Hiệu chỉnh mô hình

- Một cách cụ thể hơn, ta sẽ tìm cách *di chuyển* nghiệm của bài toán tối ưu hàm tổn thất *tới một điểm gần nó*. Hướng di chuyển sẽ là hướng ***làm cho mô hình ít phức tạp hơn*** mặc dù giá trị của hàm tổn thất có tăng lên một chút.

Mô hình có điều chỉnh (Generalized model)

Hồi quy tuyến tính đa biến

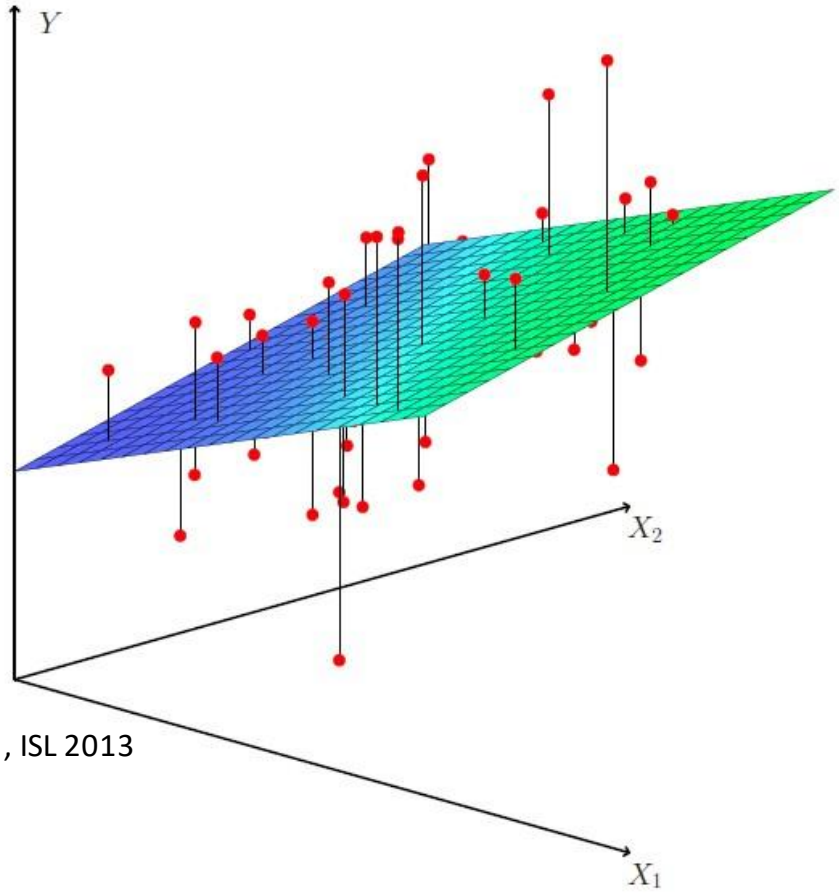


Figure3.4 , ISL 2013

$$Y = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2$$

Trường hợp phức tạp

Khi có quá nhiều biến đầu vào

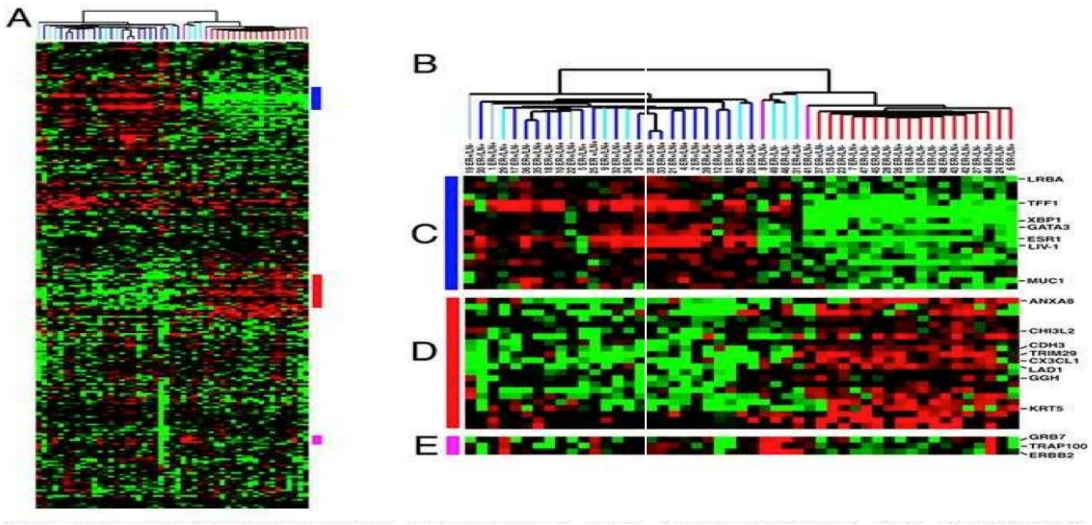
$$Y = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2 + \beta_3 \cdot X_3 + \beta_4 \cdot X_4 + \beta_5 \cdot X_5 + \beta_6 \cdot X_6 + \beta_7 \cdot X_7 + \beta_8 \cdot X_8$$

