Section 3. Ridge and LASSO Regressions

1. Locally weighted linear regression
2. Interpretation in Probability
3. Ridge Regression
4. Lasso Regression
5. Elastic net Regression

➢ Review

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

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

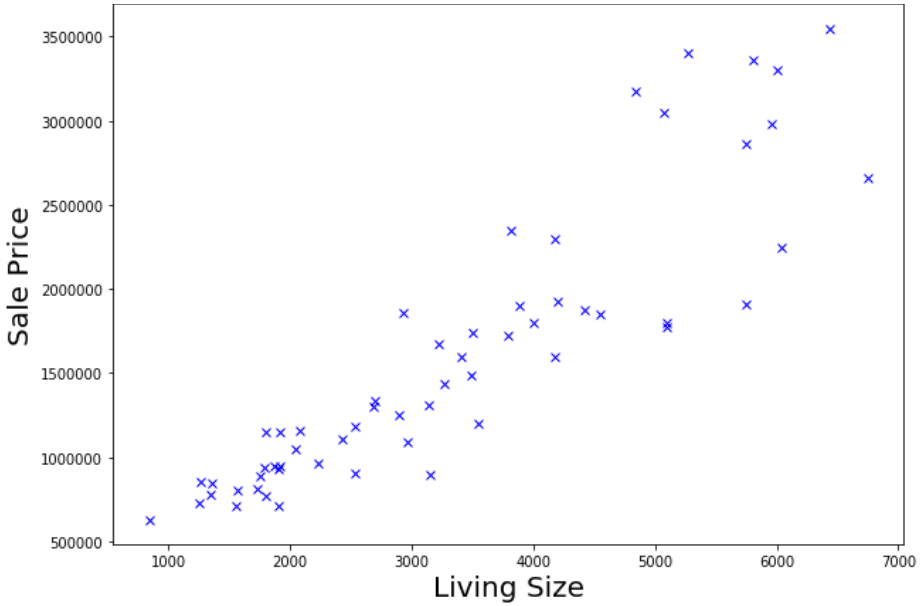
Find $\vec{\theta}$ to minimize $RSS(\vec{\boldsymbol{\theta}}) = \sum_{i=1}^{n} \left(h(\vec{x}^{(i)}) - y^{(i)}\right)^2 = \left\|\boldsymbol{X}\vec{\theta} - \vec{y}\right\|^2$

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

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

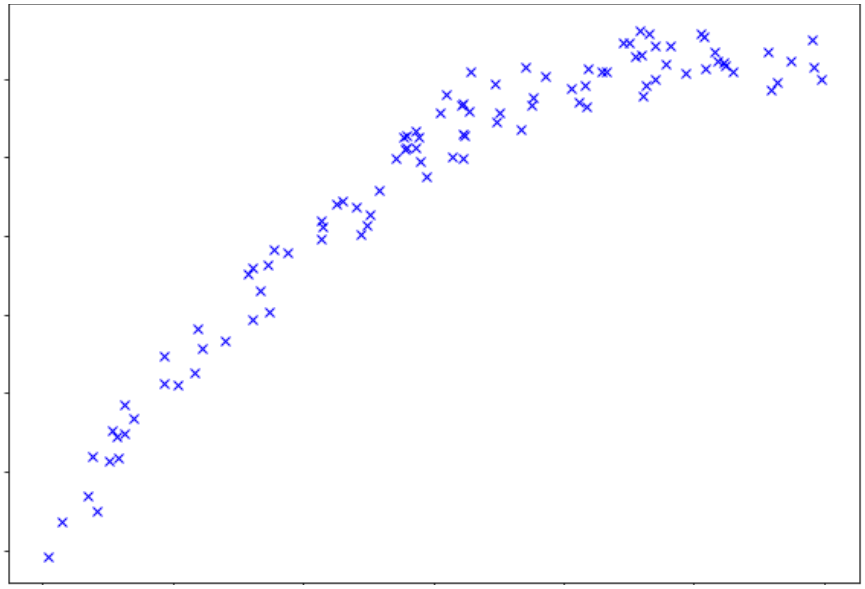
Least Squares solution $\vec{\theta} = (X^T X)^{-1} X^T \vec{y}$

- 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.



For example, New assumption:

$$h(x_1) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 \sqrt{x_1} + \theta_4 \log x_1$$

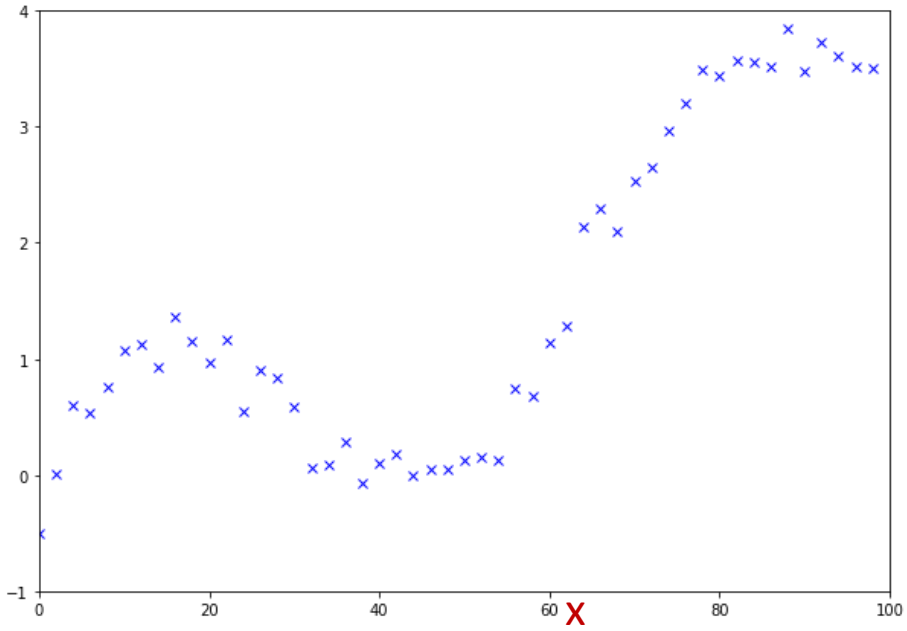
Suppose $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, we can try polynomial model using new features

