Math 4570 Matrix Methods for DA and ML

$$\vec{\theta} = (X^T X)^{-1} X^T \vec{y}$$

Section 8. Ridge and LASSO Regressions

A $\quad 0 \to$ polynomial.

B

1. Locally weighted linear regression

C

2. Ridge Regression
3. Lasso Regression
4. Elastic net Regression

Instructor: He Wang

Department of Mathematics

Northeastern University

$$\begin{bmatrix} \vec{x}^{(1)T} \\ \vdots \\ \vec{x}^{(n)T} \end{bmatrix}$$

➢ Review

$\parallel$

$(X, \vec{y})$

- Training Data $\quad D = \{(\vec{x}^{(i)}, y^{(i)}) \mid i = 1, \dots n\}$

- Linear Model Assumption: $\underline{h(\vec{x})} = \vec{\theta}^T \vec{x} = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d$

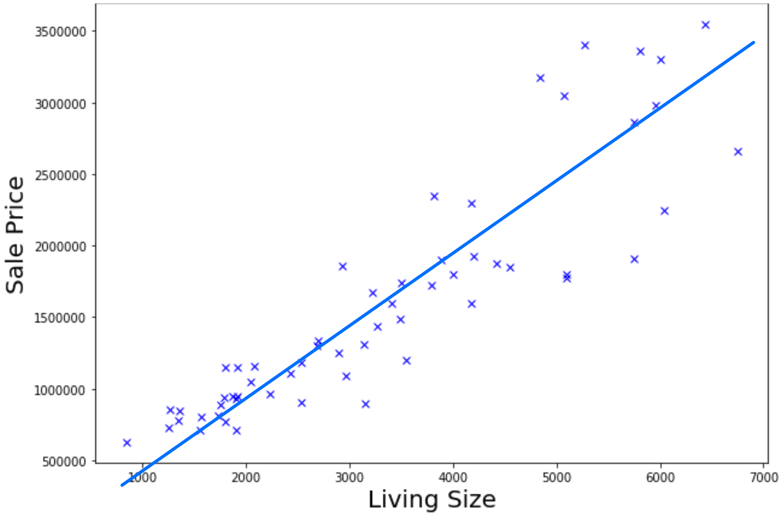
- Find $\vec{\theta}$ to minimize $\overset{\text{cost}}{\boxed{RSS(\vec{\boldsymbol{\theta}})}} = \sum_{i=1}^{n} \left(h(\vec{x}^{(i)}) - y^{(i)}\right)^2 = \|\underbrace{\boldsymbol{X}\vec{\theta}}_{\text{prediction}} - \underbrace{\vec{y}}_{\text{True}}\|^2$

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & & x_d^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_d^{(2)} \\ & \vdots & & \ddots & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} & \cdots & x_d^{(n)} \end{bmatrix} \qquad \vec{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \qquad \vec{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

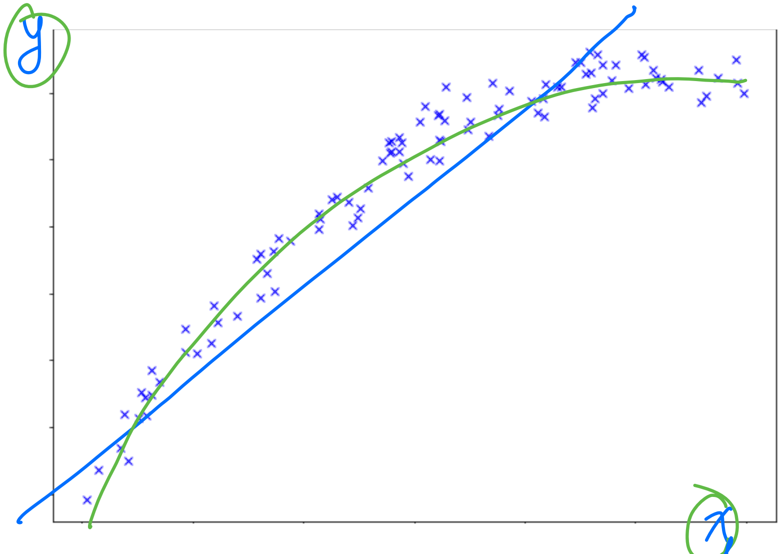
If rank X=d+1, $\underset{\vec{\theta}}{\operatorname{argmin}}\, RSS(\vec{\theta})$ is given by

Least Squares solution $\boxed{\vec{\theta} = (X^T X)^{-1} X^T \vec{y}}$

- Predict house price



Potential Disadvantage of a parametric approach: The (linear) model we choose will usually not match the true unknown form of $h(\vec{x})$. If the chosen model is too far from the true $h(\vec{x})$, then our estimate will be poor.



We can try to solve this problem by choosing flexible models that can fit many different possible functional forms flexible for $h(\vec{x})$. But in general, fitting a more flexible model requires estimating a greater number of parameters.

$$\begin{bmatrix} x_1^{(1)} \\ \vdots \\ x_1^{(n)} \end{bmatrix} = X \qquad \vec{y}$$

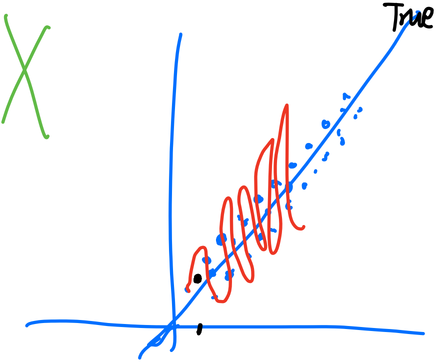
For example, New ~~assumption~~ Model:

$$h(x_1) = \underline{\theta_0} + \underline{\theta_1} \overset{x_1 =}{x_1} + \underline{\theta_2} \overset{x_2 =}{x_1^2} + \underline{\theta_3} \overset{x_3 =}{\sqrt{x_1}} + \underline{\theta_4} \overset{x_4 =}{\log x_1}$$

$$\vec{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}_{\text{opt.}} = (X^T X)^{-1} X^T \vec{y}$$

x_0	x_1	x_2	x_3	x_4
1	$x_1^{(1)}$	$x_2^{(1)} = x_1^{(1)2}$	$x_3^{(1)} = \sqrt{x_1^{(1)}}$	$\log x_1^{(1)}$
1	$x_1^{(2)}$	$x_2^{(2)} = x_1^{(2)2}$	$x_3^{(2)} = \sqrt{x_1^{(2)}}$	$\log x_1^{(2)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

$= X$



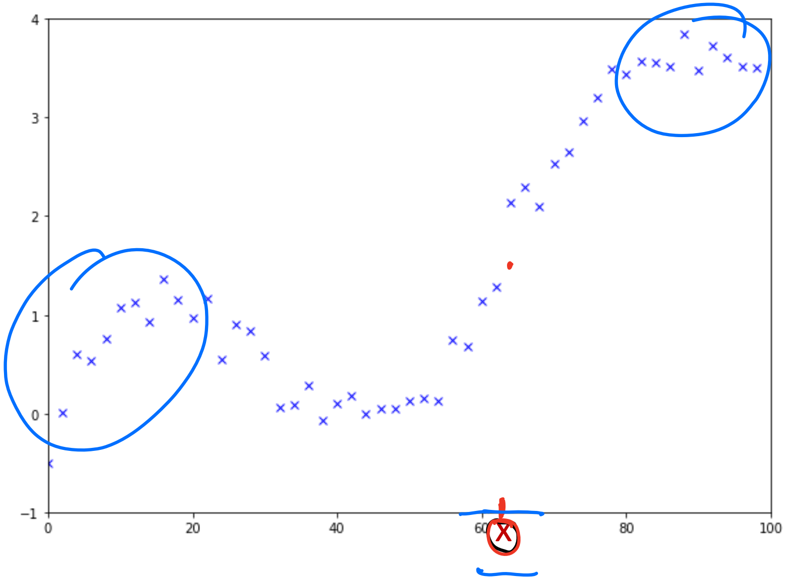
Suppose $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, we can try polynomial model using new features

$$x_1, x_2, \; x_1^2, x_2^2, x_1x_2, \; x_1^3, x_2^3, x_1^2x_2, x_1x_2^2, \; x_1^4, x_2^4, x_1^3x_2, x_1^2x_2^2, x_1x_2^3, \ldots$$

Potential Disadvantage: overfitting the data.
These more complex models can lead to a phenomenon that they follow the errors, or noise, too closely.