Hoặc khi có tương tác giữa các biến đầu vào

$$Y = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2 + \beta_3 \cdot (X_1 X_2) + \beta_4 \cdot X_1^2 + \beta_5 \cdot X_2^2 + \beta_6 \cdot \log(X_1 / X_2) + \beta_7 \cdot \sin(X_1 - X_2)$$

Trường hợp phức tạp

$$Y = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2 + \beta_3 \cdot X_3 + \beta_4 \cdot X_4 + \beta_5 \cdot X_5 + \beta_6 \cdot X_6 + \beta_7 \cdot X_7 + \beta_8 \cdot X_8$$



Điều gì xảy ra?

$$Y = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2 + \beta_3 \cdot X_3 + \beta_4 \cdot X_4 + \beta_5 \cdot X_5 + \beta_6 \cdot X_6 + \beta_7 \cdot X_7 + \beta_8 \cdot X_8$$

Câu hỏi: Ta có 8 biến và có hàng trăm mẫu. Hai biến (X_3 và X_4) có tương quan yếu với Y (do đó cũng có vai trò nhỏ cho dự đoán), tuy nhiên chúng có tương quan cao với các biến khác. Điều gì xảy ra khi diễn giải các hệ số β của hai biến X_3 và X_4?

Đa cộng tuyến (Multi-collinearity)

$$Y = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2 + \beta_3 \cdot X_3 + \beta_4 \cdot X_4 + \beta_5 \cdot X_5 + \beta_6 \cdot X_6 + \beta_7 \cdot X_7 + \beta_8 \cdot X_8$$

- Theo giả thiết của phương pháp Hồi quy tuyến tính thì các biến độc lập không có mối quan hệ tuyến tính.
- Nếu quy tắc này bị vi phạm thì sẽ có hiện tượng đa cộng tuyến: là hiện tượng **các biến độc lập trong mô hình phụ thuộc tuyến tính lẫn nhau** và thể hiện được dưới dạng hàm số

Đa cộng tuyến (Multi-collinearity)

Mức độ đa cộng tuyến được đo bởi hệ số VIF (Variance Inflation Factor) là tỷ lệ phương sai trong một mô hình có nhiều biến chia cho phương sai của một mô hình chỉ có một biến

- Ví dụ về đa cộng tuyến trên bộ dữ liệu Boston

crim	zn	indus	chas	nox	rm	age	dis	rad
1.87	2.36	3.90	1.06	4.47	2.01	3.02	3.96	7.80
tax	ptratio	black	lstat					
9.16	1.91	1.31	2.97					

Hồi quy tuyến tính đa biến

Quay lại hồi quy tuyến tính, ta cố gắng để cực tiểu hóa sai số bình phương

$$\sum_{\text{các mẫu}} [Y - (\beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2)]^2$$

Hiệu chỉnh mô hình (Regularization)

Hồi quy Ridge

Tìm giá trị β để cực tiểu lỗi phạt "penalized", tương đương với

$$\sum_{\text{các mẫu}} [Y - (\beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2)]^2 + \lambda \cdot (\beta_0^2 + \beta_1^2 + \beta_2^2)$$

0

L2

hoặc viết ở dạng khác,

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \left(\|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2 \right)$$

λ được gọi **là tham số hiệu chỉnh (tuning parameter)**

Hệ số hiệu chỉnh mô hình Ridge

Giá trị của λ:

- Lớn: Hầu hết các hệ số β giảm về 0 $\rightarrow$ underfitting
- Nhỏ: tương tự Hồi quy tuyến tính thông thường $\rightarrow$ overfitting
- Thường được lựa chọn bằng phương pháp đánh giá chéo (CV)