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

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**

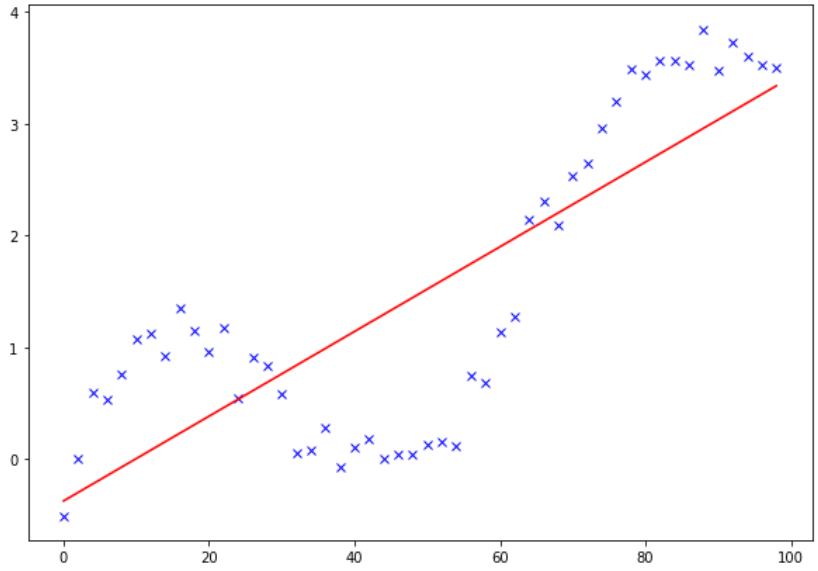


- Recall Linear Regression: Find

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

to minimize the RSS cost function:

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



- Minimize new weighted cost function

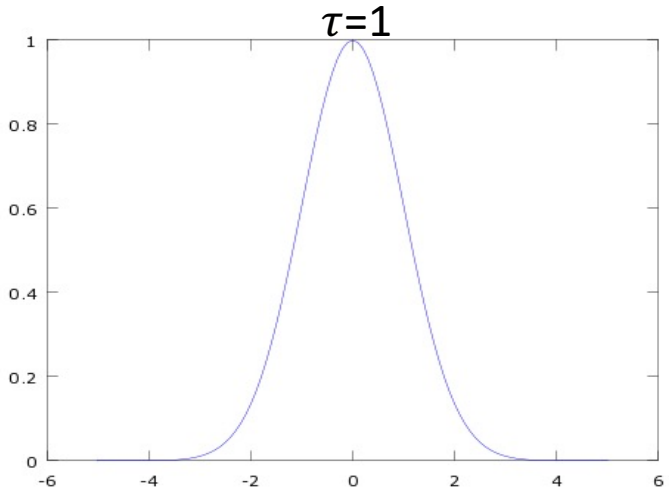
$$RSS_W(\vec{\theta}) = \sum_{i=1}^{n} w^{(i)} \left(h(\vec{x}^{(i)}) - \vec{y}^{(i)}\right)^2$$

Here, $w^{(i)} = \exp\left(-\frac{\|\vec{x}^{(i)} - \vec{x}\|^2}{2\tau^2}\right)$