- **Locally weighted regression**

Goal: Evaluate h **at certain x**

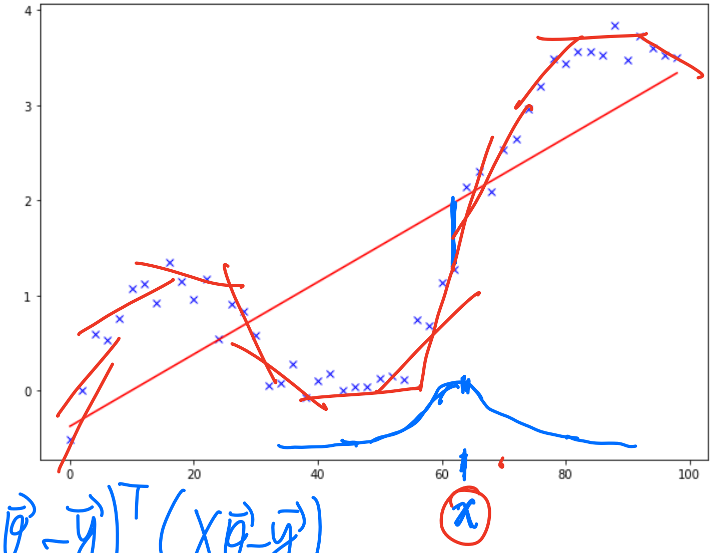


- Recall Linear Regression: Find

$$h(\vec{x}) = \vec{\theta}^T \vec{x} = \theta_0 + \theta_1 x_1$$

to minimize the RSS cost function:

$$RSS(\vec{\theta}) = \sum_{i=1}^{n} \left(h\left(\vec{x}^{(i)}\right) - \vec{y}^{(i)}\right)^2$$



$$= \| X\vec{\theta} - \vec{y} \|^2 = (X\vec{\theta} - \vec{y})^T (X\vec{\theta} - \vec{y})$$

$D = \{(\vec{x}^{(i)}, y^{(i)})\}$

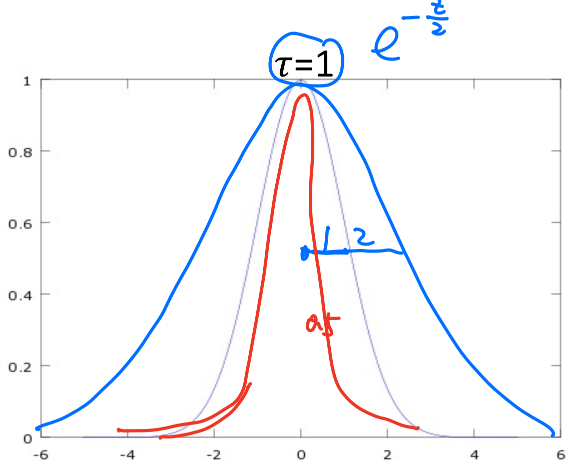
- Minimize new weighted cost function

$$RSS_W(\vec{\theta}) = \boxed{\sum_{i=1}^{n} \underbrace{w^{(i)}}_{\text{weight}} \left(h(\vec{x}^{(i)}) - \vec{y}^{(i)}\right)^2}$$

Here, $w^{(i)} = \exp\left(-\frac{\|\vec{x}^{(i)} - \vec{x}\|^2}{2\tau^2}\right) = \begin{cases} 1 & \text{if } \vec{x}^{(i)} = \vec{x} \\ \to 0 & \text{if } \|\vec{x}^{(i)} - \vec{x}\| \to \infty \end{cases}$

$$J(\vec{\theta}) = RSS_W(\vec{\theta}) = (X\vec{\theta} - \vec{y})^T W (X\vec{\theta} - \vec{y})$$

$$= \|X\vec{\theta} - \vec{y}\|_W^2$$



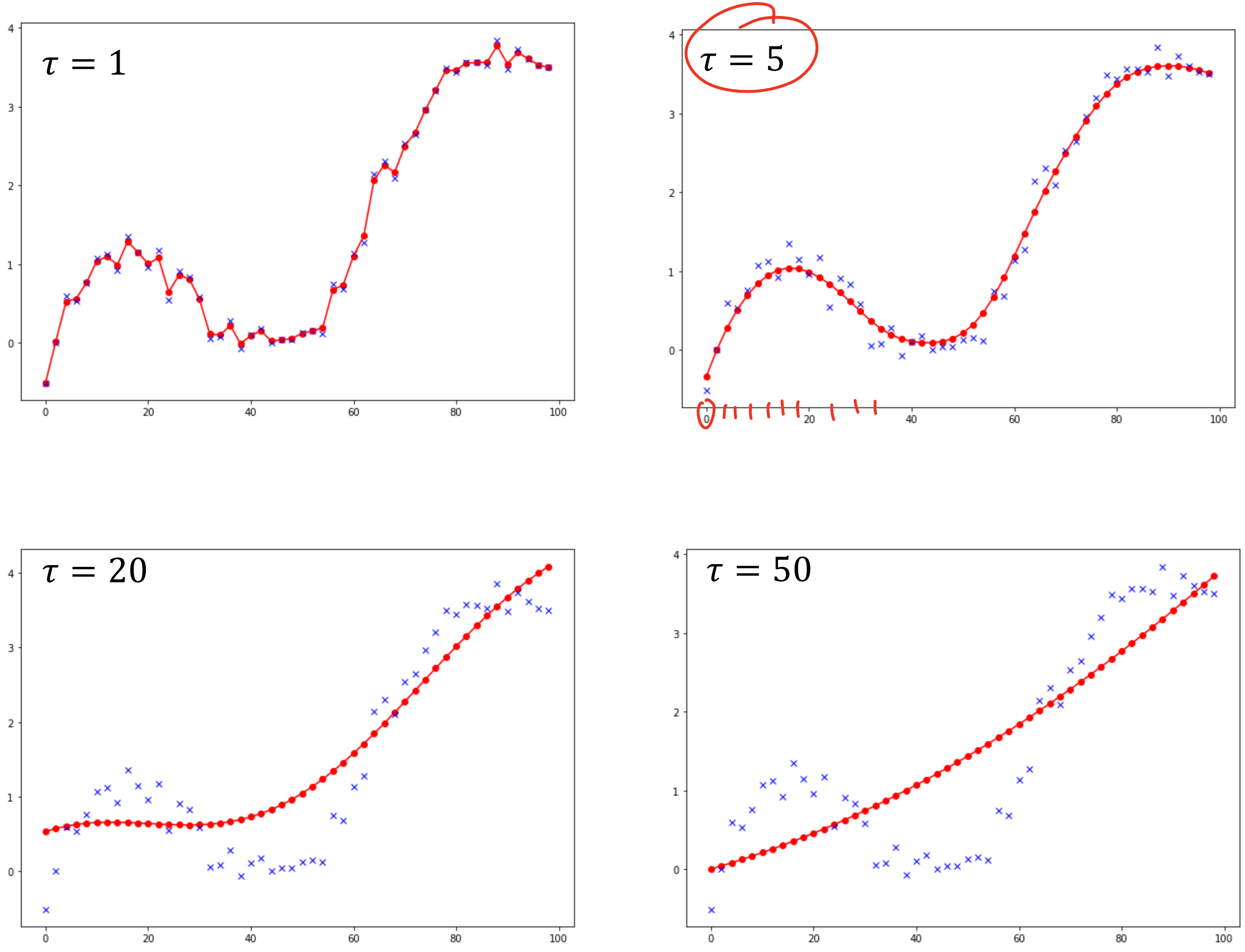
$$f(z) = \exp\left(-\frac{z}{2\tau^2}\right)$$

$$W = \begin{bmatrix} w^{(1)} & & & \\ & w^{(2)} & & \\ & & \ddots & \\ & & & w^{(n)} \end{bmatrix}$$

