Artificial Intelligence and Machine Learning

Linear Regression

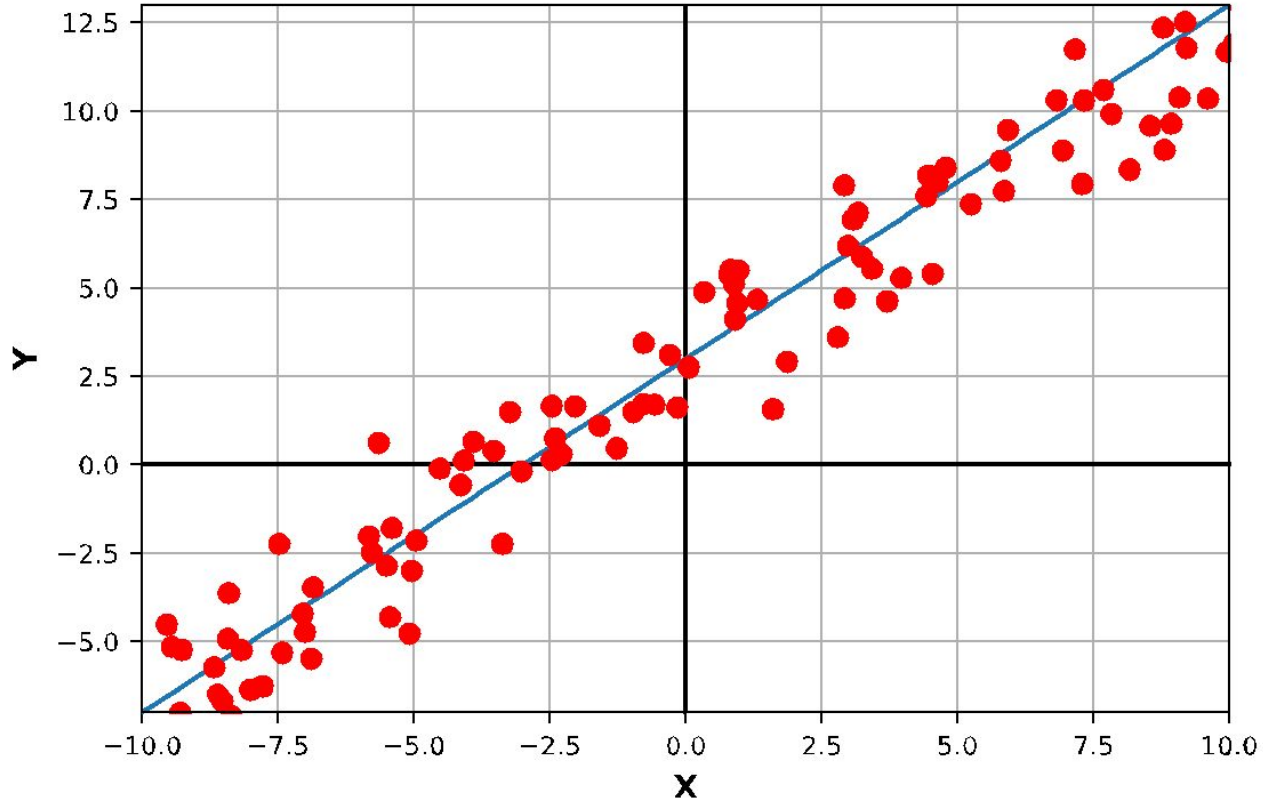
Lecture 1: Outline

- Linear Regression
- Optimization
- Applications

Motivation

- Linear Regression is “still” one of the most widely used ML/DL Algorithms
- Easy to understand and implement
- Efficient to Solve
- We will use Linear Regression to Understand the concepts of:
  - Data
  - Models
  - Loss
  - Optimization

Simple Linear Regression



Model (*Linear*)

Observation: Linear relation between x and y

$$Y = mX + b$$

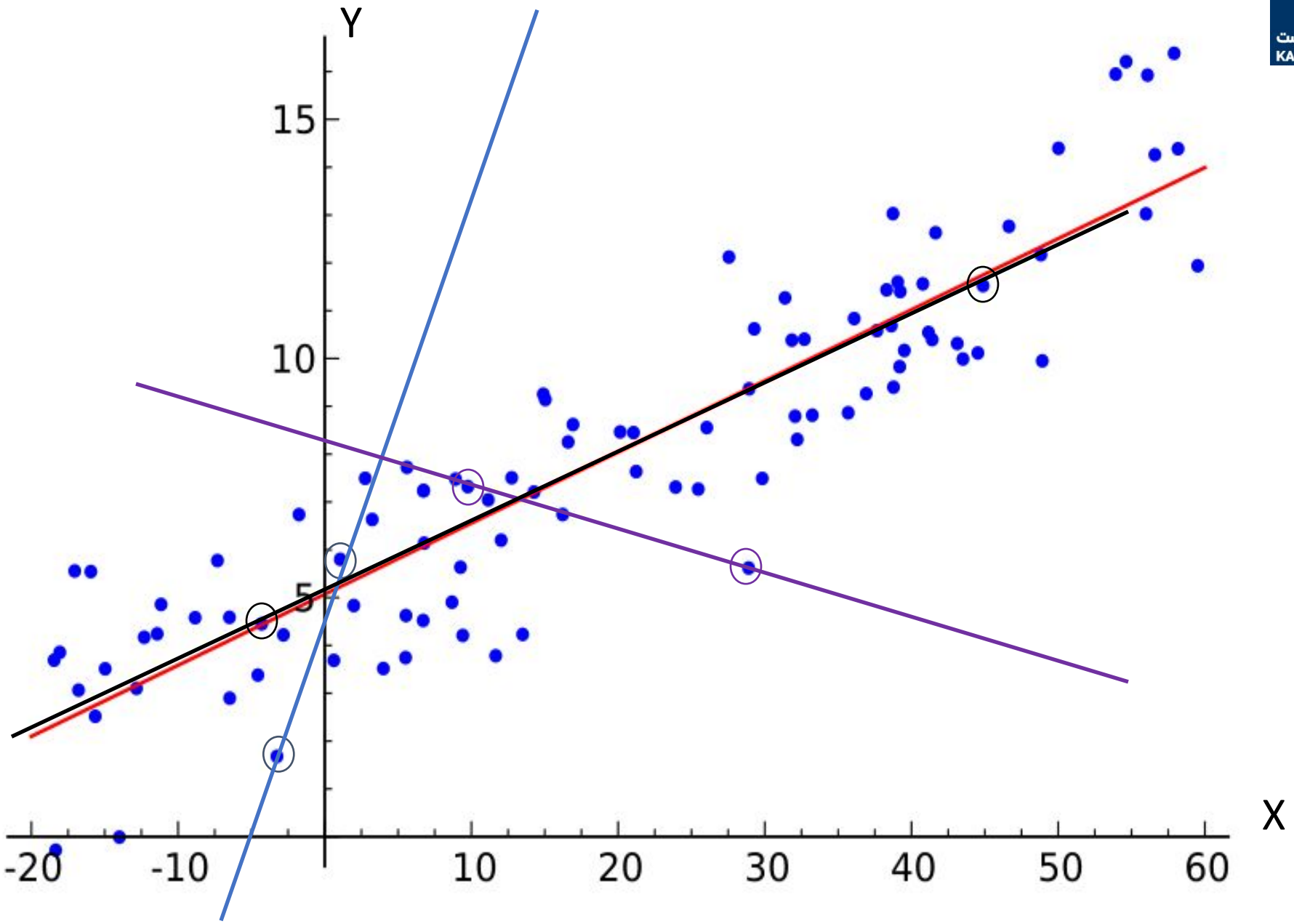
Objective: Find $\theta = \{m, b\}$

Y: Label, Response Variable
X: Features, Regressors
m: Slope
b: Bias

Simple Linear Regression

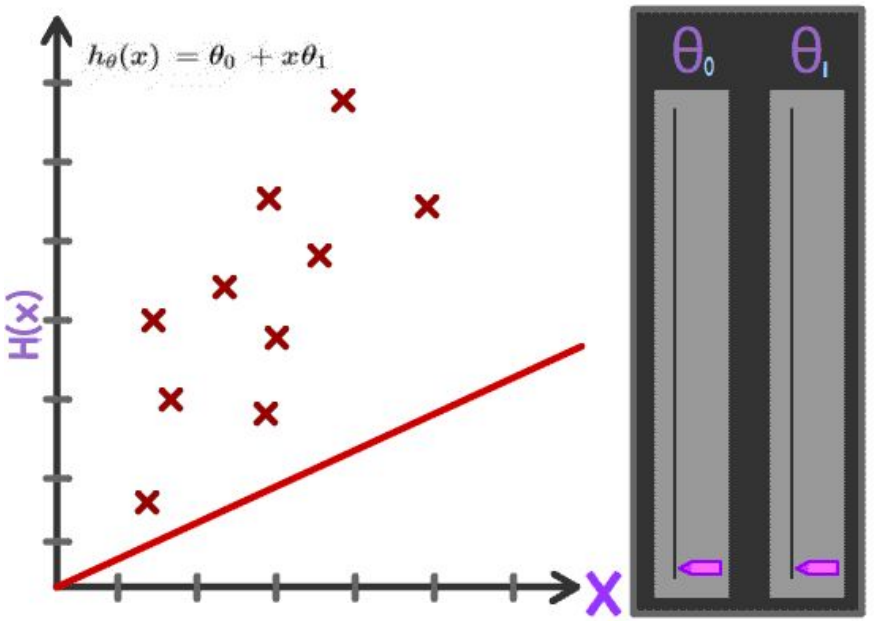
- Input: data (x_i, y_i), $i \in \{1,2, \dots, N\}$
- Goal: learn values of variable (m, b) $\quad Y = mX + b$

- Question: How many points in a plane do we need to fit a line through it?