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

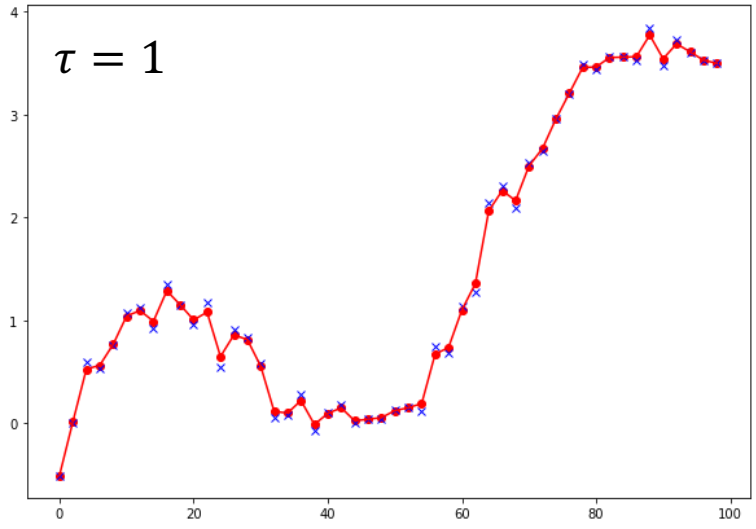
$$= \|X\vec{\theta} - \vec{y}\|_W^2$$

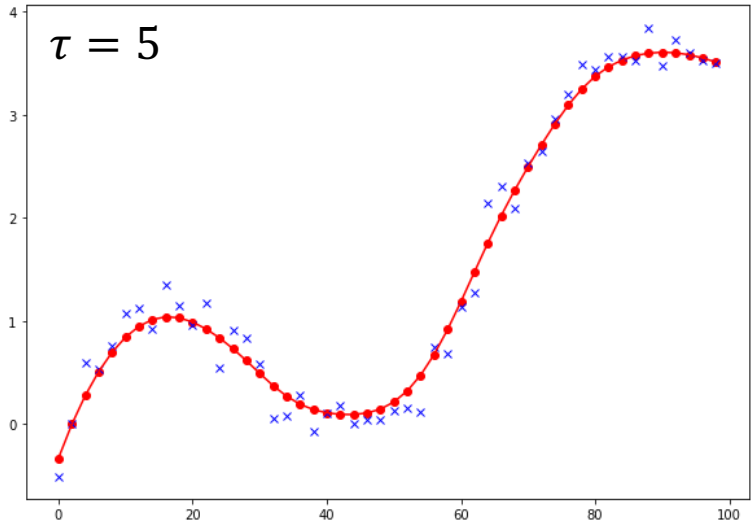
Claim: $\nabla_{\vec{\theta}} J = 2X^T W(X\vec{\theta} - \vec{y})$

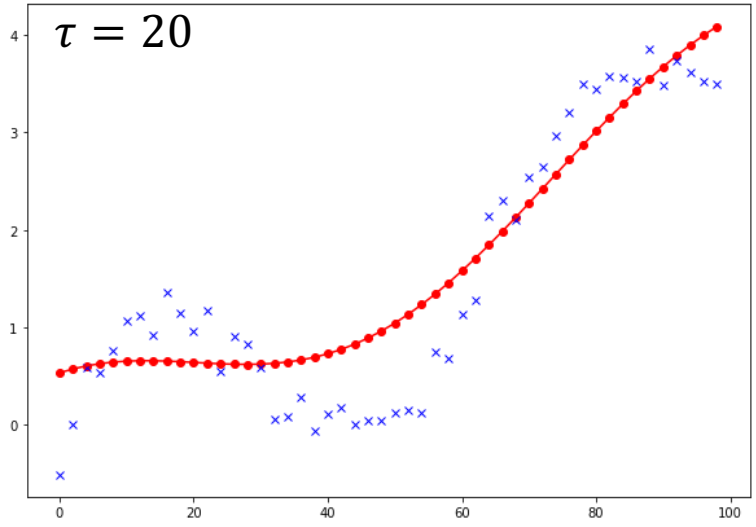


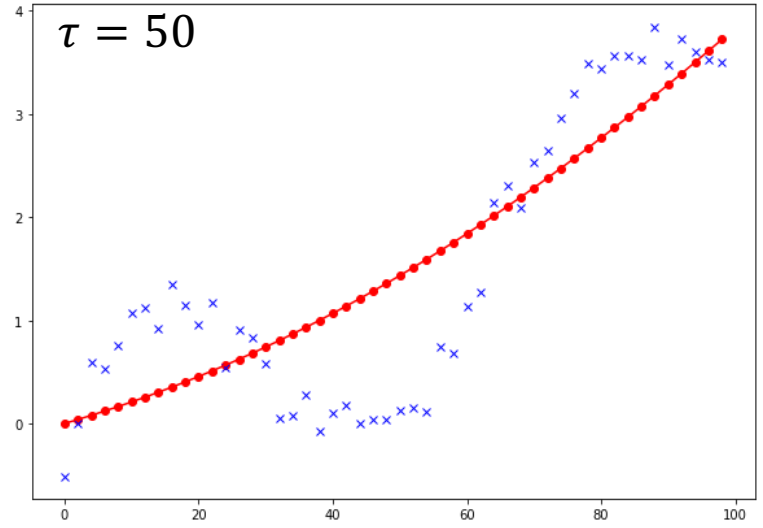
$$f(z) = \exp\left(-\frac{z}{2\tau^2}\right)$$

