Linear Regression and GD

In [271]:

```
from sklearn.datasets import load_boston
import pandas as pd
import sklearn.cross_validation
import random
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from matplotlib import pyplot as plt
```

Load and split Data

In [280]:

```
boston = load_boston()
bos = pd.DataFrame(boston.data)
bos['PRICE'] = boston.target
X = bos.drop('PRICE', axis = 1)
Y = bos['PRICE']
bos.shape
```

Out[280]:

```
(506, 14)
```

In [281]:

```
X_Train, X_Test, Y_Train, Y_Test = sklearn.cross_validation.train_test_split(X, Y, test_size = 0.3, ran
dom_state = 5)
print(X_Train.shape,X_Test.shape,Y_Train.shape,Y_Test.shape)
```

```
(354, 13) (152, 13) (354,) (152,)
```

Sk-learn LR

In [282]:

```
clf = LinearRegression()
clf.fit(X_train, Y_train)
print(clf.coef_)
print(clf.intercept_)
Y_pred = clf.predict(X_Test)
```

```
[-1.53479239e-01  4.13472741e-02 -2.48366000e-02  7.89647769e-01
 -1.30165607e+01  4.03032192e+00 -1.04163366e-02 -1.33585186e+00
  3.18055744e-01 -1.26322534e-02 -9.78680671e-01  1.28217030e-02
 -4.63297797e-01]
31.799718313818037
```

Gradient Descent

In [283]:

```
X_Train = StandardScaler().fit_transform(X_Train)
Y_Train = Y_Train[:, None]
```

In [286]:

```
iters = 10000
lr = 0.001
m = len(Y_Train)
b = 0
w = np.zeros((13,1))
for iter in range(iters):
    h = np.matmul(X_Train,w)
    loss = h - Y_Train
    CF = (np.sum(loss**2))/(2*m)
    gradient_w = (1/m)*np.matmul(X_Train.T,loss)
    gradient_b = (1/m)*np.sum(loss)
    w = w - (lr * gradient_w)
    b = b - (lr * gradient_b)
print(b)
print(w)
print(CF)
```

```
225.5621468926799
[[-1.12320466]
 [ 0.71325637]
 [-0.51857223]
 [ 0.23331179]
 [-1.09956315]
 [ 2.95292482]
 [-0.39032137]
 [-2.43151401]
 [ 1.46745459]
 [-0.86561478]
 [-2.02488144]
 [ 1.12738615]
 [-3.25113492]]
264.0543107192921
```