Chuẩn hóa các biến đầu vào

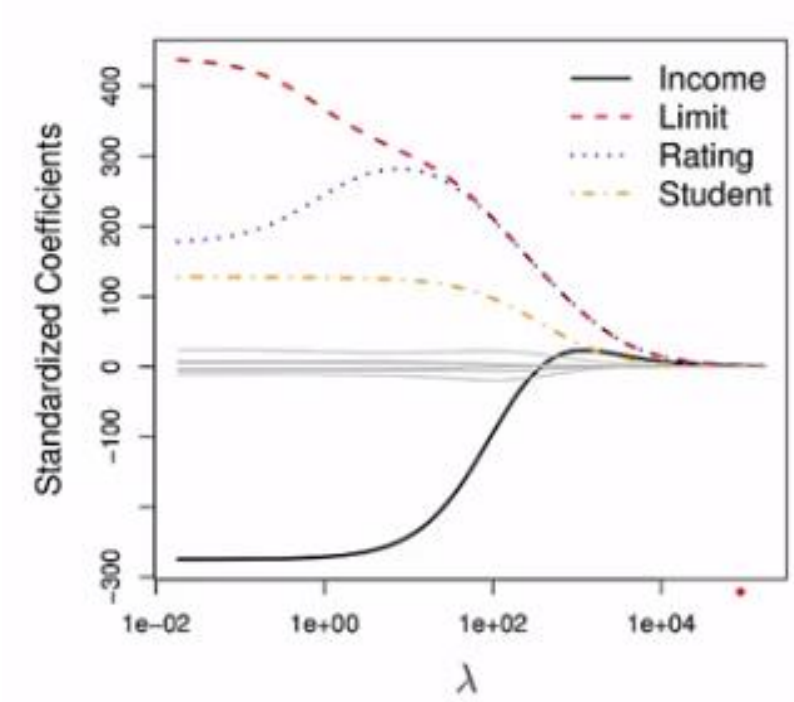
- Các biến đầu vào có đơn vị tính và khoảng giá trị khác nhau

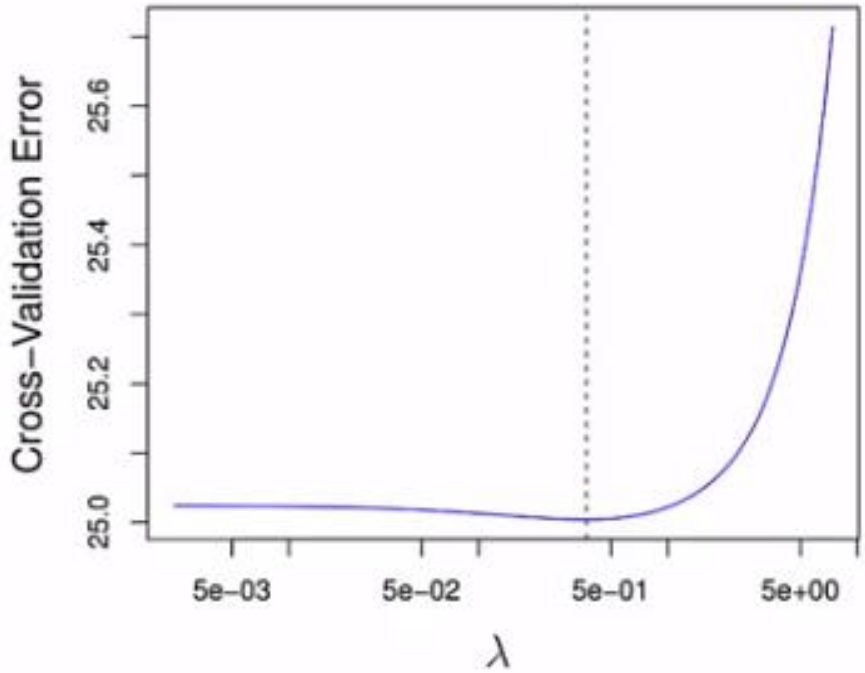
  $\rightarrow$ Các giá trị β có thể chênh lệch lớn nên cần chuẩn hóa các biến số trước khi ước lượng mô hình

- **Chuẩn hóa**:

- Lấy giá trị của mỗi biến trừ đi trung bình (mean) của nó
- Lấy kết quả nhận được chia cho độ lệch chuẩn của biến

Ví dụ: Hồi quy Ridge





Hiệu chỉnh mô hình

Ta đã xử lý:

- *Underdetermined*
- *Overfitting*
- Đa cộng tuyến (*Multi--collinearity)*

Vậy mô hình thưa là gì (*sparsity)*?

$$Y = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2 + \beta_3 \cdot X_3 + \beta_4 \cdot X_4 + \beta_5 \cdot X_5 + \beta_6 \cdot X_6 + \beta_7 \cdot X_7 + \beta_8 \cdot X_8$$

0 0 0

Mô hình thưa (Sparsity)

$$Y = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2 + \beta_3 \cdot X_3 + \beta_4 \cdot X_4 + \beta_5 \cdot X_5 + \beta_6 \cdot X_6 + \beta_7 \cdot X_7 + \beta_8 \cdot X_8$$

0 0 0

- Dùng cho lựa chọn biến (Feature selection)
- Thời gian tính toán lâu (computational efficiency)

Mô hình thưa (Sparsity)

Hồi quy Lasso

"**L**east **a**bsolute **s**hrinkage and **s**election **o**perator"

$$\sum_{samples} [Y - (\beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2)]^2 + \lambda \cdot (|\beta_0| + |\beta_1| + |\beta_2|)$$

L1

Mô hình giống như hồi quy Rigde nhưng ***khác hàm phạt***

Hiệu chỉnh mô hình

- **Hồi quy Ridge**

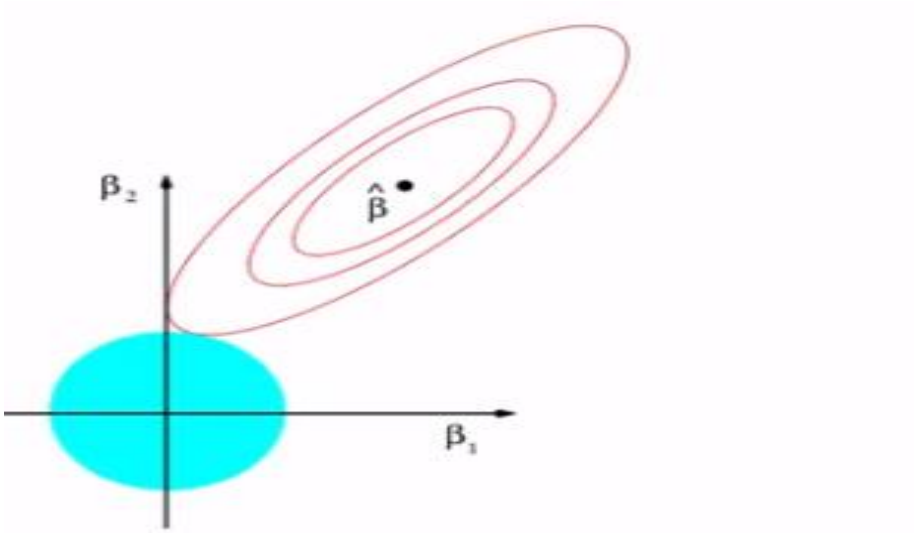
✓ Giữ lại tất cả các biến số, chỉ làm giảm các hệ số ước lượng về 0

- **Hồi quy Lasso**

✓ Giúp lựa chọn các biến có ý nghĩa, các biến số khác được gán bằng 0

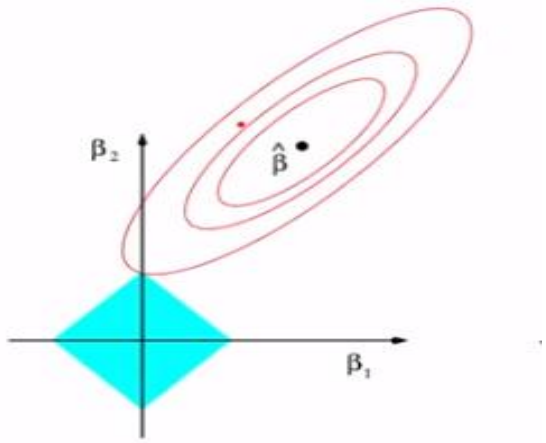
Hiệu chỉnh mô hình

- **Hồi quy Ridge**



$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2=\mathrm{RSS}+\lambda\sum_{j=1}^{p}\beta_j^2$$

- **Hồi quy Lasso**



$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|=\mathrm{RSS}+\lambda\sum_{j=1}^{p}|\beta_j|$$