Y
15
10
5
X
-20
-10
10
20
30
40
50
60

Solution Strategy for Solving the Problem

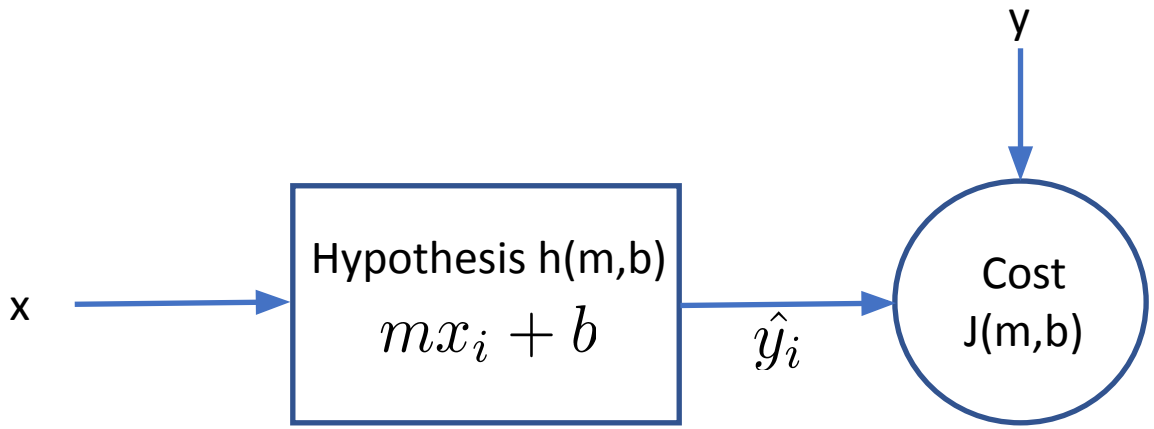
- We want a line which is in some sense the “average line” that represents the data.
- Any ideas as to how we can do it?



https://medium.com/analytics-vidhya/machine-learning-intuition-b4b49a671f65

Cost/Loss Function

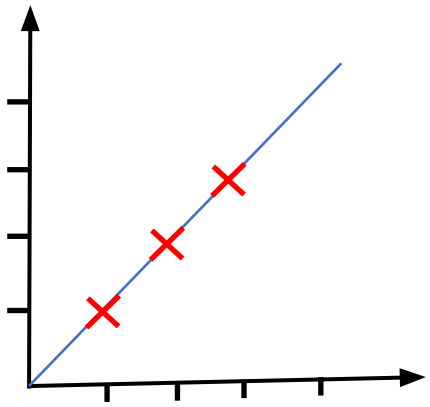
- We want to minimize the discrepancy between our model hypothesis (prediction) and the observed label (ground truth).



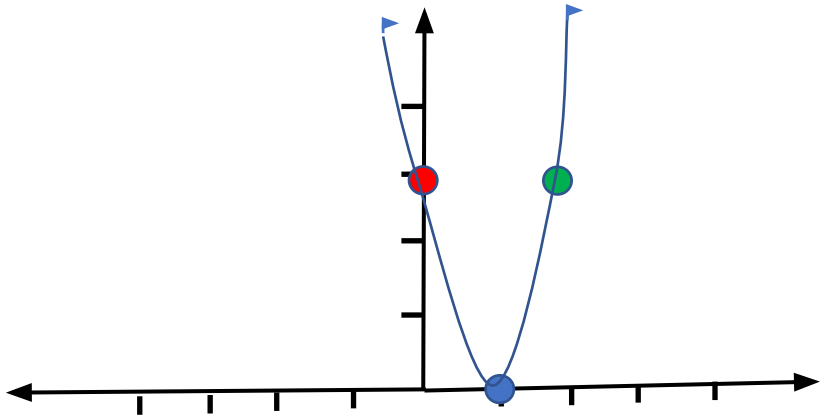
Mean Squared Error (MSE) Loss

$$J = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

Intuition of Cost Function



Hypothesis



Cost Function

$$h(x) = mx$$

$$J(m) = \sum_{i=1}^{3} (y_i - mx_i)^2$$

$$J(1) = 0$$

$$J(0) = 14$$

$$J(2) = 14$$

How to find minima of a function (Review):

$$J(m) = \sum_{i=1}^{3} (i - mi)^2$$

$$J(m) = \sum_{i=1}^{3} (y_i - mx_i)^2$$

$$\frac{dJ(m)}{dm} = \frac{d}{dm} \sum_{i=1}^{3} (i - mi)^2$$

$$\frac{dJ(m)}{dm} = \sum_{i=1}^{3} \frac{d}{dm} (i - mi)^2$$

$$\frac{dJ(m)}{dm} = \sum_{i=1}^{3} -2i(i - mi) \qquad -2 \sum_{i=1}^{3} i^2 + 2m \sum_{i=1}^{3} i^2 = 0 \qquad m = 1$$

Hypothesis Function with 2 Variables

- Let's setup regression for linear function in two variables:
- The hypothesis function is:

$$\hat{y}_i = mx_i + b$$

- Similar to the previous problem our loss function is:

$$J = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- Let's calculate the partial derivatives of the loss function w.r.t. m, b

Gradient of the cost function

- We get the following expressions for the gradient of the cost function

$$\frac{\partial J}{\partial m} = \frac{1}{N}\sum_{i=1}^{N} -2(y_i - \hat{y}_i)x_i$$

$$\frac{\partial J}{\partial b} = \frac{1}{N}\sum_{i=1}^{N} -2(y_i - \hat{y}_i)$$

Gradient of the cost function

- Simplifying the above expressions, we get:

$$\frac{\partial J}{\partial m} = \frac{-2}{N}\sum_{i=1}^{N} y_i x_i + \frac{2m}{N}\sum_{i=1}^{N} x_i^2 + \frac{2b}{N}\sum_{i=1}^{N} x_i$$

$$\frac{\partial J}{\partial b} = \frac{-2}{N}\sum_{i=1}^{N} y_i + \frac{2m}{N}\sum_{i=1}^{N} x_i + \frac{2b}{N}\sum_{i=1}^{N} 1$$

Gradient of the cost function

- Setting the Gradient equal to 0, and solving for m and b, we get

$$\underbrace{\begin{bmatrix} \frac{\sum_i x_i^2}{N} & \frac{\sum_i x_i}{N} \\ \frac{\sum_i x_i}{N} & 1 \end{bmatrix}}_{} \underbrace{\begin{bmatrix} m \\ b \end{bmatrix}}_{} = \underbrace{\begin{bmatrix} \frac{\sum_i x_i y_i}{N} \\ \frac{\sum_i y_i}{N} \end{bmatrix}}_{}$$

Practice Time!

Issues with the Approach

- Calculating gradients like this can quickly become tedious
- Each term on either side of the expression can be written a dot product of two vectors (maybe we can calculate it more efficiently)?

- Let’s explore if we can do something better through vectorization
- Before, we step into vectorization, let’s consider regression for nonlinear functions

Vectorization

- To truly appreciate the power of vectorization. Let's make the problem a little more complex. The hypothesis function is now

$$\hat{y}_i = w_0 + w_1 x_i^1 + w_2 x_i^2 + \cdots + w_M x_i^M$$

- Where w_j are the unknow weightns of the data x^j features of the input
- Next, we denote the discrepency between y_i and $\hat{y}_i$ as ϵ_i

$$y_i = \hat{y}_i + \epsilon_i$$

Vectorization

- Now let's collect the above equation for all N datapoints

$$y_1 = \hat{y}_1 + \epsilon_1$$
$$y_2 = \hat{y}_2 + \epsilon_2$$
$$\vdots$$
$$y_N = \hat{y}_N + \epsilon_N$$

Vectorization

- Replacing the values of $\hat{y}$, we get:

$$y_1 = w_0 + w_1 x_1^1 + w_2 x_1^2 + ... + w_M x_1^M + \epsilon_1$$
$$y_2 = w_0 + w_1 x_2^1 + w_2 x_2^2 + ... + w_M x_2^M + \epsilon_2$$
$$\vdots$$
$$y_N = w_0 + w_1 x_N^1 + w_2 x_N^2 + ... + w_M x_N^M + \epsilon_N$$

Vectorization

- Collecting the equations in matrix form:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_N \end{bmatrix} = \begin{bmatrix} 1 & x_1^1 & x_1^2 & \dots & x_1^M \\ 1 & x_2^1 & x_2^2 & \dots & x_2^M \\ 1 & x_3^1 & x_3^2 & \dots & x_3^M \\ & . & & & \\ & . & & & \\ & . & & & \\ 1 & x_N^1 & x_N^2 & \dots & x_N^M \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ . \\ . \\ . \\ w_M \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ . \\ . \\ . \\ \epsilon_N \end{bmatrix}$$

