使用线性回归预测房价

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

In [2]:
```
x = np.loadtxt('boston/x.txt')
y = np.loadtxt('boston/y.txt')
```

In [3]:
```
x.shape
```

Out[3]: (506, 13)

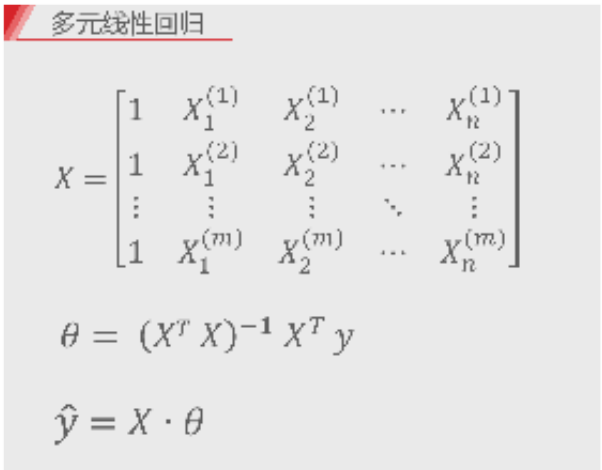
In [4]:
```
y.shape
```

Out[4]: (506,)

In [5]:
```
np.ones((5, 1))
```

Out[5]:
```
array([[1.],
       [1.],
       [1.],
       [1.],
       [1.]])
```

In [6]:
```
img = mpimg.imread('duoyuan.png')
img.shape
plt.imshow(img)
plt.axis('off')
plt.show()
```



In [7]:
```
x = np.hstack([np.ones((len(x), 1)), x])
```

In [8]:
```
x.shape
```

Out[8]:
```
(506, 14)
```

In [9]:
```
theta = np.linalg.inv(x.T.dot(x)).dot(x.T).dot(y)        # dot() 点乘
```

In [10]:
```
# numpy.linalg模块包含线性代数的函数。使用这个模块，可以计算逆矩阵、求特征值、解线性方程组以及求解行列式等。
```

In [11]:
```
theta
```

Out[11]:
```
array([ 3.64911033e+01, -1.07170557e-01,  4.63952195e-02,  2.08602395e-02,
        2.68856140e+00, -1.77957587e+01,  3.80475246e+00,  7.51061703e-04,
       -1.47575880e+00,  3.05655038e-01, -1.23293463e-02, -9.53463555e-01,
        9.39251272e-03, -5.25466633e-01])
```

In [12]:
```
y_predict  = x.dot(theta)
```

In [13]:
```
y_predict.shape
```

Out[13]:
```
(506,)
```

In [14]:
```
y_predict[:5]
```

Out[14]:
```
array([30.00821269, 25.0298606 , 30.5702317 , 28.60814055, 27.94288232])
```

In [15]:
```
from ML.linear import LinearRegression
from ML.model_selection import train_test_split
```

In [16]:
```
x = np.loadtxt('boston/x.txt')
y = np.loadtxt('boston/y.txt')
```

In [17]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, seed=10)
```

In [18]:
```
reg = LinearRegression()
reg.fit(x_train, y_train)
```

Out[18]:
```
<ML.linear.LinearRegression at 0x116ae0208>
```

In [19]:
```
reg.accuracy_rate(x_test, y_test)
```

Out[19]:
```
0.6696493622827584
```

In [20]:
```
x = x[:, 5]
```

In [21]:
```
x.shape
```

Out[21]:
```
(506,)
```

In [22]:
```
from ML.linear import SimpleLinearRegression
```

In [23]:
```
sreg = SimpleLinearRegression()
```

In [24]:
```
%timeit sreg.fit(x, y)
```

```
479 μs ± 23.6 μs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

In [25]:
```
x = x.reshape(-1, 1)
%timeit reg.fit(x, y)
```

```
43.4 μs ± 1.77 μs per loop (mean ± std. dev. of 7 runs, 10000 loops each)
```

上面说明：矩阵运算比循环快