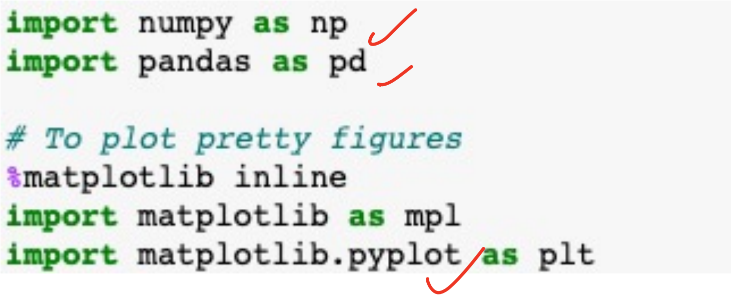
- **Claim:** $\nabla_{\vec{\theta}} J = 2X^T W (X\vec{\theta} - \vec{y}) = 0$

- $X^T W X \vec{\theta} = X^T W \vec{y}$

$\vec{\theta} = (X^T W X)^{-1} X^T W y$

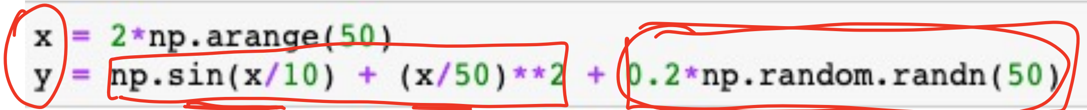


We need the training data as well as the parameters to make a prediction.

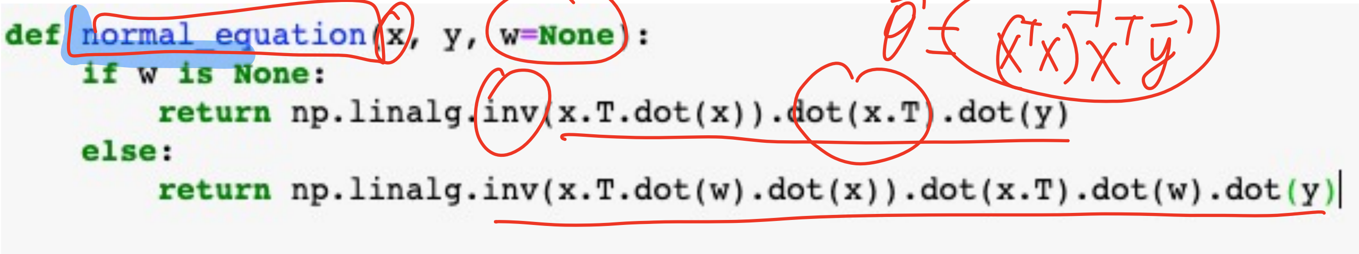


```
import numpy as np
import pandas as pd

# To plot pretty figures
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
```

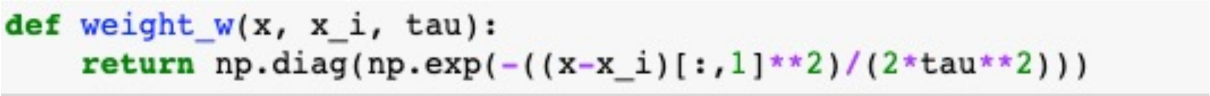


```
x = 2*np.arange(50)
y = np.sin(x/10) + (x/50)**2 + 0.2*np.random.randn(50)
```

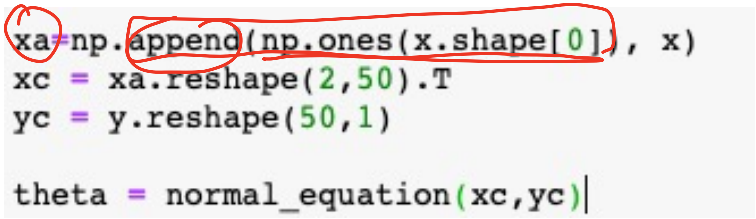


```
def normal_equation(x, y, w=None):
    if w is None:
        return np.linalg.inv(x.T.dot(x)).dot(x.T).dot(y)
    else:
        return np.linalg.inv(x.T.dot(w).dot(x)).dot(x.T).dot(w).dot(y)
```

$\vec{\theta} = (X^T X)^{-1} X^T \vec{y}$

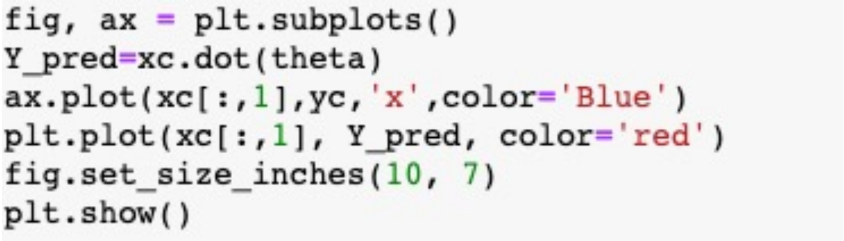


```
def weight_w(x, x_i, tau):
    return np.diag(np.exp(-((x-x_i)[:,1]**2)/(2*tau**2)))
```

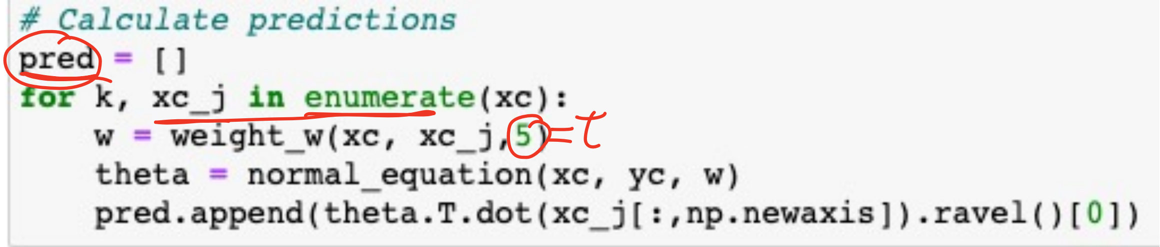


```
xa=np.append(np.ones(x.shape[0]), x)
xc = xa.reshape(2,50).T
yc = y.reshape(50,1)

theta = normal_equation(xc,yc)
```

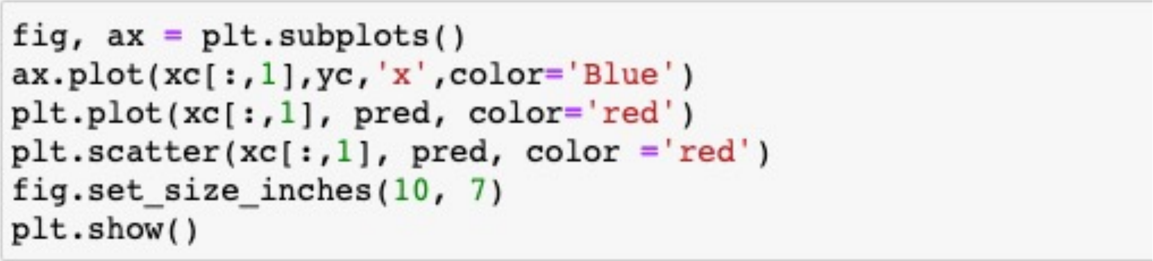


```
fig, ax = plt.subplots()
Y_pred=xc.dot(theta)
ax.plot(xc[:,1],yc,'x',color='Blue')
plt.plot(xc[:,1], Y_pred, color='red')
fig.set_size_inches(10, 7)
plt.show()
```



```
# Calculate predictions
pred = []
for k, xc_j in enumerate(xc):
    w = weight_w(xc, xc_j,5)
    theta = normal_equation(xc, yc, w)
    pred.append(theta.T.dot(xc_j[:,np.newaxis]).ravel()[0])
```