Vectorization

- Notice the rows of the matrix on the right are data samples:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_N \end{bmatrix} = \begin{bmatrix} \dots & \mathbf{x_1} & \dots \\ \dots & \mathbf{x_2} & \dots \\ \dots & \mathbf{x_3} & \dots \\ & . & \\ & . & \\ & . & \\ \dots & \mathbf{x_N} & \dots \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ . \\ . \\ . \\ w_M \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ . \\ . \\ . \\ \epsilon_N \end{bmatrix}$$

Vectorization

$$\mathcal{D} = \{(\mathbf{x_i}, \mathbf{y_i})\}_{\mathbf{i=1}}^{\mathbf{N}}$$

- Let's formalize some notations:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_N \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} \dots & \mathbf{x_1} & \dots \\ \dots & \mathbf{x_2} & \dots \\ \dots & \mathbf{x_3} & \dots \\ & . & \\ & . & \\ & . & \\ \dots & \mathbf{x_N} & \dots \end{bmatrix} \quad \boldsymbol{\theta} = \begin{bmatrix} w_0 \\ w_1 \\ . \\ . \\ . \\ w_M \end{bmatrix} \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ . \\ . \\ . \\ \epsilon_N \end{bmatrix}$$

$$\boldsymbol{y} = \boldsymbol{X\theta} + \boldsymbol{\epsilon}$$

Cost function for the Vectorized form

- Notice that we are using the MSE cost function:

$$J = \frac{1}{N}\sum_{i}(y_i - \hat{y}_i)^2$$

- Using the definition of epsilon we can write the above as:

$$J = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(\epsilon_i)^2$$

- Using the definition of dot product the above can be written as:

$$J = \frac{1}{N}\sum_{i=1}^{N}(\epsilon_i)^2 = \boldsymbol{\epsilon}^T\boldsymbol{\epsilon}$$

Optimization

- The optimization problem is now:

$$\min_{\boldsymbol{\theta}} \boldsymbol{\epsilon}^T \boldsymbol{\epsilon}$$

$$\min_{\boldsymbol{\theta}} \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} = \min_{\boldsymbol{\theta}} (\boldsymbol{y} - (\boldsymbol{X}\boldsymbol{\theta}))^T (\boldsymbol{y} - (\boldsymbol{X}\boldsymbol{\theta}))$$

- We will use chain rule to calculate the gradient of the cost function:

$$\frac{\partial}{\partial \boldsymbol{\theta}} J = \frac{dJ}{d\boldsymbol{\epsilon}} \nabla_{\boldsymbol{\theta}} \boldsymbol{\epsilon}$$

Linear Least Squares

- We get:

$$\frac{\partial}{\partial \boldsymbol{\theta}} J = \boldsymbol{X}^T 2(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})$$

- Setting it equal to zero we can solve for $\boldsymbol{\theta}$:

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

Probabilistic Interpretation of Linear Regression and MLE

- We can also look at the probabilistic Interpretation of Linear Regression.
- Keeping everything else same as the previous formulation

$$y_i = \boldsymbol{x}_i^T \boldsymbol{\theta} + \epsilon_i$$

- Now assume that $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, then $y_i \sim \mathcal{N}(\boldsymbol{x}_i^T \boldsymbol{\theta}, \sigma^2)$
- We can write the conditional distribution as :

$$\mathbb{P}(y_i | \boldsymbol{x}_i) \sim \mathcal{N}(0, \sigma^2)$$

Probabilistic Interpretation of LR

- Let's assume that all data point in the dataset are i.i.d. (independent identically distributed). Then we have:

$$\mathbb{P}(\mathcal{D}) = \prod_{i=1}^{N} \mathbb{P}(\boldsymbol{x}_i, y_i)$$

- Using Bayes Theorem we can write:

$$\prod_{i=1}^{N} \mathbb{P}(\boldsymbol{x}_i, y_i) = \prod_{i=1}^{N} \mathbb{P}(\boldsymbol{x}_i)\mathbb{P}(y_i|\boldsymbol{x}_i)$$

Maximum Likelihood Estimator

- In simple words, given the Dataset we want to find the values of the unknown parameters which maximize the probability of the Dataset.
- Using the definition of the conditional distribution we have

$$\mathbb{P}(y_i|\boldsymbol{x}_i) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-(y_i - \boldsymbol{x}_i^T\boldsymbol{\theta})\right)$$

- Using the definition we get

$$\prod_{i=1}^{N} \mathbb{P}(\boldsymbol{x}_i, y_i) = \prod_{i=1}^{N} \mathbb{P}(\boldsymbol{x}_i) \prod_{i=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-(y_i - \boldsymbol{x}_i^T\boldsymbol{\theta})\right)$$

Maximum Likelihood Estimator

- Let's try to maximize:

$$\prod_{i=1}^{N} \mathbb{P}(\boldsymbol{x}_i, y_i) = \prod_{i=1}^{N} \mathbb{P}(\boldsymbol{x}_i) \prod_{i=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-(y_i - \boldsymbol{x}_i^T\boldsymbol{\theta})\right)$$

- Note that

$$\arg\max_{\theta} \prod_{i=1}^{N} \mathbb{P}(\boldsymbol{x}_i, y_i) = \arg\max_{\theta} \prod_{i=1}^{N} \exp\left(-(y_i - \boldsymbol{x}_i^T\boldsymbol{\theta})^2\right)$$

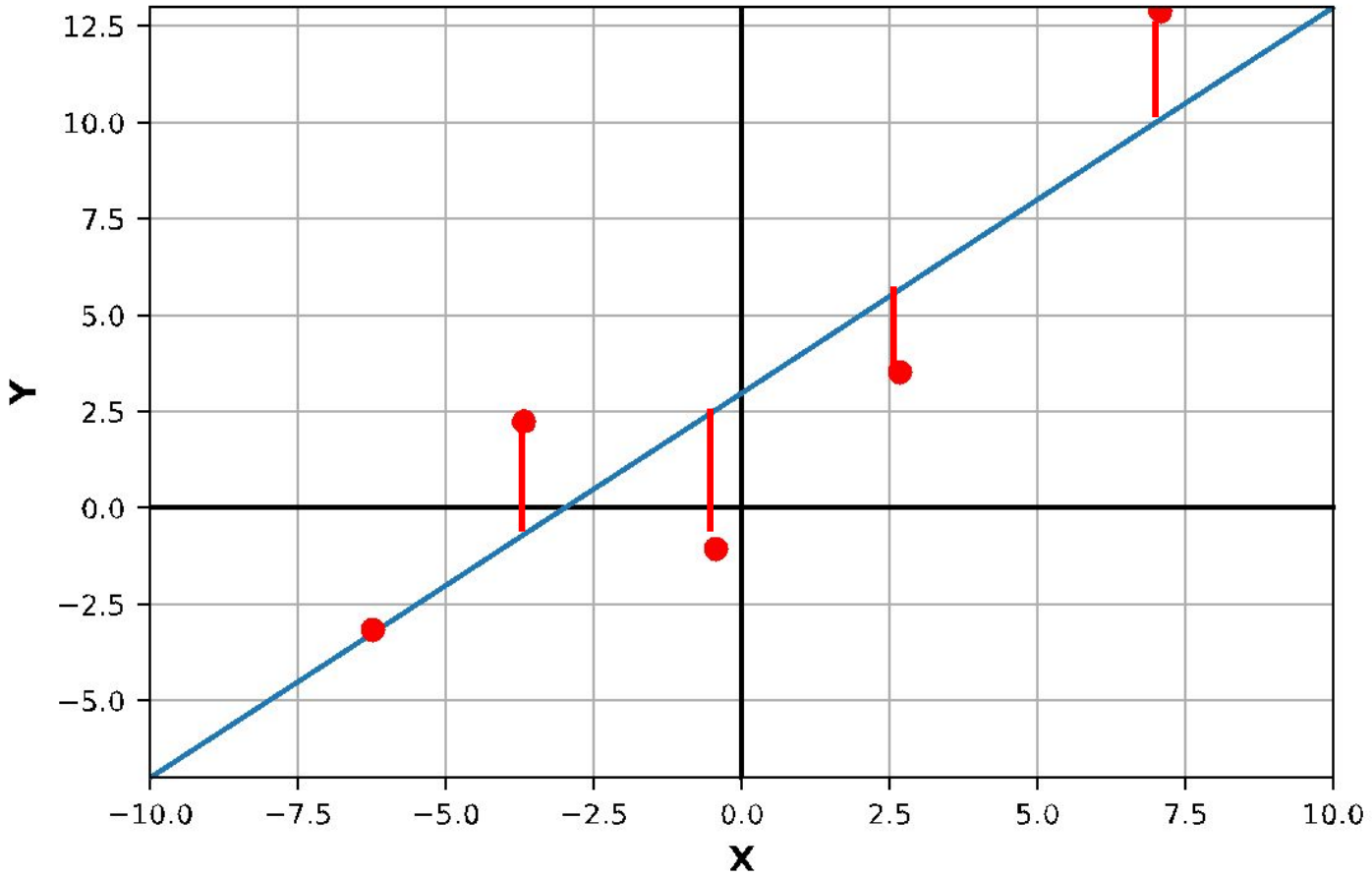
Maximum Likelihood Estimator

- Furthermore, since the right hand side of the above equation is monotonic in \theta the arg max will not change if we take log of the expression