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









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)
```

```
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()
```

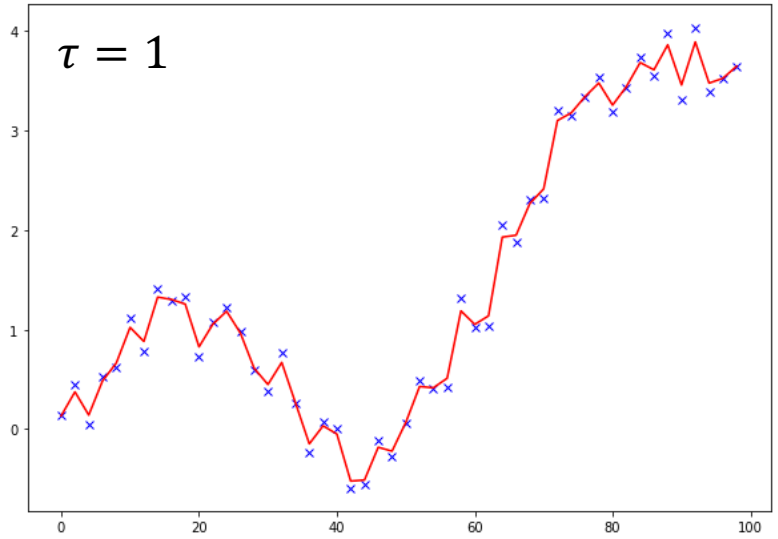
```
# Calculate predictions
pred = []
for k, xc_j in enumerate(xc):
    w = weight_w(xc, xc_j,5)
    theta = normal_equation(xc, yc, w)
    pred.append(theta.T.dot(xc_j[:,np.newaxis]).ravel()[0])
```

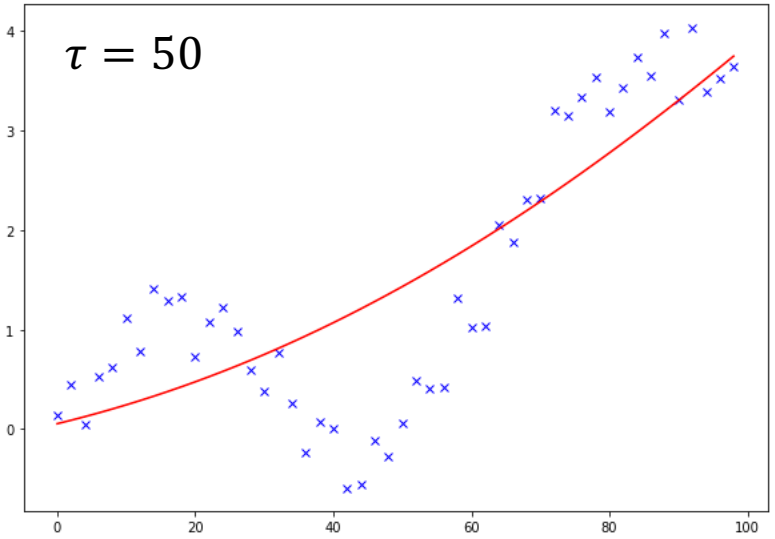
```
fig, ax = plt.subplots()
ax.plot(xc[:,1],yc,'x',color='Blue')
plt.plot(xc[:,1], pred, color='red')
plt.scatter(xc[:,1], pred, color ='red')
fig.set_size_inches(10, 7)
plt.show()
```

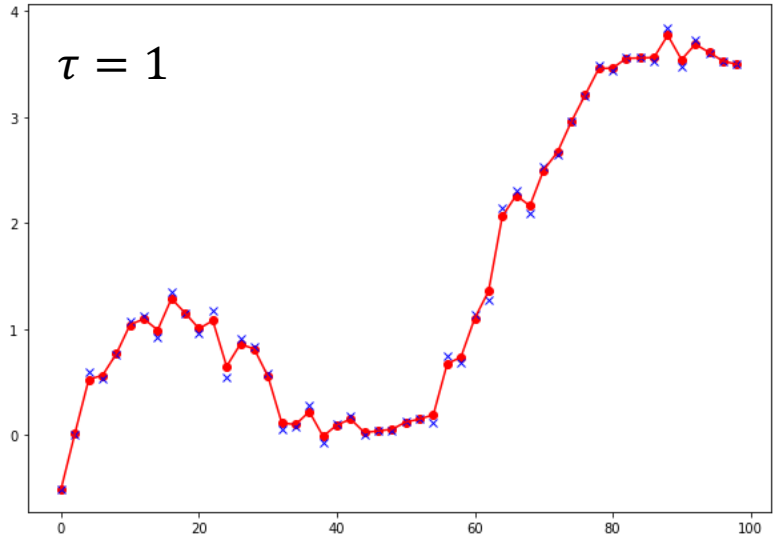
Given training Data $D = \left\{\left(\vec{x}^{(i)}, y^{(i)}\right) \mid i = 1, \ldots n\right\}$

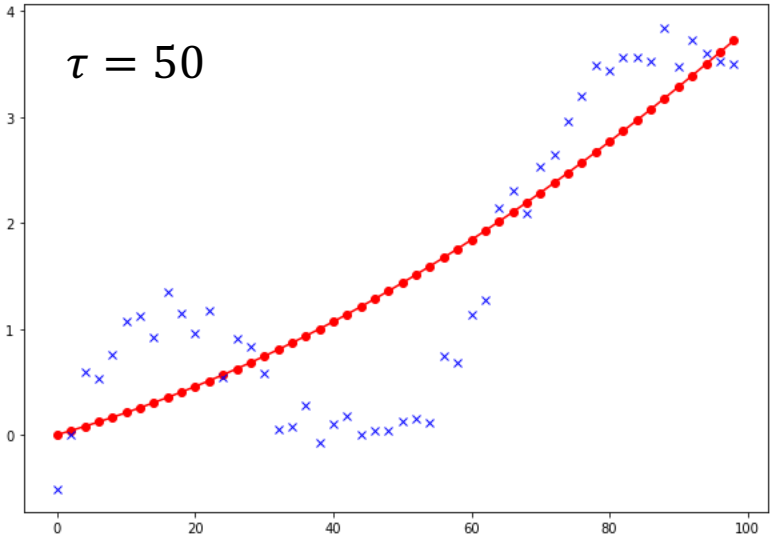
A **parametric** inference method gives a predictor function $h_{\vec{\theta}}(\vec{x})$ with a finite number of parameters $\theta_0, \theta_1, \ldots, \theta_d$.

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


$\tau = 1$


$\tau = 50$


$\tau = 1$


$\tau = 50$

- **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.

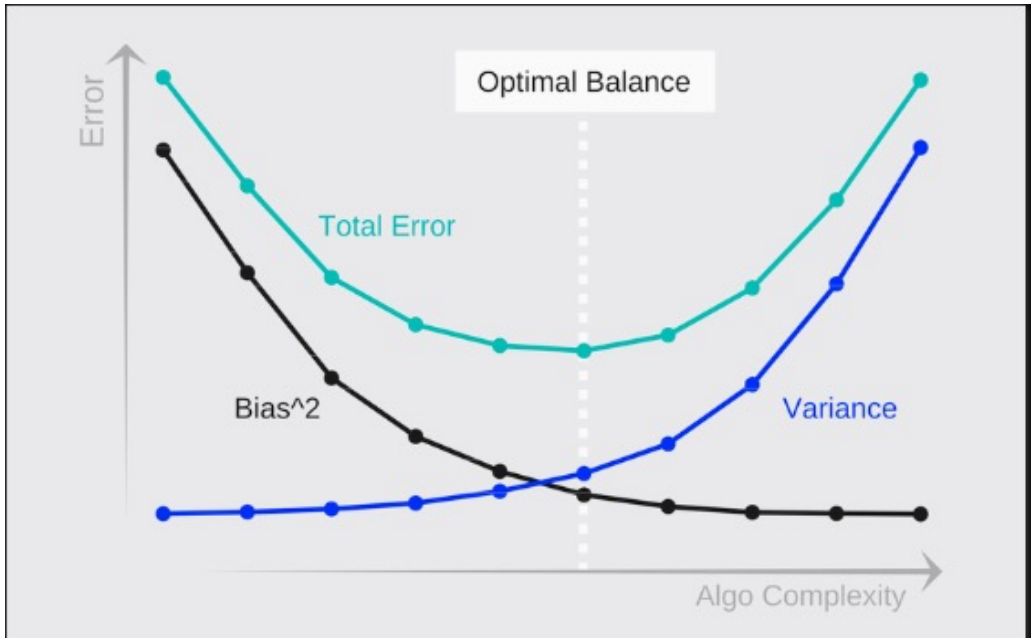
