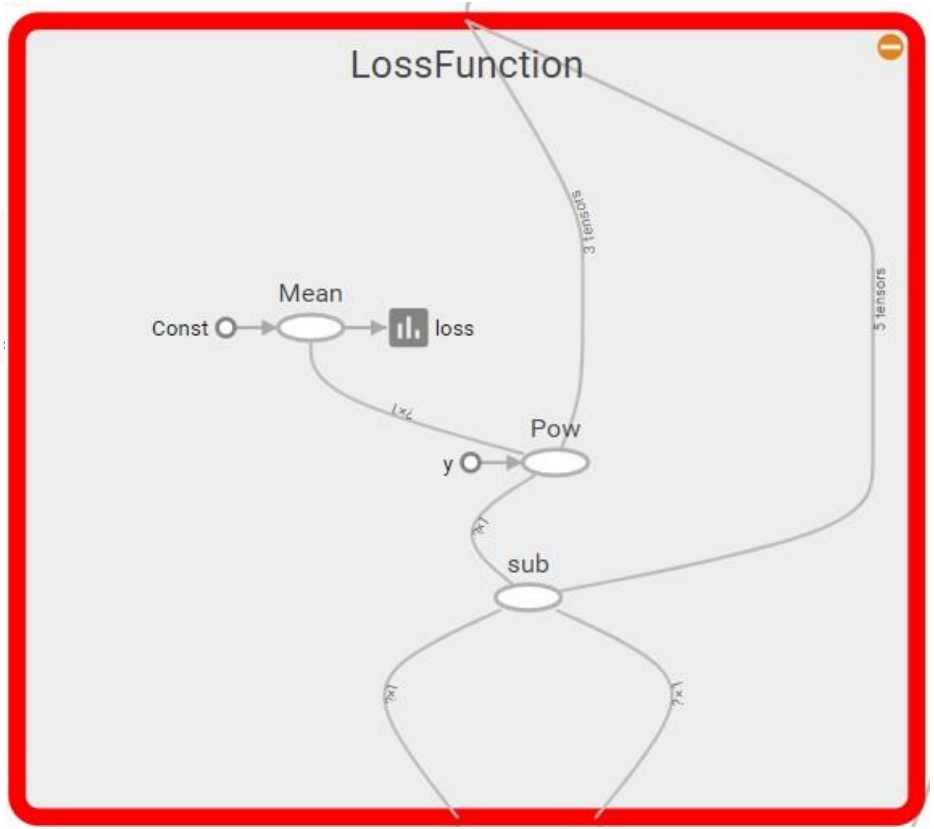
定义均方差损失函数

```
# 定义损失函数

with tf.name_scope("LossFunction"):
    loss_function = tf.reduce_mean(tf.pow(y-pred, 2)) #均方误差
```

模型训练

定义均方差损失函数

```
# 定义损失函数

with tf.name_scope("LossFunction"):
    loss_function = tf.reduce_mean(tf.pow(y-pred, 2))
```



模型训练

选择优化器

```
# 创建优化器

optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(loss_function)
```

常用优化器包括：

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

模型训练

声明会话

```
sess = tf.Session()

# 定义初始化变量的操作
init = tf.global_variables_initializer()
```

启动会话

```
sess.run(init)
```

模型训练

浙江大学城市学院 ZHEJIANG UNIVERSITY CITY COLLEGE

迭代训练

```
for epoch in range (train_epochs):
    loss_sum = 0.0
    for xs, ys in zip(x_data, y_data):

        xs = xs.reshape(1,12)
        ys = ys.reshape(1,1)

        _, loss = sess.run([optimizer,loss_function], feed_dict={x: xs, y: ys})

        loss_sum = loss_sum + loss

    # 打乱数据顺序
    x_data, y_data = shuffle(x_data, y_data)

    b0temp=b.eval(session=sess)
    w0temp=w.eval(session=sess)
    loss_average = loss_sum/len(y_data)

    print("epoch=", epoch+1,"loss=", loss_average,"b=", b0temp,"w=", w0temp )
```