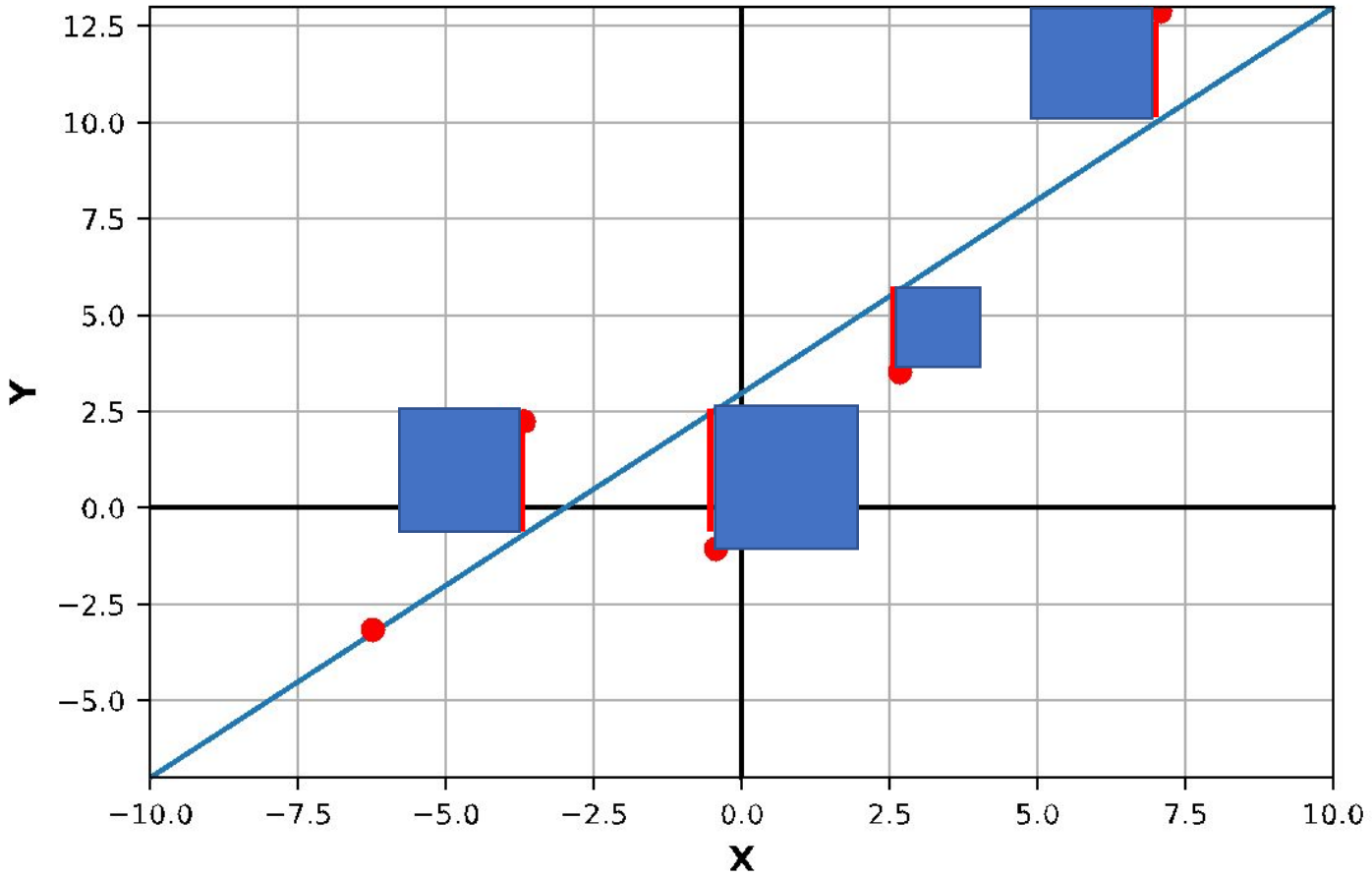
$$\arg\max_{\theta} \prod_{i=1}^{N} \exp\left(-(y_i - \boldsymbol{x}_i^T \boldsymbol{\theta})^2\right) = \arg\max_{\theta} \sum_{i=1}^{N} \left(-(y_i - \boldsymbol{x}_i^T \boldsymbol{\theta})^2\right)$$

- Notice that the right hand side is minising the MSE.
- **Hence solution of minimizing the MSE is equivalent to Maximum Likelihood Estimator for linear regression.**

Error Visualization

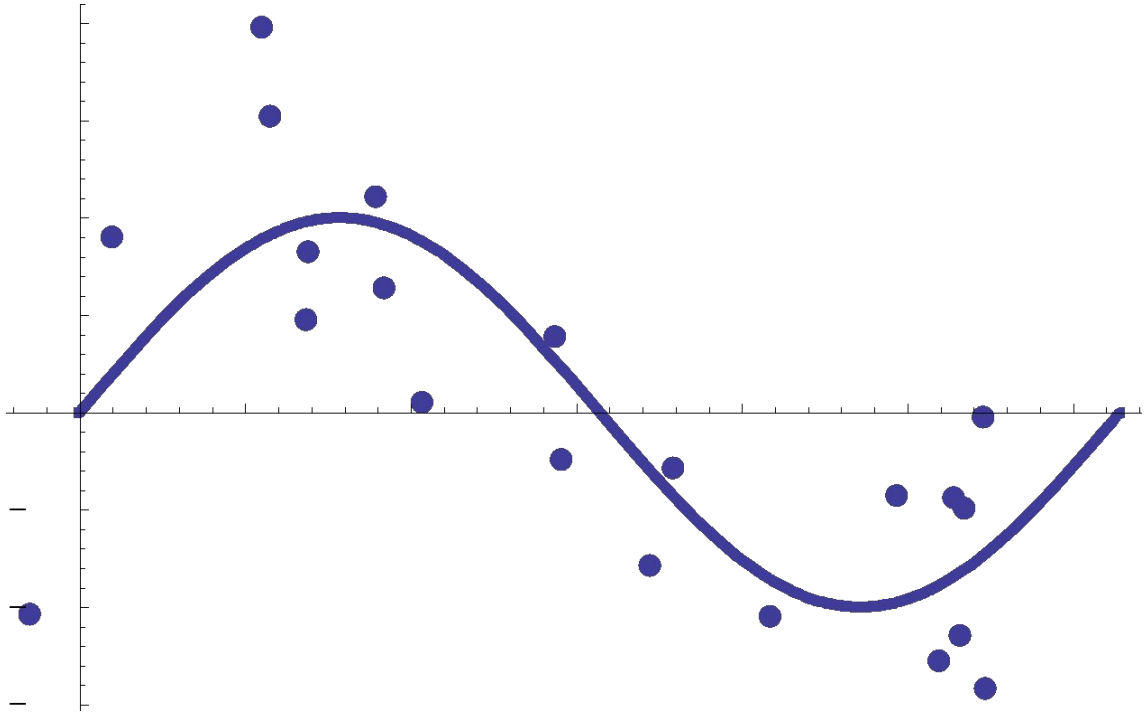


Error Visualization



Fitting Non-linear Data

- What if Y has a non-linear response?



- Can we still use a linear model?

Notation

- Some clarification about the notation we will use for this course

$$X_i^{j,[k]}$$

- i is the index of the data, j is the feature number, and k is the power.

Transforming the Feature Space

- We can transform features x_i

$$x_i = (x_i^1, x_i^2, x_i^3, ..., x_i^m)$$

- We will apply some non-linear transformation ϕ:

$$\phi : \mathbb{R}^m \rightarrow \mathbb{R}^M$$

- For example, Polynomial transformation:

$$\phi(x_i) = \{1, x_i^1, x_i^{1,[2]}, ..., x_i^{1,[k]}, x_i^2, x_i^{2,[2]}, ..., x_i^{2,[k]}, ..., x_i^m, x_i^{m,[2]}, ..., x_i^{m,[k]}\}$$

  - others: cosine, splines, radial basis functions, etc.
  - Expert engineered features (modeling)