This contrast between the two algorithms is know as the **bias-variance trade off.**

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.)

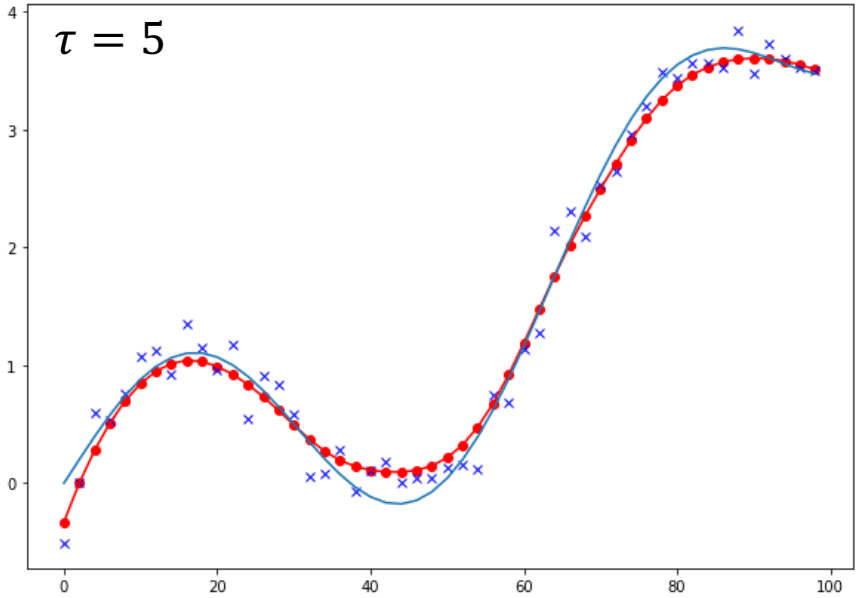
The **variance** is roughly the sensitivity of the model to the training data. (This part is due to the model's excessive sensitivity to small variations in the training data.)

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

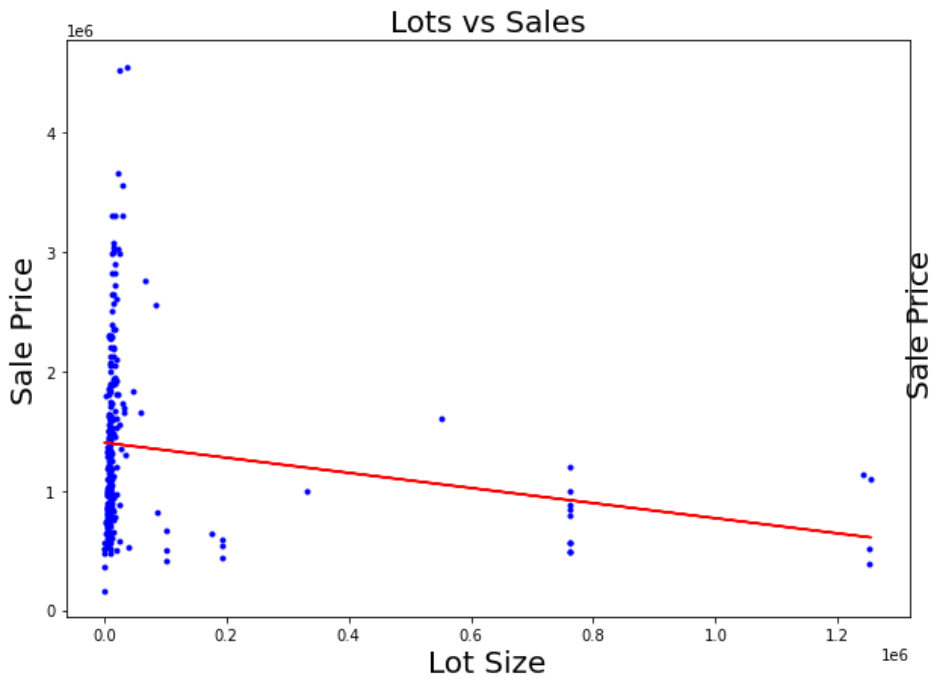
Irreducible Error is due to the noisiness of the data itself. The only way to reduce this part of the error is to clean up the data.



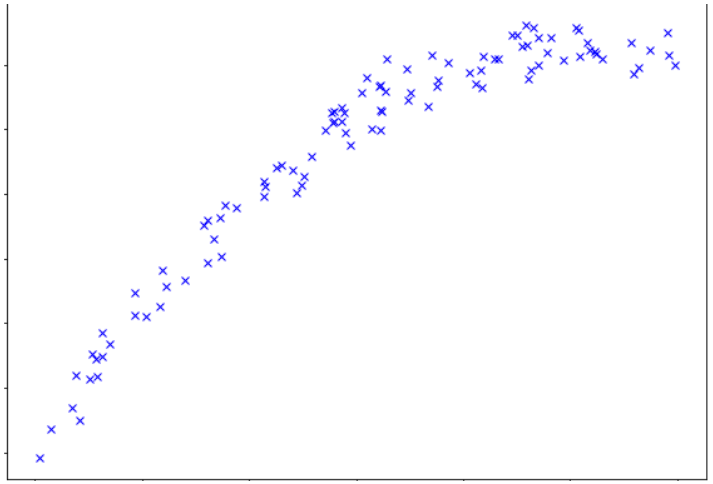
The best fit lies somewhere in between the extreme ends.



Remark: Clean up the data.



Interpretation in Probability



Data: $D = \{(\vec{x}^{(1)}, y^{(1)}), \ldots, (\vec{x}^{(n)}, y^{(n)})\}$

Goal: maximize the probability:

$$P(y^{(i)} \mid \vec{x}^{(i)}; \theta)$$

Suppose the data follows (linear or non-linear) model with unmodeled error ϵ.

$$y = h(\vec{x}) + \epsilon$$

This unmodeled error ϵ is also called **irreducible error**.

It is a random error term, which is independent from $\vec{x}$ and has mean $E(\epsilon)$=0.