Feed数据必须和Placeholder的shape一致

模型训练

思考：为什么要打乱数据顺序？

```
# 打乱数据顺序
x_data, y_data = shuffle(x_data, y_data)
```

模型训练

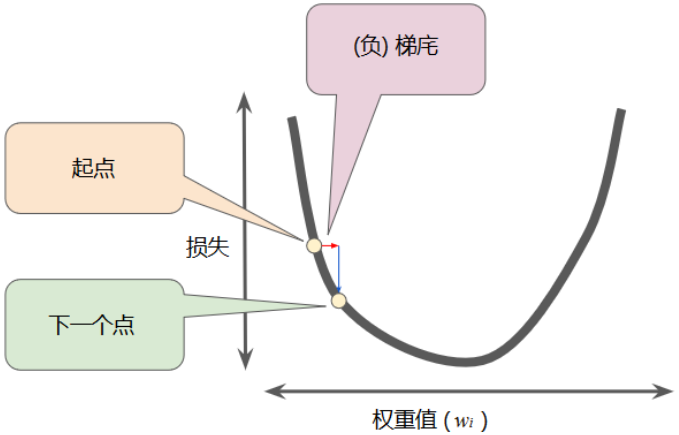
训练过程输出

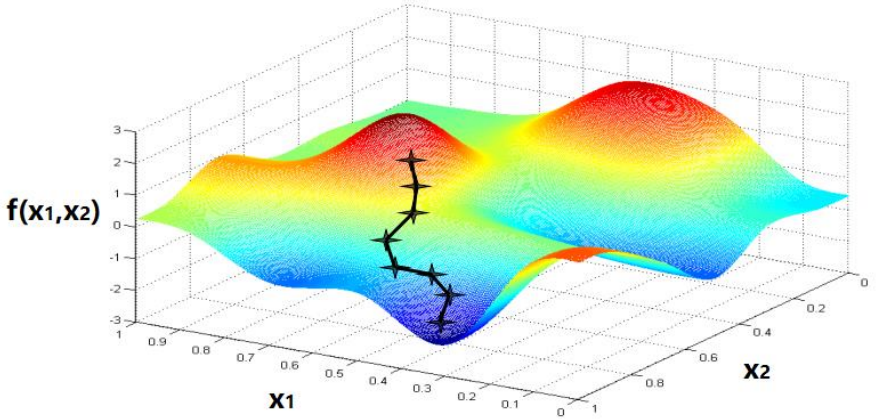
```
    print("epoch=", epoch+1,"loss=", loss_average,"b=", b0temp,"w=", w0temp )
```

```
epoch= 1 loss= nan b= nan w= [[ nan]
 [ nan]
 [ nan]
 [ nan]
 [ nan]
 [ nan]
 [ nan]
 [ nan]
 [ nan]
 [ nan]
 [ nan]
 [ nan]]
epoch= 2 loss= nan b= nan w= [[ nan]
 [ nan]
 [ nan]
 [ nan]
 [ nan]
 [ nan]
```

探究训练结果异常的原因：从梯度下降讲起

梯度下降





单变量（一元）

两个变量（二元）

梯度下降

要考虑不同特征值取值范围大小的影响

杭州笋干老鸭汤

原材料的配比：鸭、火腿、笋干、青菜、水、油、盐、酱油、葱、姜、蒜、胡椒粉.....

梯度下降

要考虑不同特征值取值范围大小的影响

	CRIM	ZN	INDUS	CHAS	NOX	RM	\
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	
75%	3.677082	12.500000	18.100000	0.000000	0.624000	6.623500	
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	

	AGE	DIS	RAD	TAX	PTRATIO	LSTAT	\
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	
mean	68.574901	3.795043	9.549407	408.237154	18.455534	12.653063	
std	28.148861	2.105710	8.707259	168.537116	2.164946	7.141062	
min	2.900000	1.129600	1.000000	187.000000	12.600000	1.730000	
25%	45.025000	2.100175	4.000000	279.000000	17.400000	6.950000	
50%	77.500000	3.207450	5.000000	330.000000	19.050000	11.360000	
75%	94.075000	5.188425	24.000000	666.000000	20.200000	16.955000	
max	100.000000	12.126500	24.000000	711.000000	22.000000	37.970000	

归一化

(特征值-特征值最小值)/ (特征值最大值 - 特征值最小值）

0-1