Transforming the Feature Space

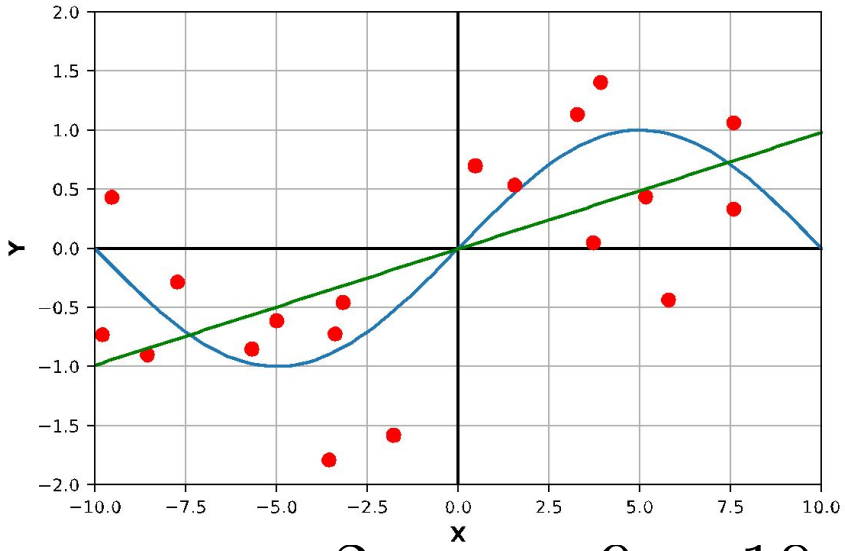
- Transform features x_i

$$x_i = (X_{i,1}, X_{i,2}, \ldots, X_{i,p})$$

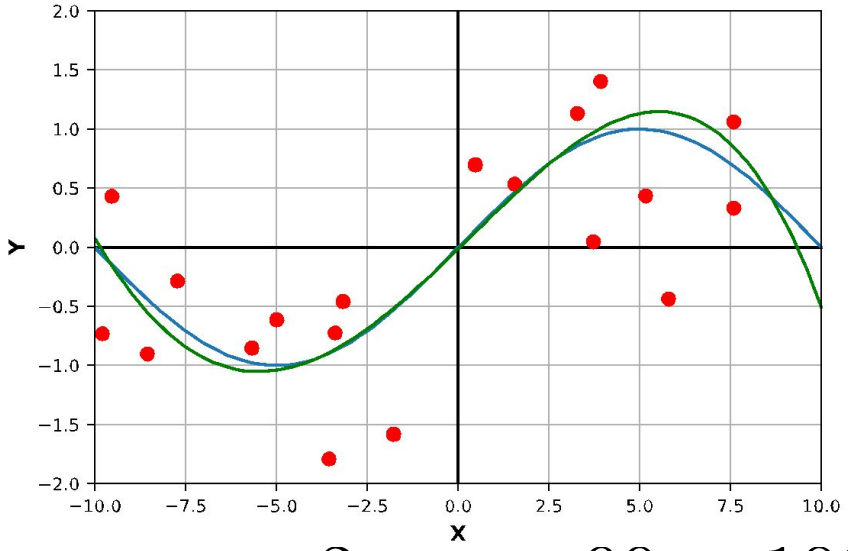
- By applying non-linear transformation ϕ:

- Example:
$$\phi : \mathbb{R}^p \rightarrow \mathbb{R}^k$$
$$\phi(x) = \{1, x, x^2, \ldots, x^k\}$$
  - others: splines, radial basis functions, …
  - Expert engineered features (modeling)