= τ



```
fig, ax = plt.subplots()
ax.plot(xc[:,1],yc,'x',color='Blue')
plt.plot(xc[:,1], pred, color='red')
plt.scatter(xc[:,1], pred, color ='red')
fig.set_size_inches(10, 7)
plt.show()
```

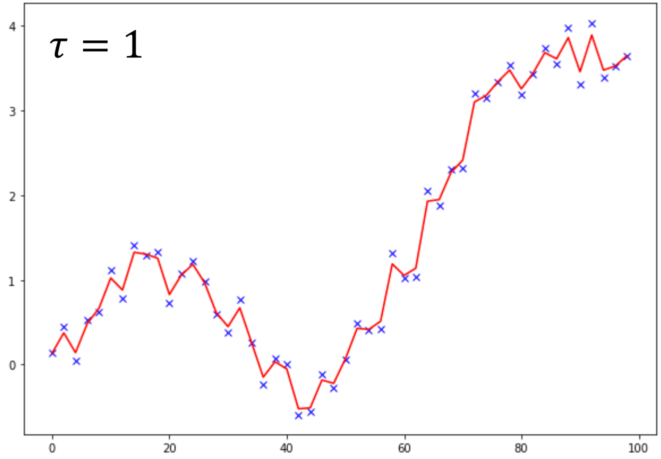
Given training Data $D = \{(\vec{x}^{(i)}, y^{(i)}) \mid i = 1, \ldots n\}$

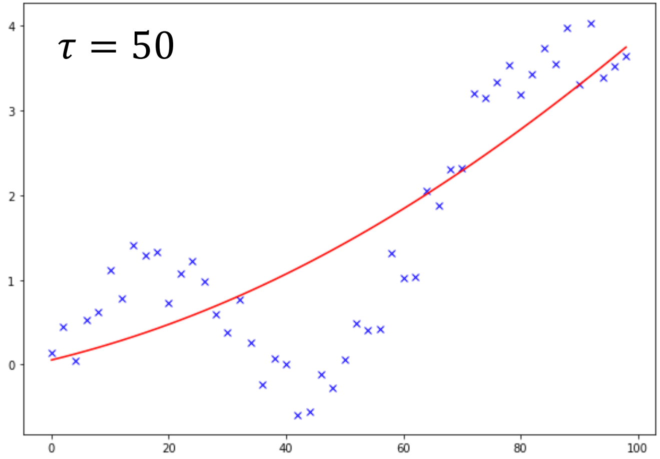
$\theta_0 + \theta_1 X_1 + \cdots + \theta_d X_d$

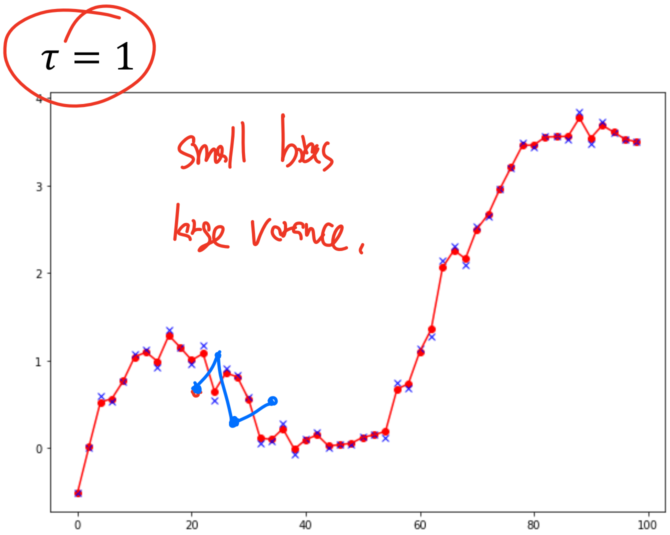
//

A **parametric** inference method gives a predictor function $h_{\vec{\theta}}(\vec{x})$ with a finite number of parameters $\theta_0, \theta_1, \ldots, \theta_d$.

A **non-parametric** inference method estimates without parameters, or with an infinite number of parameters.

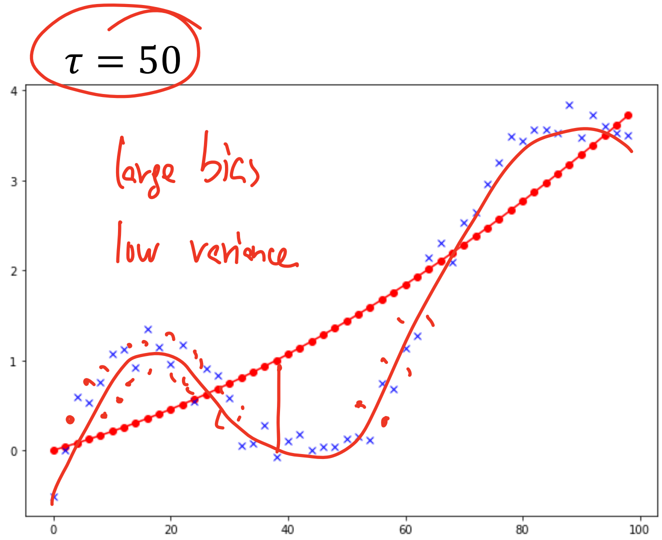






small bias

large variance.



large bias

low variance

$$\boxed{\text{Total error}} = \underline{\text{bias}^2 + \text{Variance}} + \underline{\text{noise}}$$

➢ Bias-Variance Trade Off (first view)

When $\tau = 1$, if we change the training data from the same model and recompute the linear regression(or classifier), the **change** in the fit is very high.

By contrast, when $\tau = 50$, if we change the training data from the same model and recompute the linear regression, the **change** in the fit is very low.

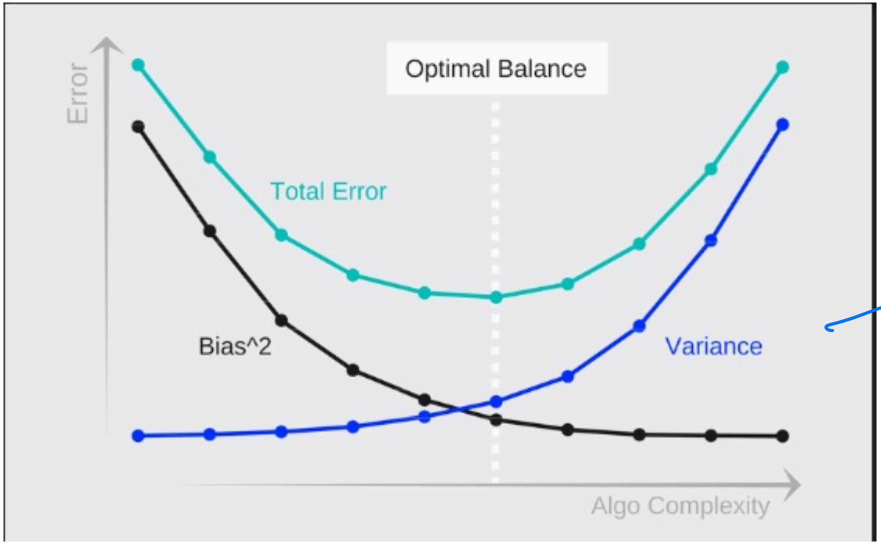
This contrast between the two algorithms is know as the **bias-variance trade off.**

For a class of models, the **bias** roughly is the expected error of the best model (regression or classifier) in the model class given a random set of training data. (This part of the generalization error is due to wrong assumptions.)