- Even if it was possible to form a perfect estimate for $h(\vec{x})$, so that our estimated response took the form, our prediction would still have some error in it!

- The reason is that ϵ does not come from the model. we cannot control or reduce the error introduced by ϵ. The quantity ϵ may also contain unmeasurable variation.

- Our focus is on techniques for estimating $h(\vec{x})$ with the aim of minimizing the reducible error comes from the model.

- Keep in mind that the irreducible error will always provide an **upper bound** on the accuracy of our prediction for $\mathrm{y} = h(\vec{x})$. This bound is almost always unknown in practice.

Linear Regression

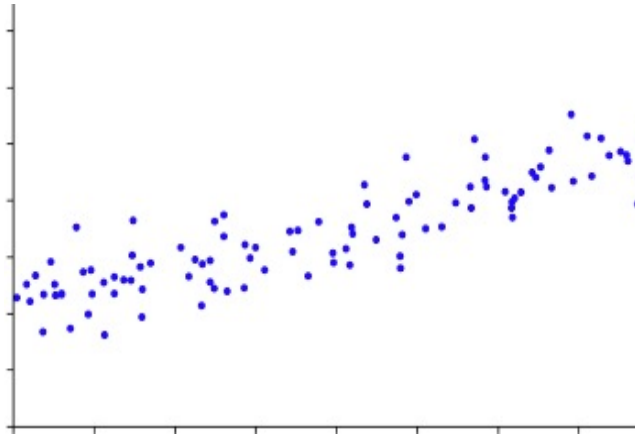
Suppose the data follows **linear** model $y = \theta^T \vec{x} + \epsilon$ with unmodeled error ϵ.

Suppose the unmodeled errors $\epsilon^{(i)}$ are independent identical distribution (IID).

$$y^{(i)} = \theta^T \vec{x}^{(i)} + \epsilon^{(i)}$$

A more restrictive (but common) **assumption**:
Suppose the unmodeled errors follow normal distribution,

$$\epsilon \sim \text{Normal}(0, \sigma^2)$$



That is $$P(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{1}{2}\left(\frac{\epsilon^{(i)}}{\sigma}\right)^2\right)$$