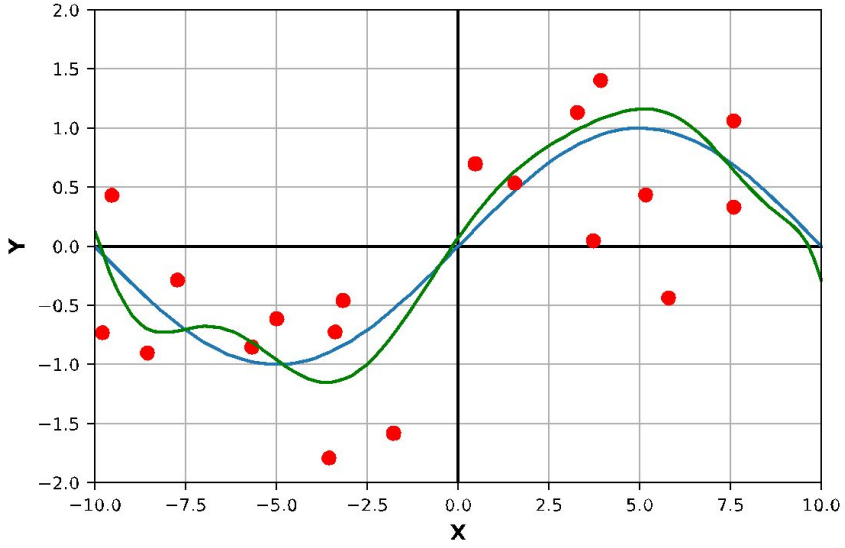
$\{1, x\}$



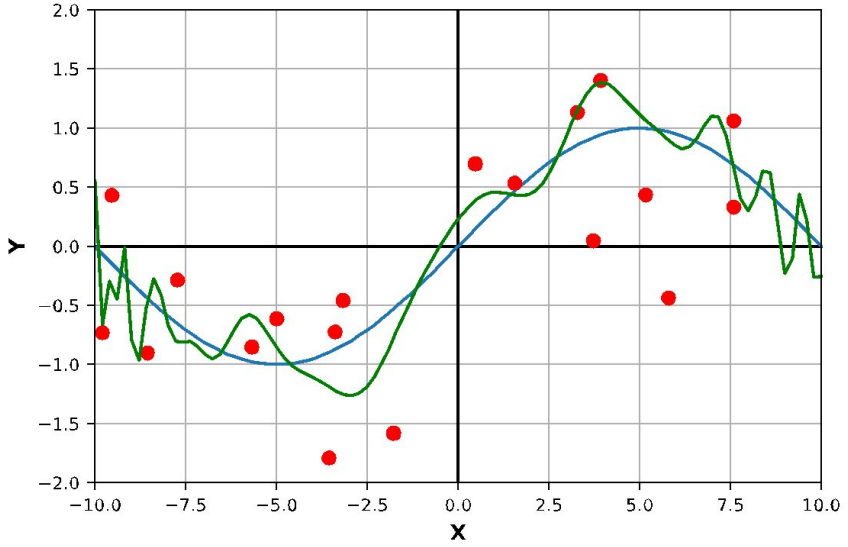
$\{1, x, x^2, x^3, x^4\}$



$\{1, x, x^2, ..., x^9, x^{10}\}$

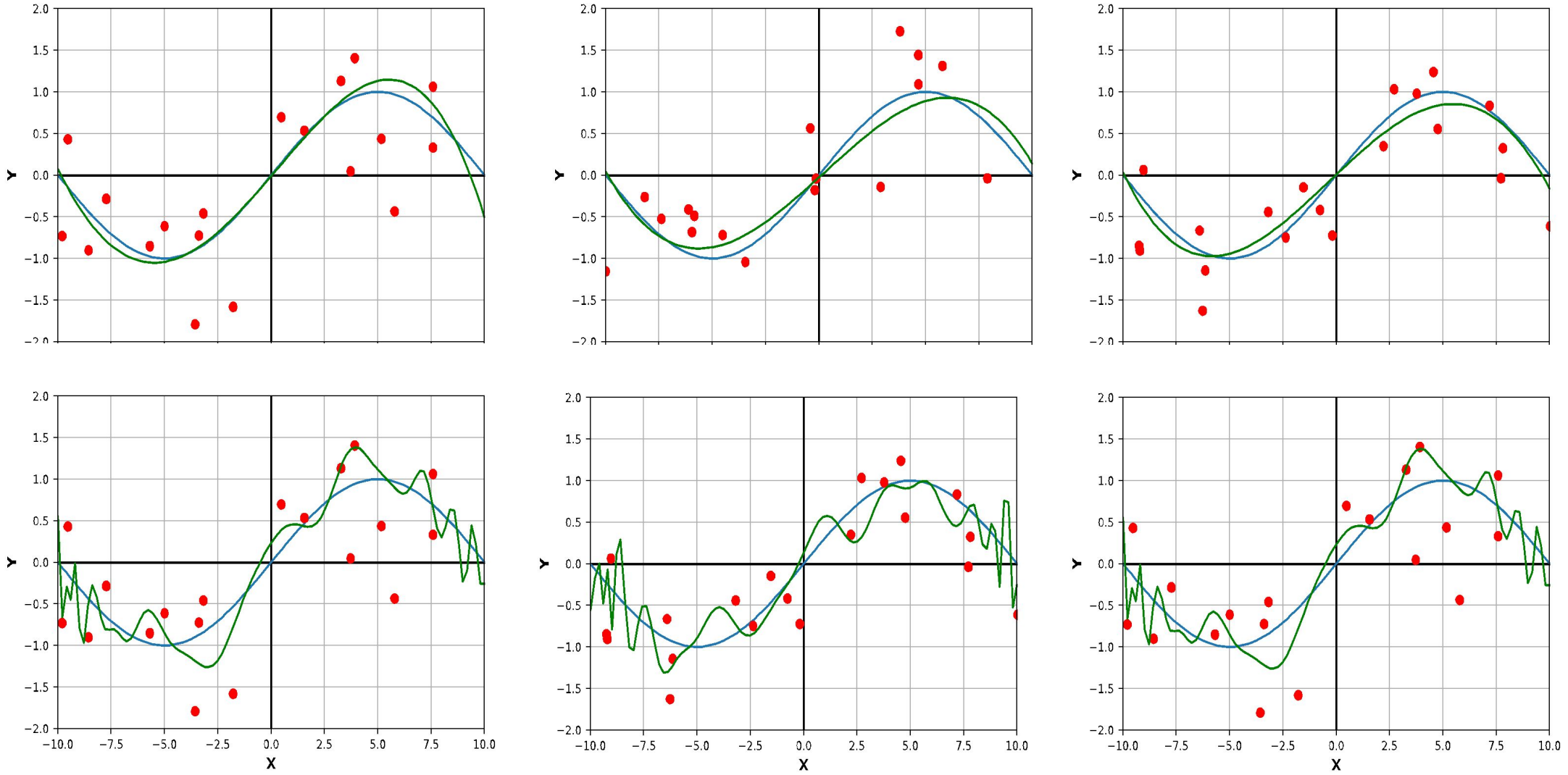


$\{1, x, x^2, ..., x^{99}, x^{100}\}$

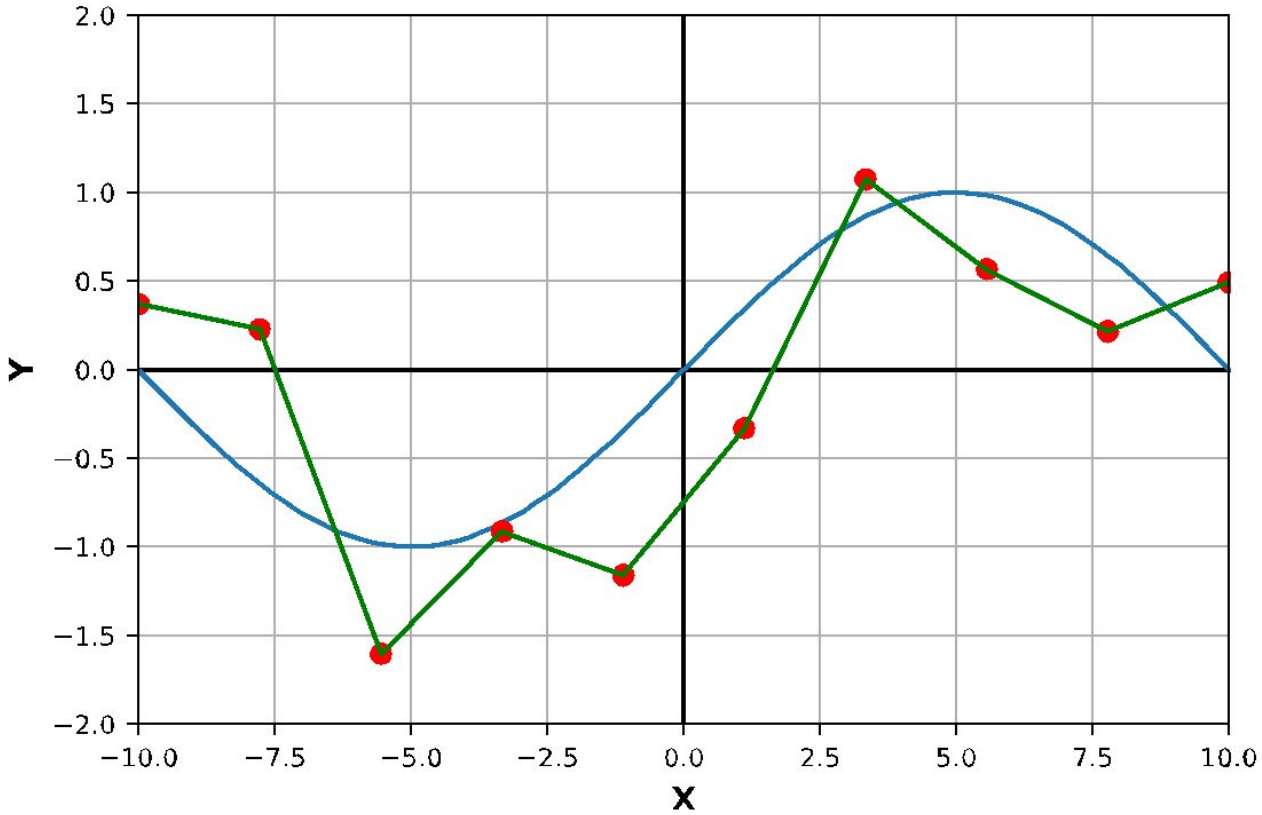


What is Bias and Variance?

$$\{1, x, x^2, x^3, x^4\}$$



Real Bad Overfit?



Bias-Variance Tradeoff

- So far we have minimized the error (loss) with respect to **training data**
  - Low training error does not imply good expected performance: **over-fitting**
- We would like to reason about the **expected loss (Prediction Risk)** over:
  - Training Data: $\{(y_1, x_1), \ldots, (y_n, x_n)\}$
  - Test point: (y_*, x_*)
- We will decompose the expected loss into:

$$\mathbf{E}_{D,(y_*,x_*)}\left[(y_* - f(x_*|D))^2\right] = \text{Noise} + \text{Bias}^2 + \text{Variance}$$

Bias-Variance Tradeoff

If we denote the variable we are trying to predict as Y and our covariates as X, we may assume that there is a relationship relating one to the other such as $Y = f(X) + \epsilon$ where the error term ϵ is normally distributed with a mean of zero like so $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon)$.

We may estimate a model $\hat{f}(X)$ of $f(X)$ using linear regressions or another modeling technique. In this case, the expected squared prediction error at a point x is:

$$Err(x) = E\left[(Y - \hat{f}(x))^2\right]$$

This error may then be decomposed into bias and variance components:

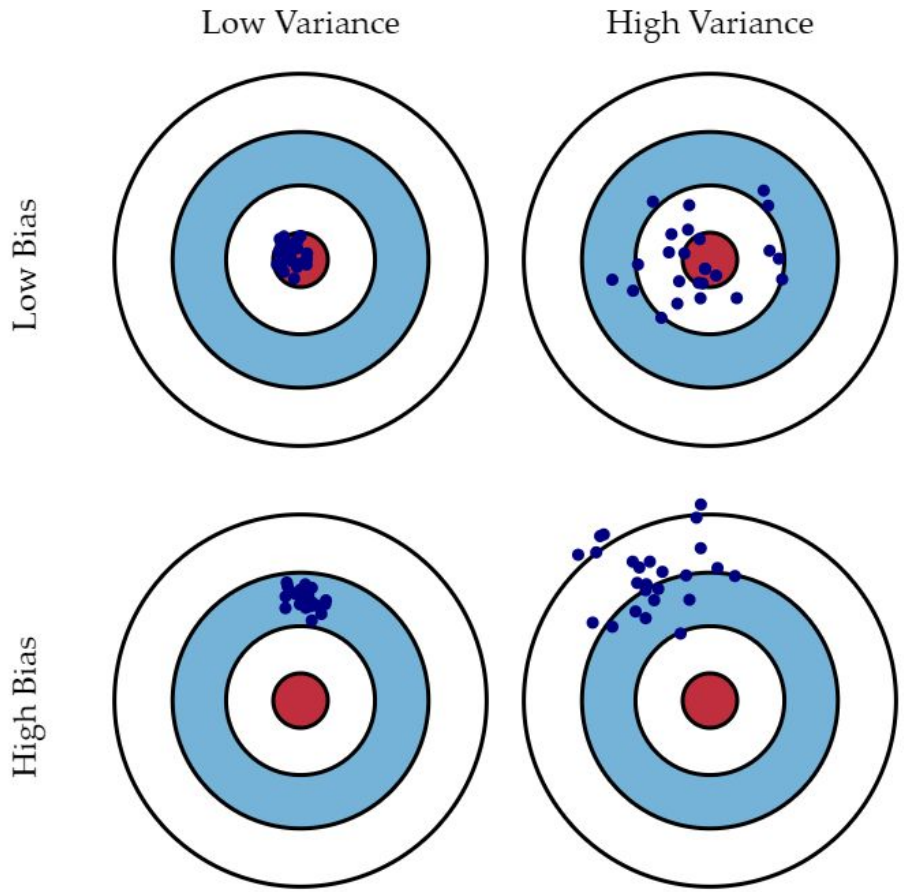
$$Err(x) = \left(E[\hat{f}(x)] - f(x)\right)^2 + E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right] + \sigma_e^2$$

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

That third term, irreducible error, is the noise term in the true relationship that cannot fundamentally be reduced by any model. Given the true model and infinite data to calibrate it, we should be able to reduce both the bias and variance terms to 0. However, in a world with imperfect models and finite data, there is a tradeoff between minimizing the bias and minimizing the variance.