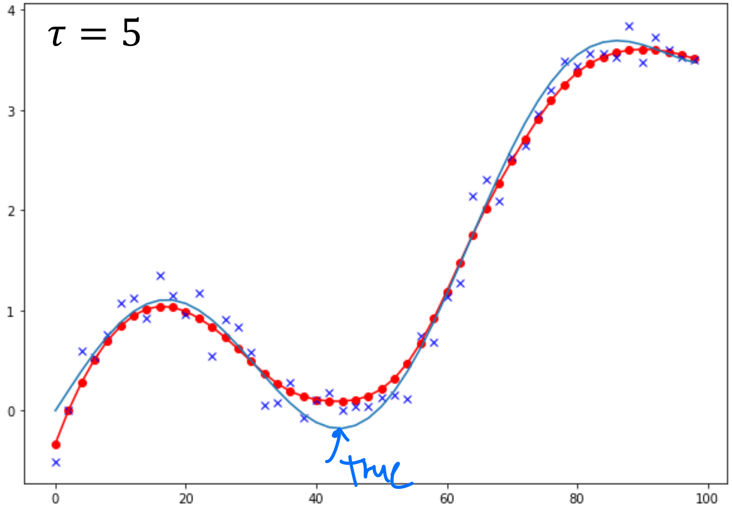
The **variance** is roughly the sensitivity of the model to the training data. (This part is due to the model's excessive sensitivity to small variations in the training data.)

$$\text{Total Error} = (\text{Bias})^2 + \text{Variance} + \text{Irreducible Error}$$

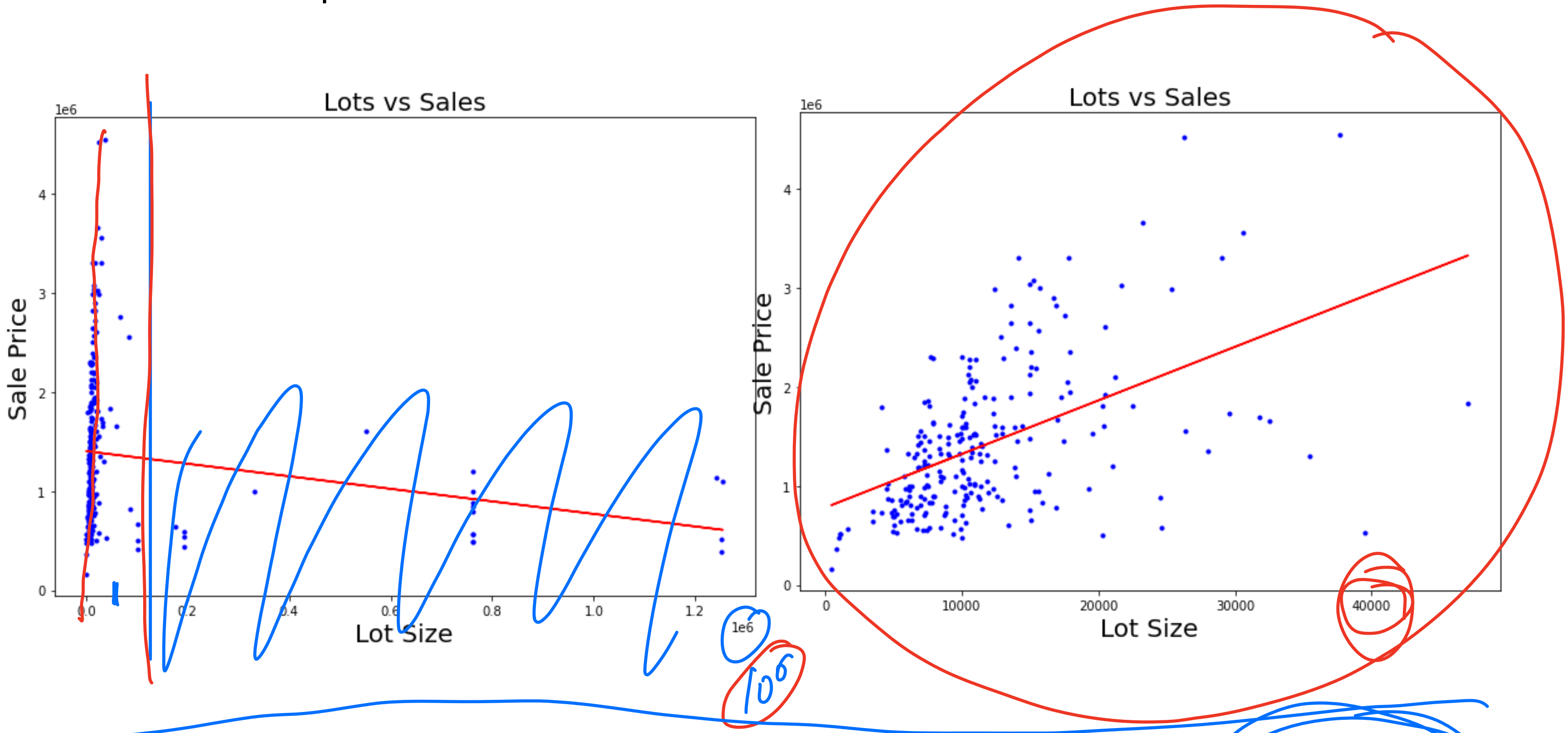
Irreducible Error is due to the noisiness of the data itself. The only way to reduce this part of the error is to clean up the data.



The best fit lies somewhere in between the extreme ends.



Remark: Clean up the data.



0, 3, 6

2, 3, 4

$\frac{2+3+4}{3} = \text{mean} = 3$

$\sqrt{\frac{2^2+3^2+4^2}{3}}$

$\sqrt{\frac{1}{n} RSS(\vec{\theta})}$

$\{(\vec{x}^{(i)}, y^{(i)}) \mid i = 1, \cdots, n\}$

$\vec{x}^{(i)} \in \mathbb{R}^d$

Feature selection (Shrinkage/regularization methods)

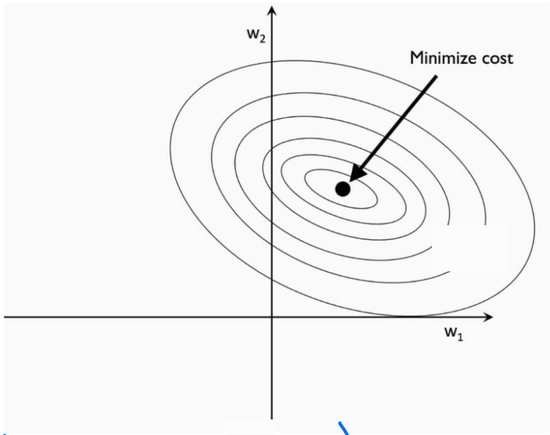
Data: ~~$h(\vec{x}) = \vec{\theta}^T\vec{x} = \theta_0 + \theta_1 x_1 + \cdots +$~~ $\theta_d x_d$

1000 997

Model: $h(\vec{x}) = \vec{\theta}^T\vec{x} = \theta_0 + \overset{2}{\theta_1 x_1} + \overset{1}{\theta_i x_i} + \cdots + \theta_d x_d$

Square sum: L_2

$$\mathrm{RSS}(\vec{\theta}) = \sum_{i=1}^{n}\left(h(\vec{x}^{(i)}) - y^{(i)}\right)^2$$



Ridge Regression

$$\vec{x}^i = \begin{bmatrix} x_1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}$$

suppose $x_1 \approx x_2$

$\lambda \geq 0$

complexity parameter (hyper-parameter)

Ridge regression cost function:

$$J^{Ridge}(\vec{\theta}) = \sum_{i=1}^{n}\left(y^{(i)} - h_\theta(\vec{x}^{(i)})\right)^2 + \lambda \sum_{j=1}^{d} {\theta_j}^2$$

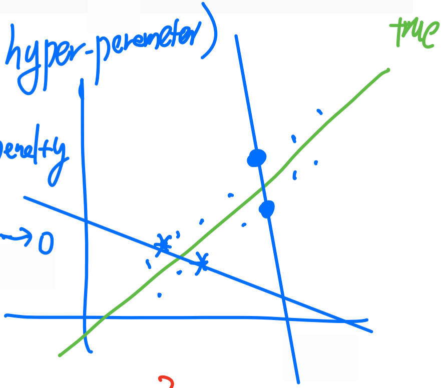
$\lambda = 0$ no penalty

$\lambda = +\infty$ $\theta_j \to 0$

$= \mathrm{RSS}(\vec{\theta}) + \text{penalty}$

$\|\vec{y} - X\vec{\theta}\|^2 + \lambda\|\vec{\theta}\|^2$

suppose $\theta_0 = 0$



- Goal: Find $\underset{\vec{\theta}}{\arg\min}\, J^{Ridge}(\vec{\theta})$

level curve for RSS($\vec{\theta}$)

$$(X^TX)^{-1}X^T\vec{y} = \underset{\vec{\theta}}{\arg\min}\ RSS(\vec{\theta})$$

Lagrange Multiplier method (Karush–Kuhn–Tucker)

