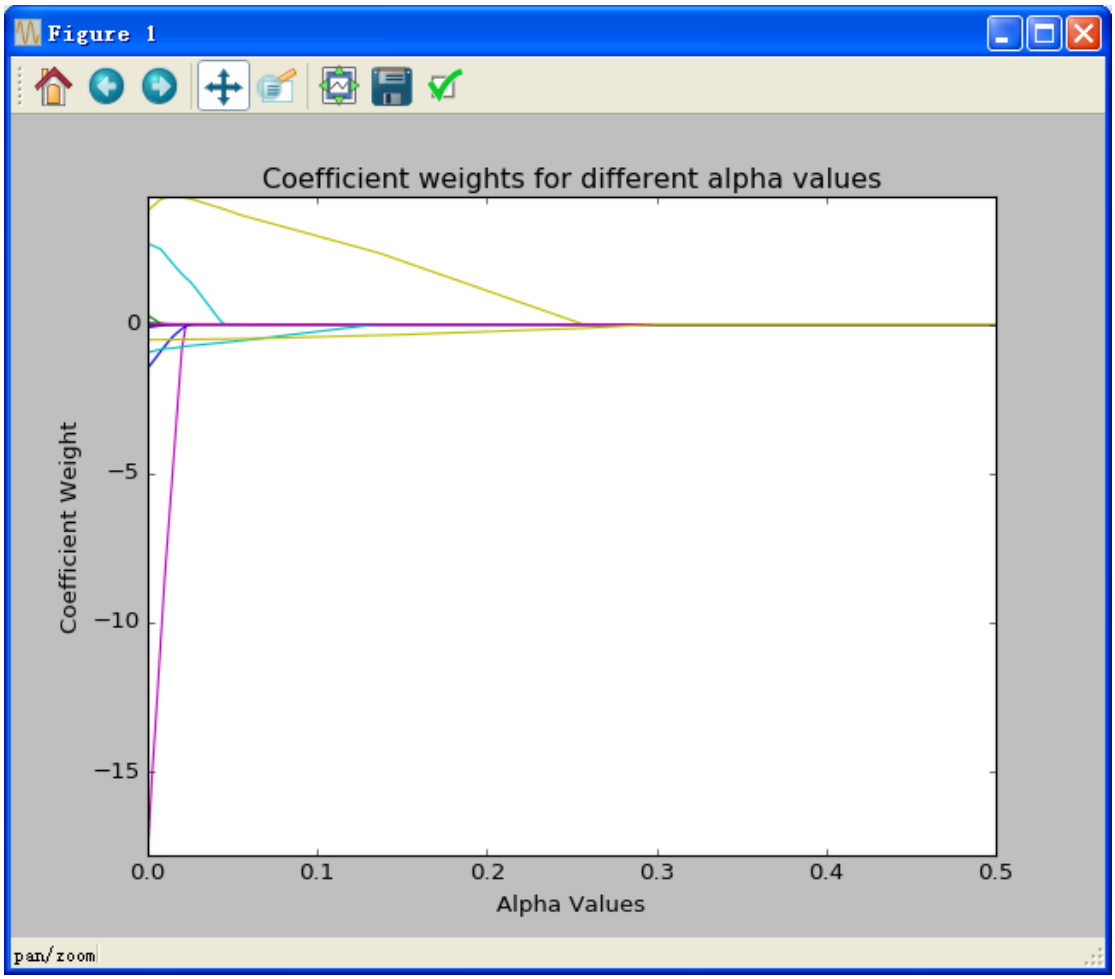
Lasso 案例

```
from sklearn.linear_model import Lasso

data = load_boston()
X = data['data']
y = data['target']

# 使用不同的 alpha 构建模型，记录各次系数
In [81]: alpha_range= np.linspace(0,0.5,200)

In [82]: model = Lasso(normalize=True)

In [83]: coefs = []

In [84]: for a in alpha_range:
    ...:     model.set_params(alpha=a)
    ...:     model.fit(X,y)
    ...:     coefs.append(model.coef_)
    ...:

# 绘制上面得到的不同 alpha 下所得模型的系数
In [85]: plt.cla()
In [86]: plt.figure(1)
In [87]: plt.xlabel("Alpha Values")
In [88]: plt.ylabel("Coefficient Weight")
In [89]: plt.title("Coefficient weights for different alpha values")
In [90]: plt.plot(alpha_range,coefs)
In [91]: plt.axis('tight')
```



可见随 alpha 增加，系数的权值总体趋于 0。

```
# 查看系数
In [92]: for i,coef in enumerate(model.coef_):
    ...:     print "Coefficient %d %0.3f" % (i+1,coef)
    ...:
Coefficient 1 -0.000
Coefficient 2 0.000
Coefficient 3 -0.000
Coefficient 4 0.000
Coefficient 5 -0.000
Coefficient 6 0.000
Coefficient 7 -0.000
Coefficient 8 0.000
Coefficient 9 -0.000
Coefficient 10 -0.000
Coefficient 11 -0.000
Coefficient 12 0.000
Coefficient 13 -0.000
```

```
In [94]: print "Intercept %0.3f" % model.intercept_
Intercept 22.533

# 简单线性回归：利用所有变量预测
In [95]: full_model = LinearRegression(normalize=True)
In [96]: full_model.fit(X,y)
In [97]: y_predicted = full_model.predict(X)
In [98]: print "MSE = %0.2f" % mean_squared_error(y,y_predicted)
MSE = 21.90

# 基于 Lasso 来选择系数：不同 alpha 上的模型
In [99]: alpha_values=[0.22,0.08,0.01]

In [101]: for a in alpha_values:
     ...:     m = Lasso(normalize=True,alpha=a)
     ...:     m.fit(X,y)
     ...:     c = m.coef_
              # 非 0 系数的索引
     ...:     indices = [i for i,coef in enumerate(c) if abs(coef)>0.0]
     ...:     print "alpha = %0.2f, Number of variables selected = %d"
                    % (a, len(indices))
     ...:     print "attributes include ",indices
     ...:     X_new = X[:,indices]   # 选择非 0 系数特征，缩减的数据集
     ...:     m1 = LinearRegression(normalize=True)
     ...:     m1.fit(X_new,y)
     ...:     y_predicted = m1.predict(X_new)
     ...:     print "MSE = %0.2f" % mean_squared_error(y,y_predicted)
     ...:
alpha = 0.22 Number of variables selected = 2
attributes include  [5, 12]
MSE = 30.51
alpha = 0.08 Number of variables selected = 3
attributes include  [5, 10, 12]
MSE = 27.13
alpha = 0.01 Number of variables selected = 9
attributes include  [0, 1, 3, 4, 5, 7, 10, 11, 12]
MSE = 22.89     # 与使用所有特征的 LR 的 21.9 相比
```