Bias and Variance

We can plot four different cases representing combinations of both high and low bias and variance.



from https://scott.fortmann-roe.com/docs/BiasVariance.html

Bias Variance Plot

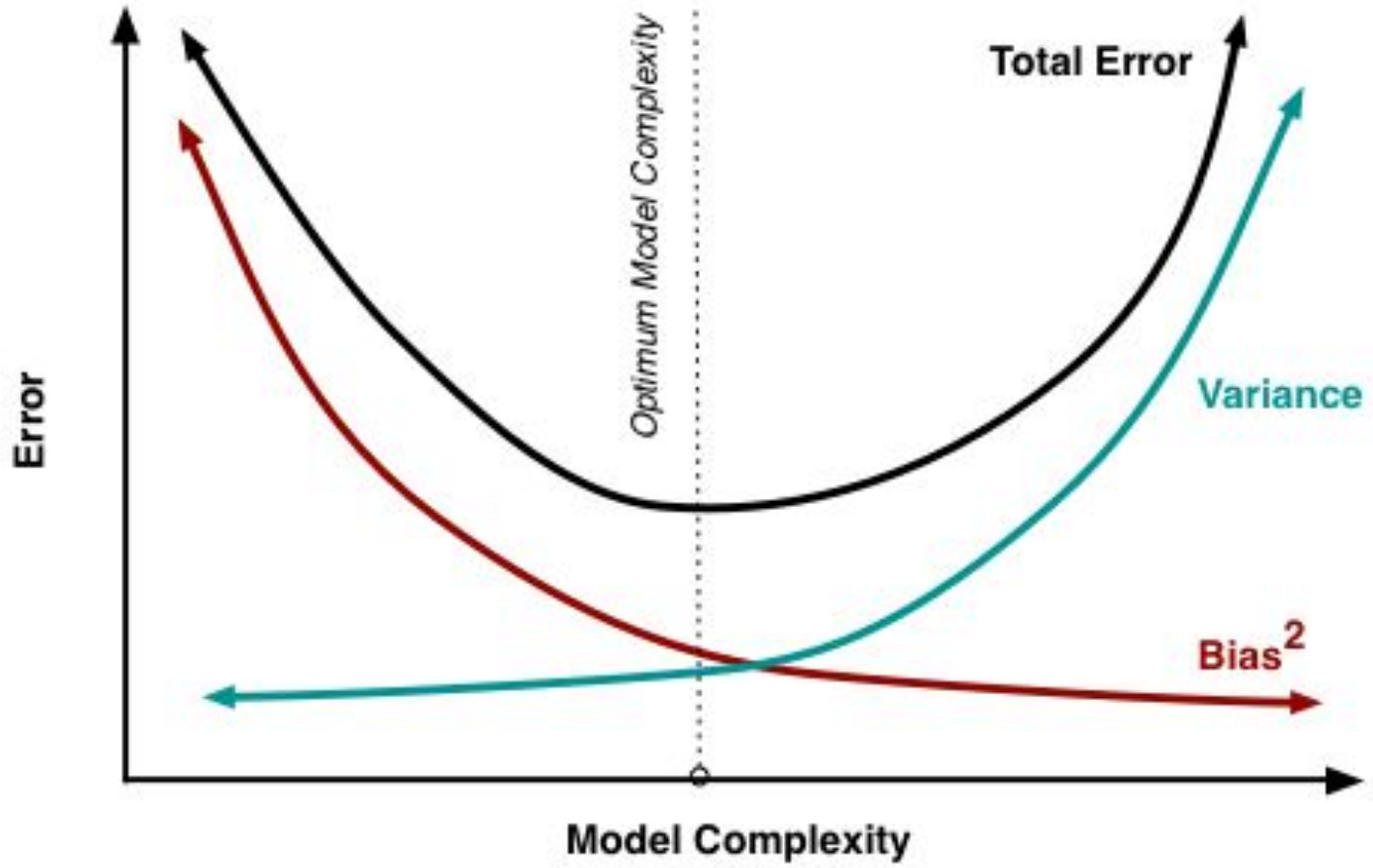


Image from http://scott.fortmann-roe.com/docs/BiasVariance.html

Managing Bias Variance

The rule is we should seek **lower bias** and **lower variance**.

In general what we really care about is **lower overall error**, not the specific decomposition.

At its root, dealing with bias and variance is really about **dealing with over- and under-fitting**.

What to do in practice?

Keep a validation/test datasets to check the model performance.

Data : manipulate the data through data augmentation and resampling techniques.

Complexity: manipulate the model complexity by adding or reducing parameters.

Learned model: use regularization techniques.

Practice Time!

Regularization

Slides from: https://harvard-iacs.github.io/2019-CS109A/lectures/lecture8/presentation/Lecture8a_Regularization.pptx

Regularization: An Overview

$$L_{reg}(\beta) = L(\beta) + \lambda R(\beta),$$

LASSO Regression

Since we wish to discourage extreme values in model parameter, we need to choose a regularization term that penalizes parameter magnitudes. For our loss function, we will again use MSE.

Together our regularized loss function is:

$$L_{LASSO}(\beta) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \boldsymbol{\beta}^\top \boldsymbol{x}_i|^2 + \lambda \sum_{j=1}^{J}|\beta_j|.$$

Note that $\sum_{j=1}^{J}|\beta_j|$ is the $\boldsymbol{l}_1$ norm of the vector $\boldsymbol{\beta}$

$$\sum_{j=1}^{J}|\beta_j| = \|\boldsymbol{\beta}\|_1$$

Ridge Regression

Alternatively, we can choose a regularization term that penalizes the squares of the parameter magnitudes. Then, our regularized loss function is:

$$L_{Ridge}(\beta) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \boldsymbol{\beta}^\top \boldsymbol{x}_i|^2 + \lambda \sum_{j=1}^{J} \beta_j^2.$$

Note that $\sum_{j=1}^{J} |\beta_j|^2$ is the square of the $\boldsymbol{l_2}$ norm of the vector $\boldsymbol{\beta}$

$$\sum_{j=1}^{J} \beta_j^2 = \|\boldsymbol{\beta}\|_2^2$$

Choosing λ

In both ridge and LASSO regression, we see that the larger our choice of the **regularization parameter** λ, the more heavily we penalize large values in $\boldsymbol{\beta}$,

- If λ is close to zero, we recover the MSE, i.e. ridge and LASSO regression is just ordinary regression.
- If λ is sufficiently large, the MSE term in the regularized loss function will be insignificant and the regularization term will force $\boldsymbol{\beta}_{\text{ridge}}$ and $\boldsymbol{\beta}_{\text{LASSO}}$ to be close to zero.

To avoid ad-hoc choices, we should select λ using cross-validation.

Ridge, LASSO - Computational complexity

Solution to ridge regression:

$$\beta = (X^T X + \lambda I)^{-1} X^T Y$$

The solution to the LASSO regression:

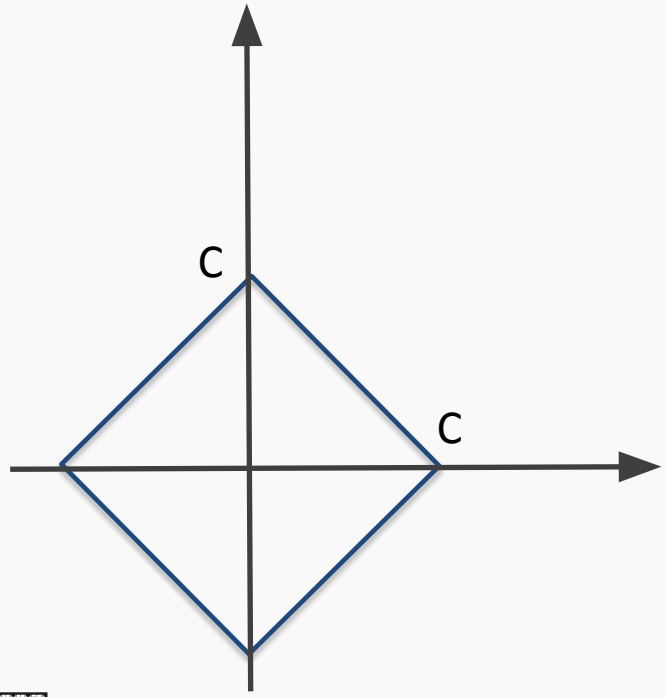
LASSO has no conventional analytical solution, as the L1 norm has no derivative at 0. We can, however, use the concept of subdifferential or subgradient to find a manageable expression.

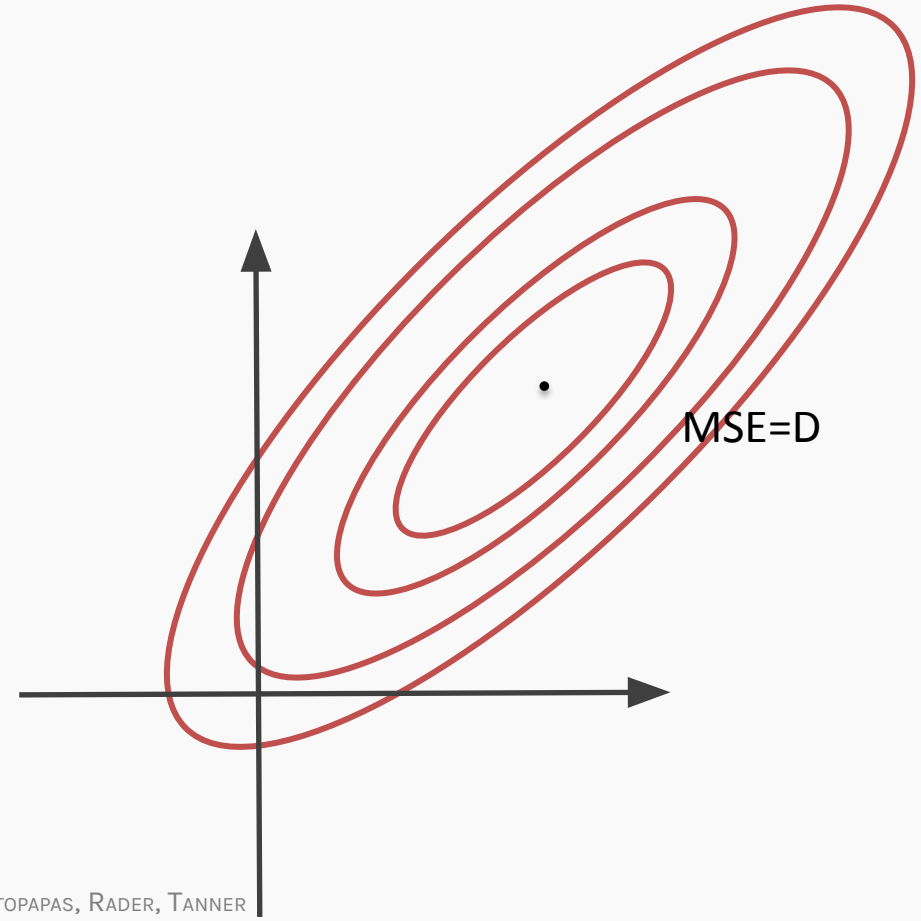
Regularization Parameter with a Validation Seet