So, $$y^{(i)}|\vec{x}^{(i)} \sim \text{Normal}(\vec{\theta}^T \vec{x}^{(i)}, \sigma^2)$$

That is $$P(y^{(i)} \mid \vec{x}^{(i)}; \theta) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{1}{2}\left(\frac{y^{(i)} - \theta^T \vec{x}^{(i)}}{\sigma}\right)^2\right)$$

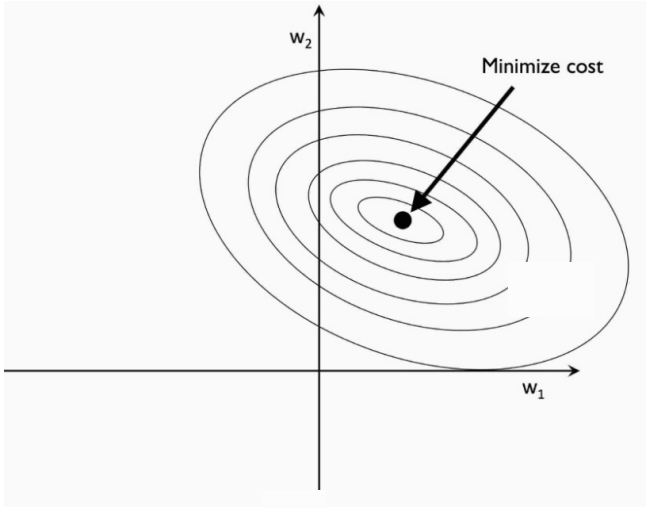
- Interpretation in Probability

➢ Feature selection (Shrinkage/regularization methods)

Data: $D = \{(\vec{x}^{(1)}, y^{(1)}), \ldots, (\vec{x}^{(n)}, y^{(n)})\}$

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

Square sum: $\text{RSS}(\vec{\theta}) = \sum_{i=1}^{n} (h(\vec{x}^{(i)}) - y^{(i)})^2$

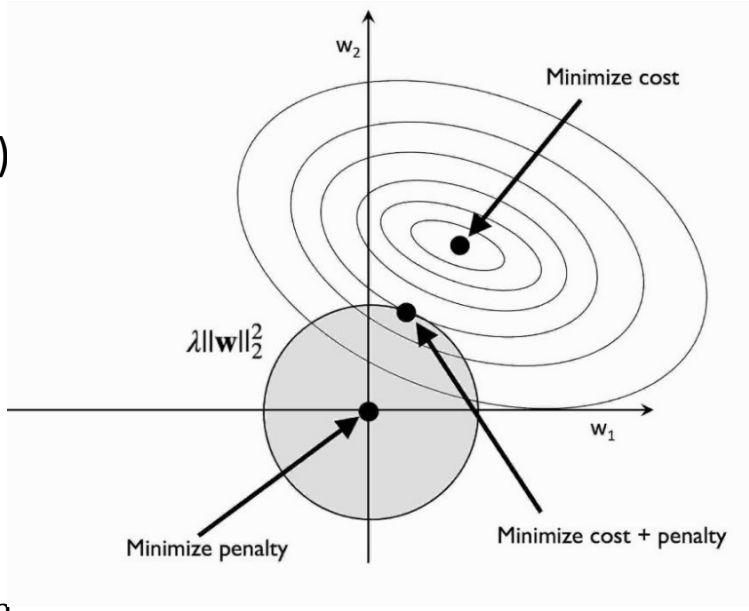


➢ Ridge Regression

Ridge regression cost function:

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

Lagrange Multiplier method (Karush–Kuhn–Tucker)



Minimize the Ridge cost function

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

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

- **Remarks on Ridge Regression**

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.

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

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

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