Mục tiêu khác: Mô hình thưa

Lasso

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} \equiv \hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \quad s.t. \sum_{j=1}^{p} |\beta_j| \leq t$$

Ridge

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} \equiv \hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \quad s.t. \sum_{j=1}^{p} \beta_j^2 \leq t$$

Ví dụ về Hồi quy Ridge


```
from sklearn import linear_model
reg = linear_model.Ridge(alpha=0.1)
reg.fit([[0, 0], [0, 1], [1, 1]], [0, .1, 1])
print('He so w =',reg.coef_)
print('w0 =', reg.intercept_)
print('Du doan cho x =[1,0] la: ',reg.predict([[1,0]]))
```

```
He so w = [0.76223776    0.14685315]
w0 = 0.014685314685314754
Du doan cho x =[1,0] la: [0.77692308]
```

Lựa chọn tham số hiệu chỉnh

```
from sklearn import linear_model
X = [[0, 0], [0, 1], [1, 2]]
y = [0,1,2]
reg = linear_model.RidgeCV(alphas = (1,10,100,0.1,0.01,0.05))
reg.fit(X,y)
print('He so alpha = ',reg.alpha_)
reg1 = linear_model.Ridge(alpha = reg.alpha_)
reg1.fit(X,y)
y_pred1 = reg1.predict(X)
print('y du doan: ',y_pred1)
print('He so w =',reg1.coef_)
print('w0 =', reg1.intercept_)
```

```
He so alpha = 0.01
y du doan: [0.00953439 0.99074331 1.9997223 ]
He so w = [0.02777006 0.98120892]
w0 = 0.0095343885957602 8
```

Hồi quy Ridge với bộ dữ liệu home_data.csv

```
import pandas as pd
import numpy as np
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split
home = pd.read_csv("home_data.csv", sep=",")
X = home.drop("askprice", axis=1)
Y = home['askprice']
X_train, X_test, Y_train, Y_test = train_test_split(np.array(X), np.array(Y), test_size = 0.2)
# Hoi quy Ridge
ridge_linear = Ridge(alpha = 0.1)
# Training process
ridge_linear.fit(X_train, Y_train)
# Evaluating the model
print("ridge.coef_: ",ridge_linear.coef_)
print("ridge.intercept_:",ridge_linear.intercept_)
print("Training score:",ridge_linear.score(X_train, Y_train))
print("Test score:",ridge_linear.score(X_test, Y_test))
```

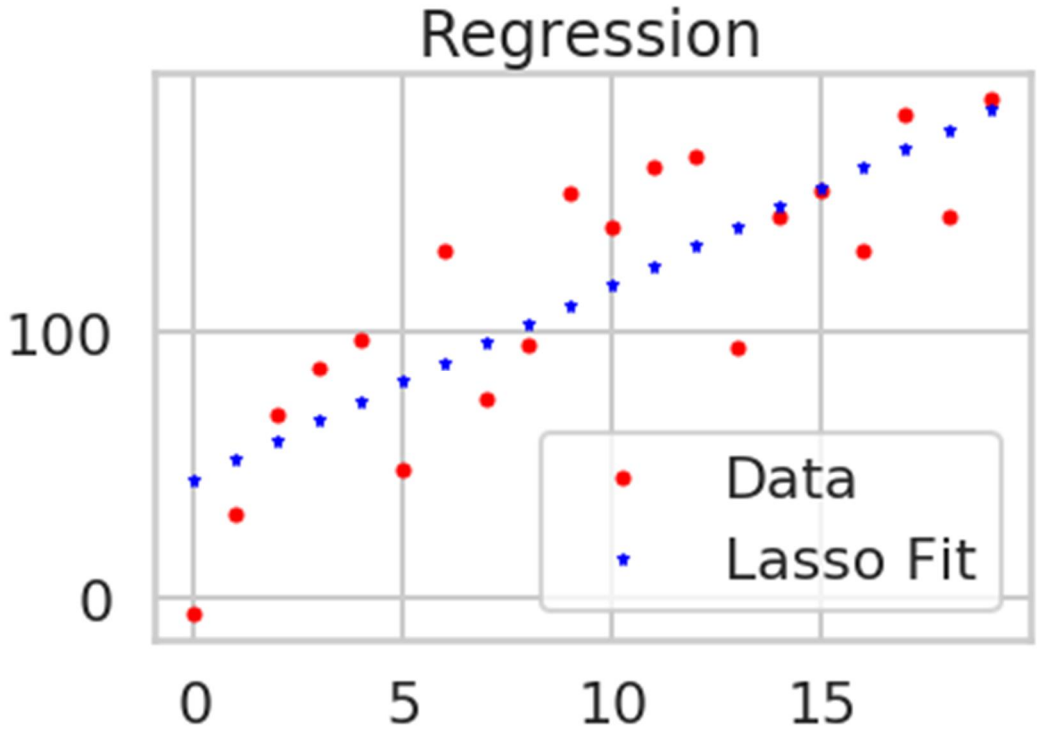
Hồi quy Lasso

```
from sklearn.linear_model import Lasso
reg = Lasso(alpha=0.1)
X = [[0, 0], [0, 1], [1, 2]]
y = [0, 1, 2]
reg.fit(X,y)
y_pred = reg.predict(X)
print('y du doan: ',y_pred)
print('He so w =',reg.coef_)
print('w0 =', reg.intercept_)
print('Du doan cho x =[1,1] la: ',reg.predict([[1, 1]]))
```

```
from sklearn import linear_model
X = [[0, 0], [0, 1], [1, 2]]
y = [0,1,2]
reg = linear_model.LassoCV(alphas = (1,10,100, 0.1,0.01,0.001,0.05))
reg.fit(X,y)
print('He so alpha = ',reg.alpha_)
reg1 = linear_model.Lasso(alpha = reg.alpha_)
reg1.fit(X,y)
y_pred1 = reg1.predict(X)
print('y du doan: ',y_pred1)
print('He so w =',reg1.coef_)
print('w0 =', reg1.intercept_)
```