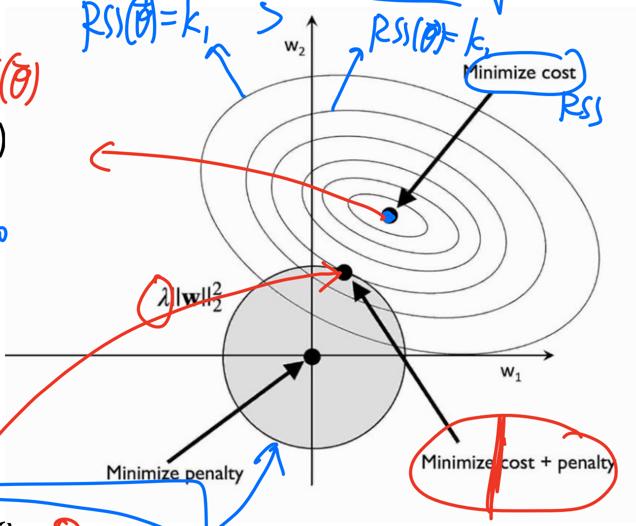
$\to\infty$ $\theta_j \to 0$

Minimize the Ridge cost function

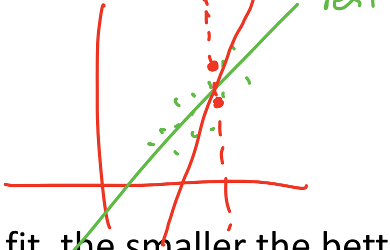
easier $\to$ $$J^{Ridge}(\vec{\theta}\,) = \boxed{\text{RSS}(\vec{\theta}) + \lambda \sum_{j=1}^{d} {\theta_j}^2}$$

is equivalent to $\boxed{\text{minimize RSS}(\vec{\theta})}$ subject to $\sum_{j=1}^{d} {\theta_j}^2 - t \leq 0$

hard



$$\underset{\vec{\theta}}{\arg\min}\ J^{Ridge}(\vec{\theta})$$



➢ Remarks on Ridge Regression

$RSS(\vec{\theta})$

1. The first term measures **goodness** of fit, the smaller the better.

2. The second term is called **shrinkage penalty**, which shrinkage θ_i towards to 0.

3. The shrinkage **reduces variance** (at the cost increased bias). Ridge regression works best in situations where the least squares estimates have high variance.

4. The intercept θ_0 is not penalized.

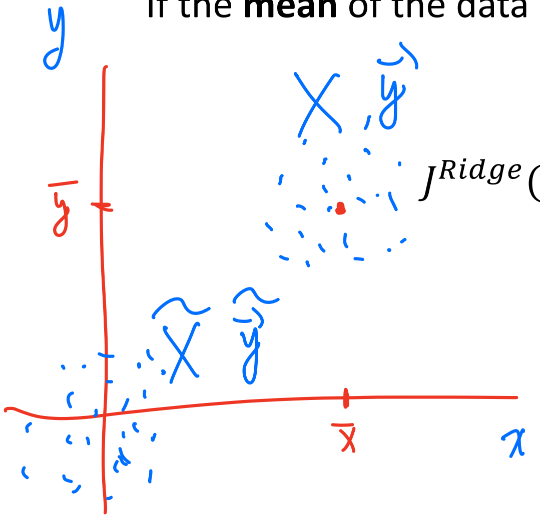
5. Ridge Regression is affected by the scale. (Least squares solution is unaffected by the scale.)

6. Ridge regression also has substantial computational advantages.

$$\nabla_{\vec{\theta}} J^{ridge} = 0 \Longrightarrow \vec{\theta} = (X^T X + \lambda I)^{-1} X^T \vec{y}$$

assume $\theta_0 = 0$

If the **mean** of the data matrix X is **zero**, then $\theta_0 = 0$, in this case,



$$J^{Ridge}(\vec{\theta}\,) = \sum_{i=1}^{n}\left(y^{(i)} - h_\theta(\vec{x}^{(i)})\right)^2 + \lambda\sum_{j=1}^{d}\theta_j{}^2$$

$$= (X\vec{\theta} - \vec{y})^T(X\vec{\theta} - \vec{y}) + \lambda\vec{\theta}^T\vec{\theta}$$

$\vec{x} \rightarrow y$

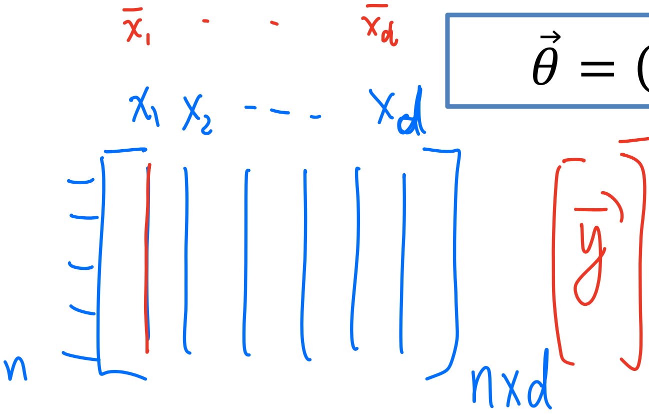
Data $(X, \vec{y})$

$$mean(\vec{x}) = \begin{bmatrix} \bar{x}_1 \\ \vdots \\ \bar{x}_d \end{bmatrix}$$

Calculate $\nabla_{\vec{\theta}} J = 0$, we get

$$\boxed{\vec{\theta} = (X^TX + \lambda I)^{-1}X^T\vec{y}}$$

$$\begin{cases} \tilde{\vec{x}} = \tilde{\vec{x}} - mean(\vec{x}) \\ \tilde{y} = y - mean(\vec{y}) \end{cases}$$



$$\tilde{X} = X - \begin{bmatrix} mean(\vec{x})^T \\ \vdots \\ mean(\vec{x})^T \end{bmatrix}$$

$\tilde{y} = \bar{y} - \begin{bmatrix}\bar{y}\\ \vdots \\ \bar{y}\end{bmatrix}$

Remark: (Standardization of feature variables/ Feature Scaling)

Before applying the Ridge/Lasso/ Elastic net regressions, we need to rescale an original variable to have equal range or variance.