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

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

Remark: (Standardization of feature variables/ Feature Scaling)

Before applying the Ridge/Lesso/ 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)

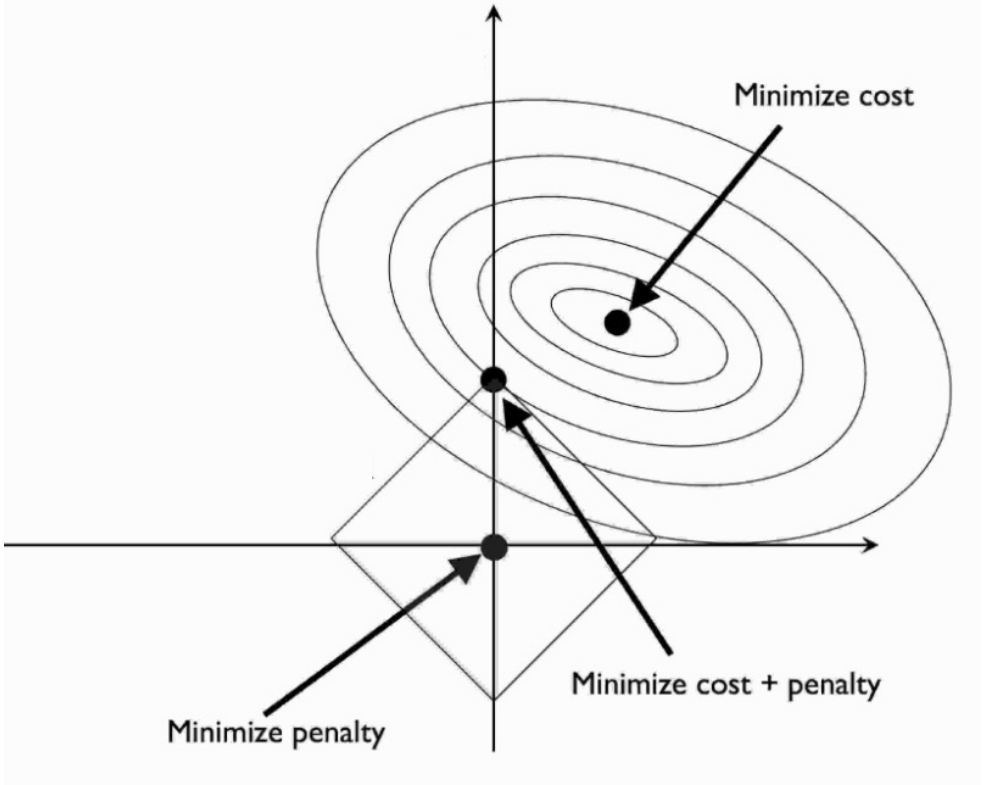
$$\frac{x_i - mean(x_i)}{s(x_i)}$$

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

Lasso Regression

Lasso regression cost function:

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



Minimize the Lasso cost function

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

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

- Remarks on Lasso and Ridge Regressions

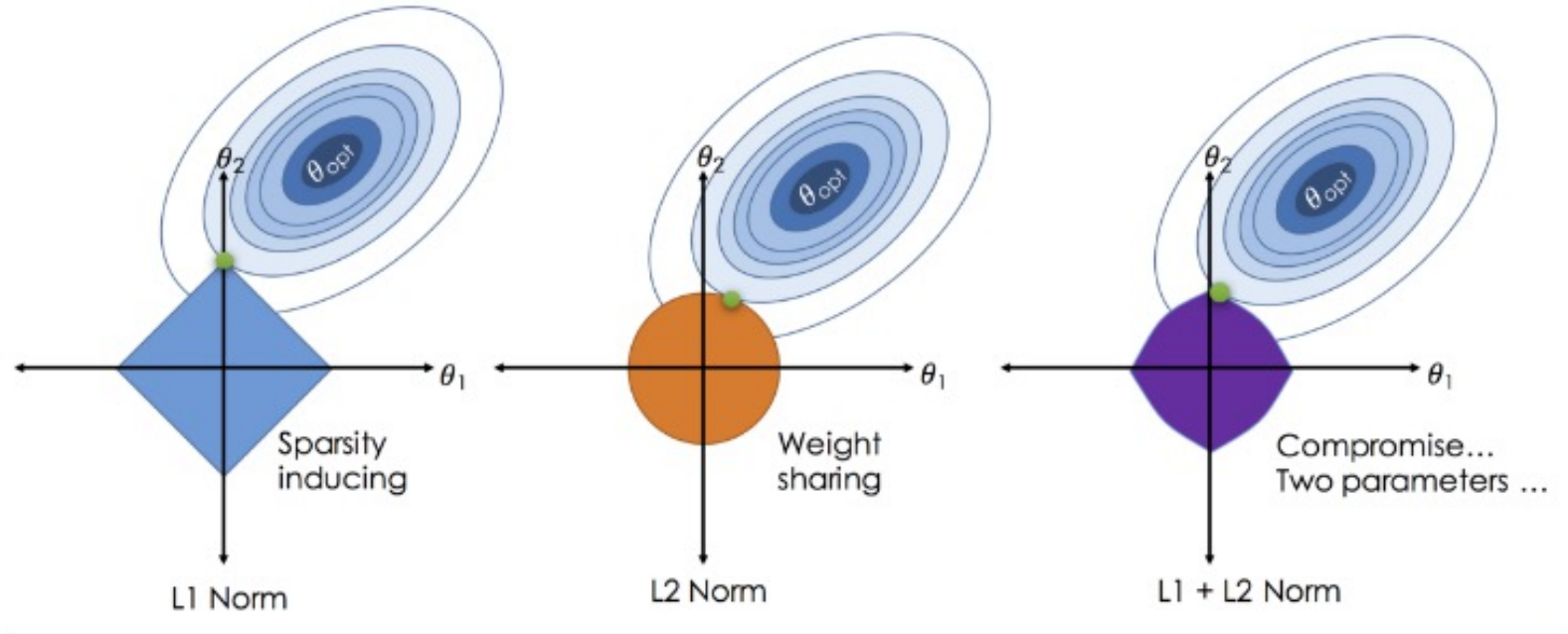
0. Lasso stands for Least Absolute Shrinkage and Selection Operator.

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.
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.

Elastic net Regression

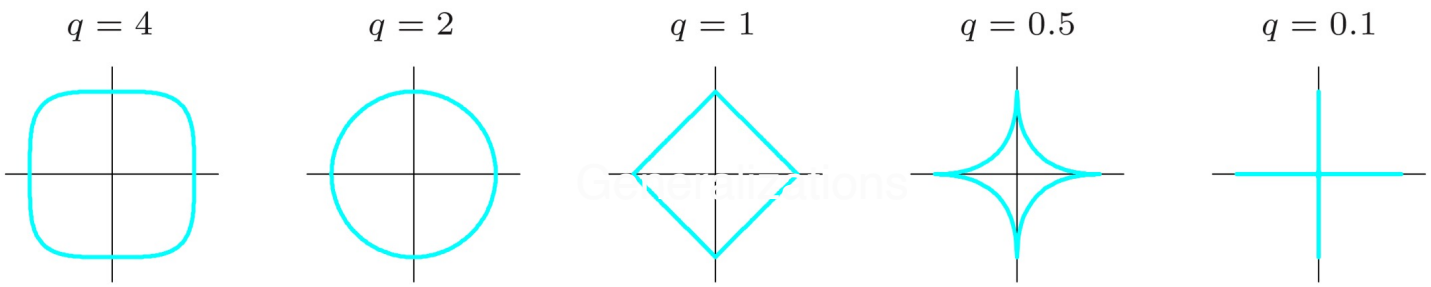
Elastic net regression cost function:

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



Generalizations: For any positive number q.

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



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?)