scikit-learn ゼミ

1.1. Generalized Linear Models
1.1.1. Ordinary Least Squares
1.1.1.1. Ordinary Least Squares Complexity
1.1.2. Ridge Regression
1.1.2.1. Ridge Complexity
1.1.2.2. Setting the regularization parameter: generalized Cross-Validation

新納浩幸

線形回帰

$\mathbf{x} = (x_1, x_2, \cdots, x_p)$　　入力

$y \in R$　　出力

$y = f(\mathbf{x})$　　f の推定を回帰という

f を線形でモデル化する

$$y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_p x_p$$

$\mathbf{w} = (w_1, w_2, \cdots, w_p)$ ← coef_

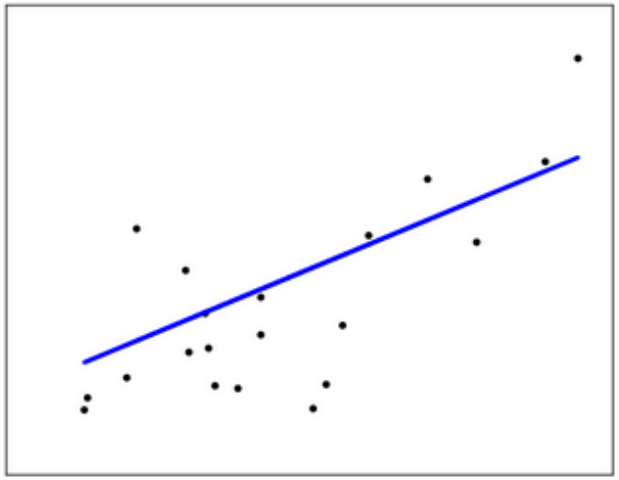
w_0 ← intercept_

Ordinary Least Squares （最小2乗法）

$$X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$$
$$\mathbf{y} = (y_1, y_2, \cdots, y_n)$$

観測データ

$$\min_{\mathbf{w}} \|X\mathbf{w} - \mathbf{y}\|^2$$



2次元の場合、直線で
モデル化と同じ

実行例

64 bit

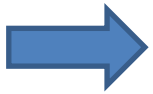
ActivePython

```
ActivePython 2.7.6.9 (ActiveState Software Inc.) based on
Python 2.7.6 (default, Feb 27 2014, 14:13:40) [MSC v.1500 64 bit (AMD64)]
Type "help", "copyright", "credits" or "license" for more information.
>>> >>> >>> >>>
>>> from sklearn import linear_model
>>> clf = linear_model.LinearRegression()
>>> clf.fit ([[0, 0], [1, 1], [2, 2]], [0, 1, 2])
LinearRegression(copy_X=True, fit_intercept=True, normalize=False)
>>> clf.coef_
array([ 0.5,  0.5])
>>> clf.intercept_
1.1102230246251565e-16
```

sklearn インストール成功

最小自乗法の注意

各次元（素性）の独立性が大事、独立性が弱いと、



multicollinearity（多重共線性）
通称「マルチコ」

外れ値に弱い、ｗの分散が大きくなる

最小自乗法の計算量

$$\mathbf{w} = (w_1, w_2, \cdots, w_p)$$

P はモデルの次数、
小さい値

$$X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$$

n はデータ数、
巨大な値

n が小さいと逆行列を求めればよいが、
巨大な値だと特異値分解する手法を使う

$$O(np^2)$$

Ridge Regression （リッジ回帰）

過学習になると係数の絶対値が大きくなる
これを抑えるために係数の２乗を罰則項に
含めた目的関数を使う

$$\min_{w}\left(\|X\mathbf{w}-\mathbf{y}\|^2+\alpha\|\mathbf{w}\|^2\right)$$

α はパラメータ、どの程度罰則を効かせるか

実行例

```
ActivePython 2.7.6.9 (ActiveState Software Inc.) based on
Python 2.7.6 (default, Feb 27 2014, 14:13:40) [MSC v.1500 64 bit (AMD64)]
Type "help", "copyright", "credits" or "license" for more information.
>>> >>> >>> >>> from sklearn import linear_model
>>> clf = linear_model.Ridge (alpha = .5)
>>> clf.fit ([[0, 0], [0, 0], [1, 1]], [0, .1, 1])
Ridge(alpha=0.5, copy_X=True, fit_intercept=True, max_iter=None,
   normalize=False, solver='auto', tol=0.001)
>>> clf.coef_
array([ 0.34545455,  0.34545455])
>>> clf.intercept_
0.13636363636363641
>>>
```

パラメータは交差検定で・・・

αの適切な値は交差検定で求める

調べるαの候補

```
>>> clf = linear_model.RidgeCV(alphas=[0.1, 1.0, 10.0])
>>> clf.fit([[0, 0], [0, 0], [1, 1]], [0, .1, 1])
RidgeCV(alphas=[0.1, 1.0, 10.0], cv=None, fit_intercept=True, gcv_mode=None,
    loss_func=None, normalize=False, score_func=None, scoring=None,
    store_cv_values=False)
>>> clf.alpha_
0.10000000000000001
```

選択されたα

この回帰の問題では α＝0 でしょう・・・