1. Min-max scaling/**normalization**/ 0-1 scaling (Scikit-Learn: MinMaxScaler)

$$\frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)}$$

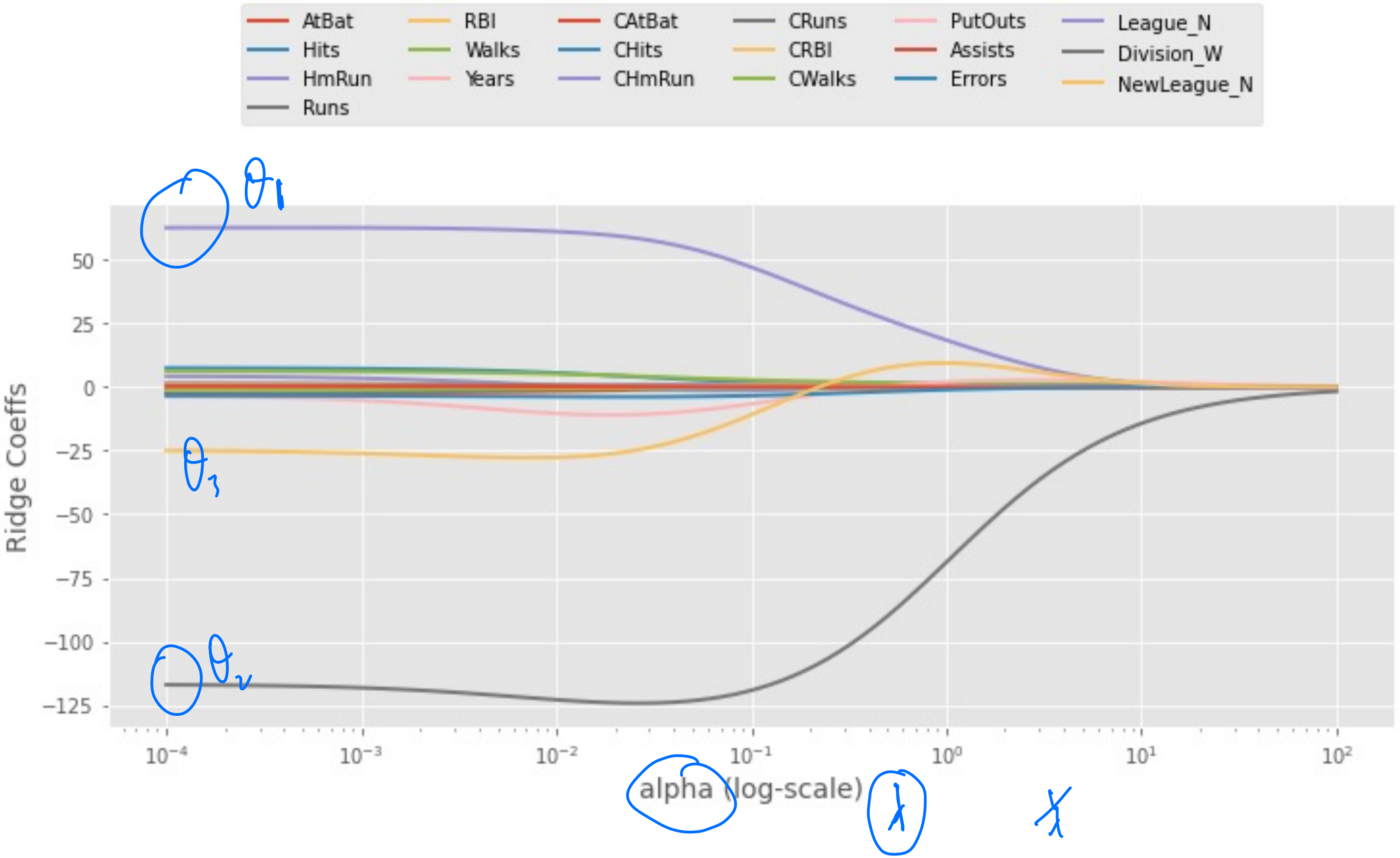
2. **Standardization** (Scikit-Learn: StandardScaler)

$$\frac{x_i - mean(x_i)}{s(x_i)}$$

$$= \bar{x}_i = \frac{\sum_{j=1}^{n} x_i^{(j)}}{n}$$

where $s(x)$ is the standard deviation of x.

$$= \sqrt{\frac{1}{n-1}\left(x_i^{(j)} - \bar{x}_i\right)^2}$$


AtBat
Hits
HmRun
Runs
RBI
Walks
Years
CAtBat
CHits
CHmRun
CRuns
CRBI
CWalks
PutOuts
Assists
Errors
League_N
Division_W
NewLeague_N
Ridge Coeffs
50
25
0
−25
−50
−75
−100
−125
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
alpha (log-scale)
θ_1
θ_3
θ_2
λ
λ

C1

➢ Lasso Regression

model

$h(\vec{x}) = \theta_0 + \theta_1 X_1 + \theta_2 X_2$

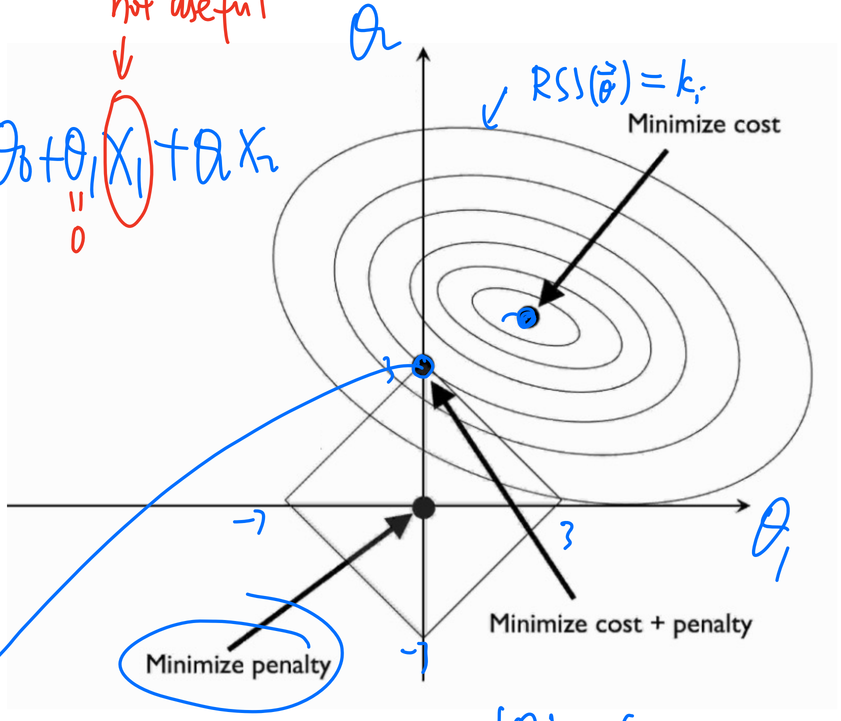
$\theta_1 = 0$

not useful

Lasso regression cost function:

$$J^{Lasso}(\vec{\theta}\,) = \underline{\text{RSS}(\vec{\theta})} + \lambda \boxed{\sum_{j=1}^{d} |\theta_j|}$$

penalty



$|\theta_1| + |\theta_2| = 3$

Minimize the Lasso cost function

is equivalent to minimize RSS($\vec{\theta}$) subject to $\sum_{j=1}^{d} |\theta_j| - \text{t} \leq 0$

$$\vec{\theta}^{Lasso} = \underset{\theta}{\text{argmin}}\, J^{Lasso}(\vec{\theta}) = \underset{\theta}{\text{argmin}}\, RSS(\vec{\theta}) \text{ subject to } \boxed{\sum_{j=1}^{d} |\theta_j| - \text{t} \leq 0}$$

- Remarks on Lasso and Ridge Regressions

0. Lasso stands for Least Absolute Shrinkage and Selection Operator.

1. Lasso tends to completely eliminate the weights of the least important features. (It performs variable selection, and yields sparse models.) Hence, Lasso is more interpretable than ridge.

2. The lasso implicitly assumes that a number of the coefficients truly equal zero.

3. Ridge regression outperforms the lasso in terms of prediction error.

4. Both ridge and Lasso can improve over the traditional least squares by trade off variance with bias.

5. There are significant improvement when the variance of the least squares is large, mostly with small n and large d.

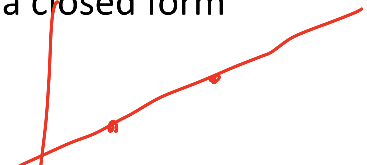
more data than features.