Ví dụ hồi quy Lasso

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.utils import check_random_state
n = 20
x = np.arange(n)
rs = check_random_state(0)
y = rs.randint(-50, 50, size=(n,)) + 50. * np.log1p(np.arange
# Fit Lasso models
lr = linear_model.Lasso(alpha=0.001)
xx =x[:, np.newaxis] #chuyen ve vecto cot
lr.fit(xx, y)
y_pred = lr.predict(xx)
fig = plt.figure()
plt.plot(xx, y, 'ro', markersize=5)
plt.plot(xx, y_pred, 'b*',markersize=5)
plt.legend(('Data', 'Lasso Fit'), loc='lower right')
plt.title('Regression')
plt.show()
```



```
import pandas as pd
import numpy as np
from sklearn import linear_model
from sklearn.model_selection import train_test_split
home = pd.read_csv("home_data.csv", sep=",")
X = home.drop("askprice", axis=1)
Y = home['askprice']
X_train, X_test, Y_train, Y_test = train_test_split(np.ar
ray(X), np.array(Y), test_size=0.2)
```

```
linear = linear_model.LinearRegression()
linear.fit(X_train, Y_train)
score_trained = linear.score(X_test, Y_test)
print('Hoi quy tuyen tinh:',score_trained)

lasso_linear = linear_model.Lasso()
lasso_linear.fit(X_train, Y_train)
score_lasso = lasso_linear.score(X_test, Y_test)
print('Hoi quy Lasso:',score_lasso)
```

Ví dụ Hồi quy tuyến tính, Ridge, Lasso

```
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
# Chieu cao (cm)
X = np.array([[147, 150, 153, 158, 163, 165, 168, 17
0, 173, 175, 178, 180, 183]]).T
# Can nang (kg)
y = np.array([49, 50, 51, 54, 58, 59, 60, 62, 63, 64, 6
6, 67, 68])
linear = linear_model.LinearRegression()
linear.fit(X,y)
lasso_linear = linear_model.Lasso(alpha=1.0)
lasso_linear.fit(X, y)
ridge_linear = linear_model.Ridge(alpha=1.0)
ridge_linear.fit(X, y)
y_pred = linear.predict(X)
y_pred1 = lasso_linear.predict(X)
y_pred2 = ridge_linear.predict(X)
print ("y_true\n", y)
print ("y hoi quy tuyen tinh \n", y_pred)
print ("y hoi quy Lasso\n", y_pred1)
print ("y hoi quy Ridge \n", y_pred2)
plt.plot(X, y,'ro')  # data
plt.plot(X, y_pred, 'b*')
plt.plot(X, y_pred1, 'g^')
plt.plot(X, y_pred2, 'ys')
plt.xlabel('Cao (cm)')
plt.ylabel('Nang (kg)')
plt.show()
```