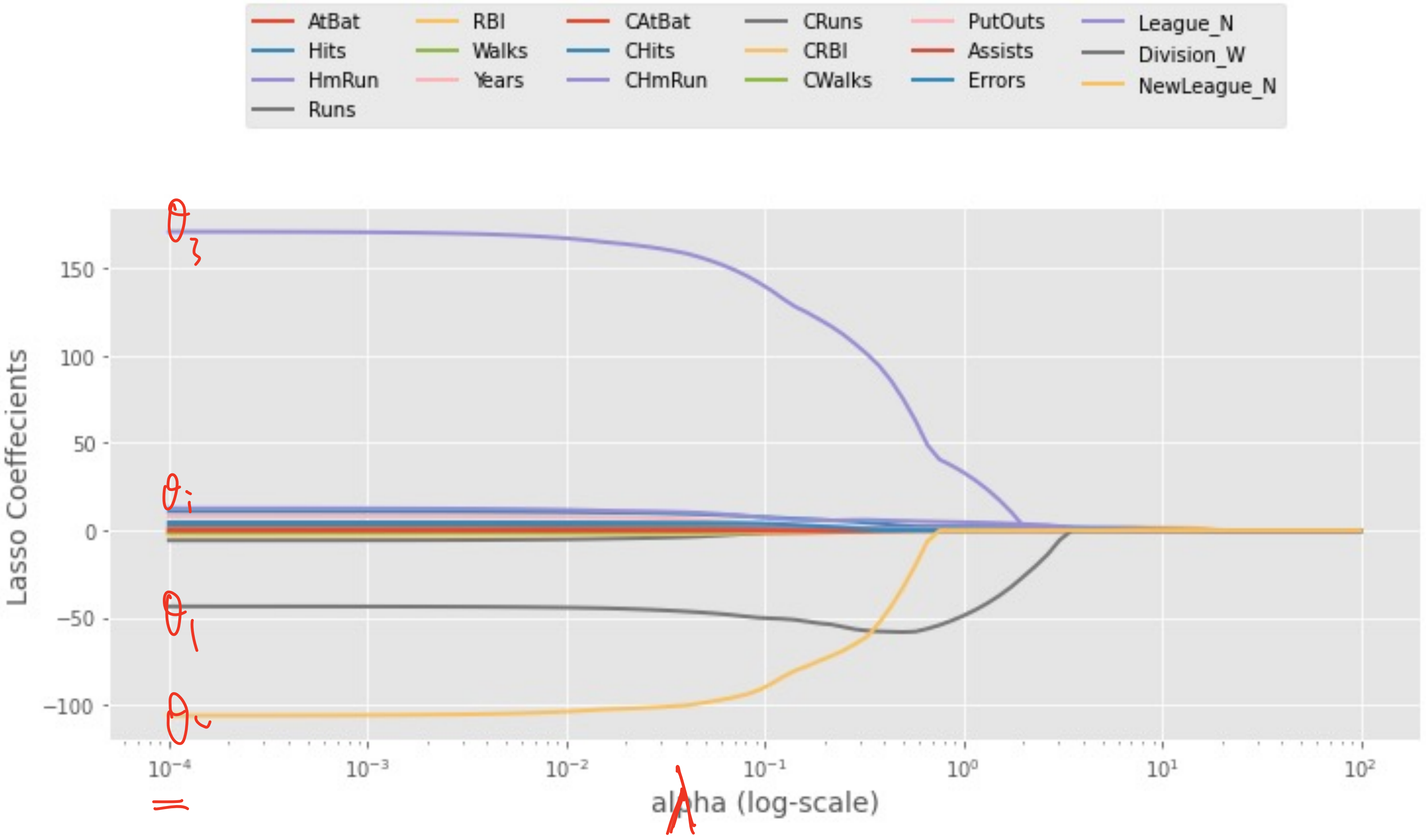
$$\underline{\underline{X^T X}}\,\vec{\theta} = \underline{\underline{X^T \vec{y}}}$$

6. Lasso has **feature selection**, while ridge does not.

7. Use **cross validation** to determine which one has better prediction.

8. Ridge has closed form solution. Lasso generally does not have a closed form solution.




AtBat
Hits
HmRun
Runs
RBI
Walks
Years
CAtBat
CHits
CHmRun
CRuns
CRBI
CWalks
PutOuts
Assists
Errors
League_N
Division_W
NewLeague_N
Lasso Coeffecients
150
100
50
0
−50
−100
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
alpha (log-scale)
θ_3
θ_i
θ_1
θ_u
=
λ

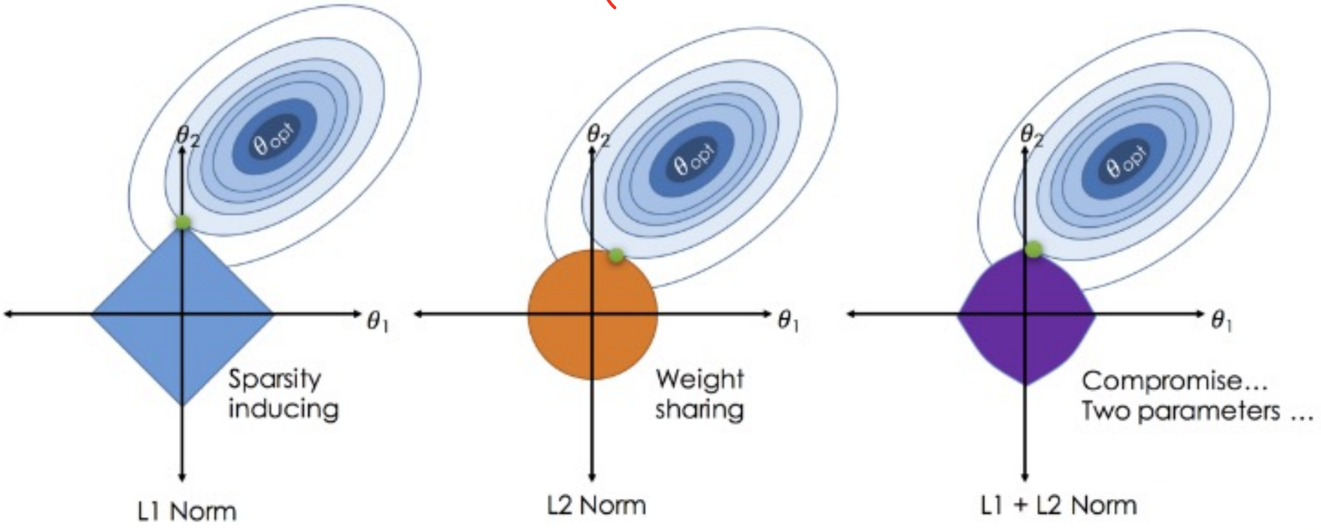
➢ Elastic net Regression

Sci-kit or Statmodels

Elastic net regression cost function:

$$J(\vec{\theta}\,) = \mathrm{RSS}(\vec{\theta}) + \lambda \sum_{j=1}^{d} |\theta_j| + \eta \sum_{j=1}^{d} {\theta_j}^2$$

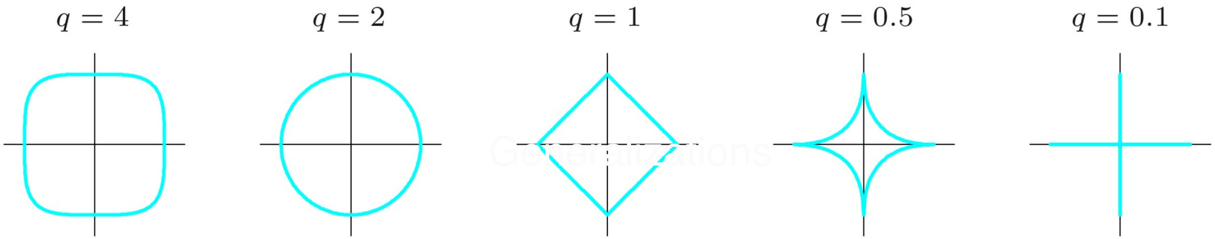
$$\alpha\left((1-\lambda)\sum + \lambda\sum \quad\right)$$



Generalizations: For any positive number q.

L_q

$$J(\vec{\theta}\,) = \mathrm{RSS}(\vec{\theta}) + \lambda \sum_{j=1}^{d} \underline{\underline{|\theta_j|}}^{\boxed{q}}$$



We introduced a few different statistical learning methods.
Which one is the best approach?

George Box 1987: “All models are wrong, but some are useful.” (No Free Lunch Theorem) Hence, no one method dominates all others over all possible data sets. On a particular data set, one specific method may work best, but some other method may work better on a similar but different data set.

It is an important task to decide for any given set of data which method produces the best results. Selecting the best approach can be one of the most challenging parts of machine learning in practice. (Project?)