The solution of the Ridge/Lasso regression involves three steps:

- Select λ
- Find the minimum of the ridge/Lasso regression loss function (using the formula for ridge) and record the *MSE* **on the validation set**.
- Find the λ that gives the smallest *MSE*

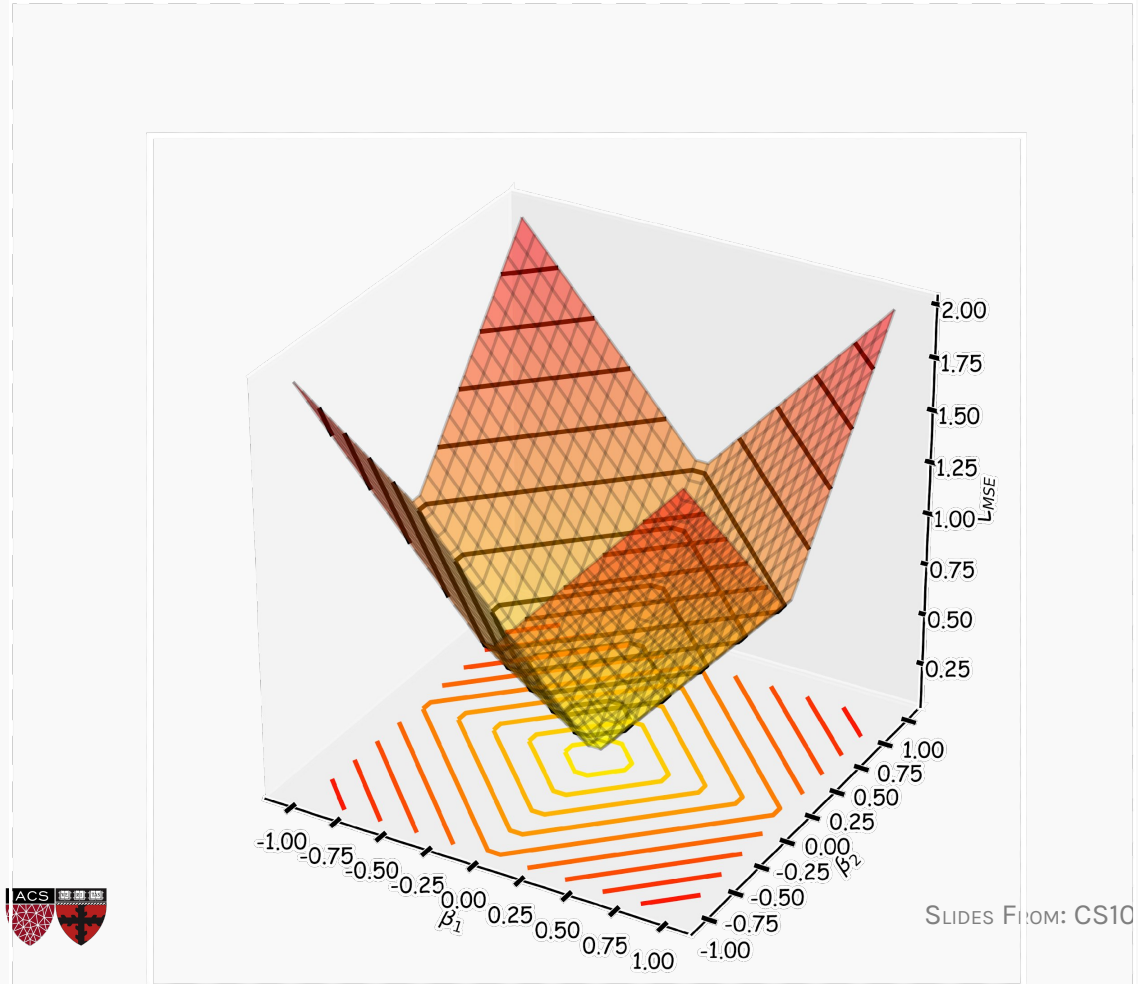
The Geometry of Regularization (LASSO)

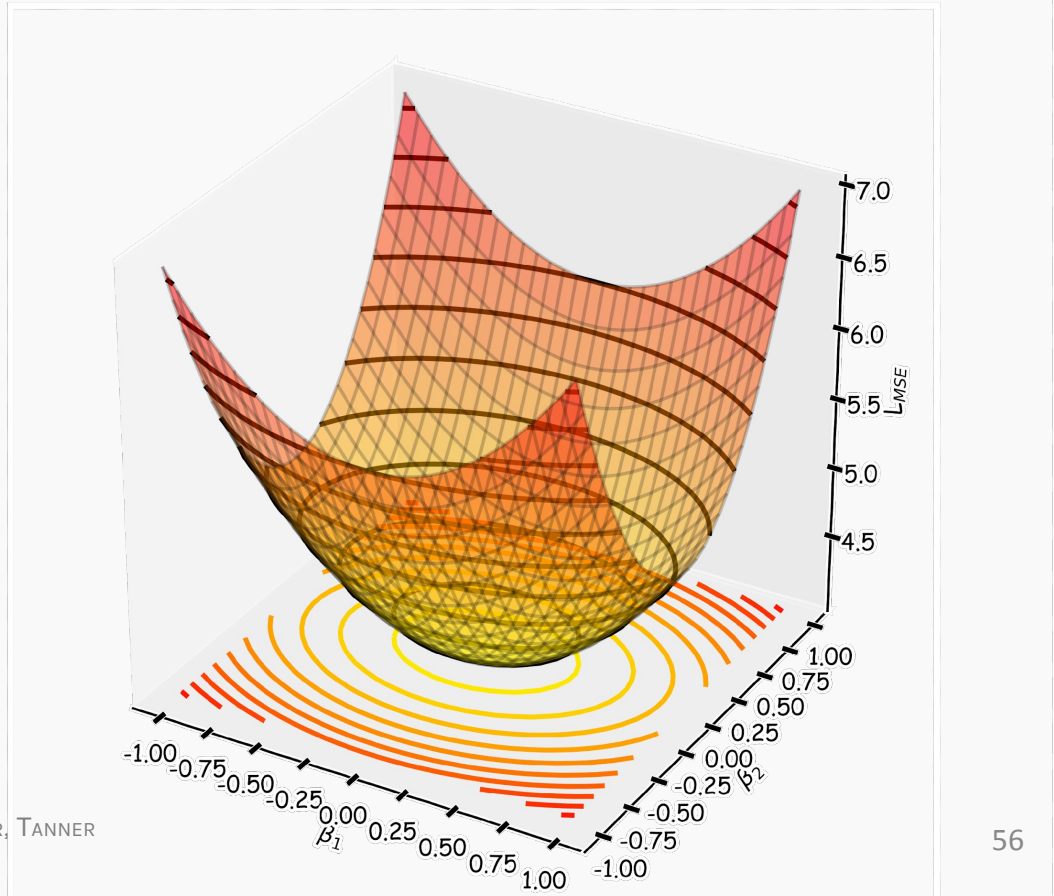




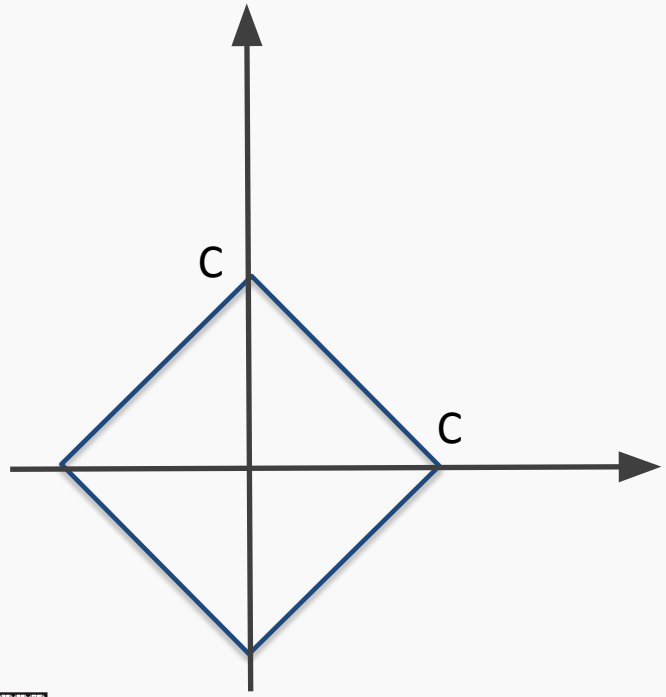
Slides From: CS109A, Protopapas, Rader, Tanner

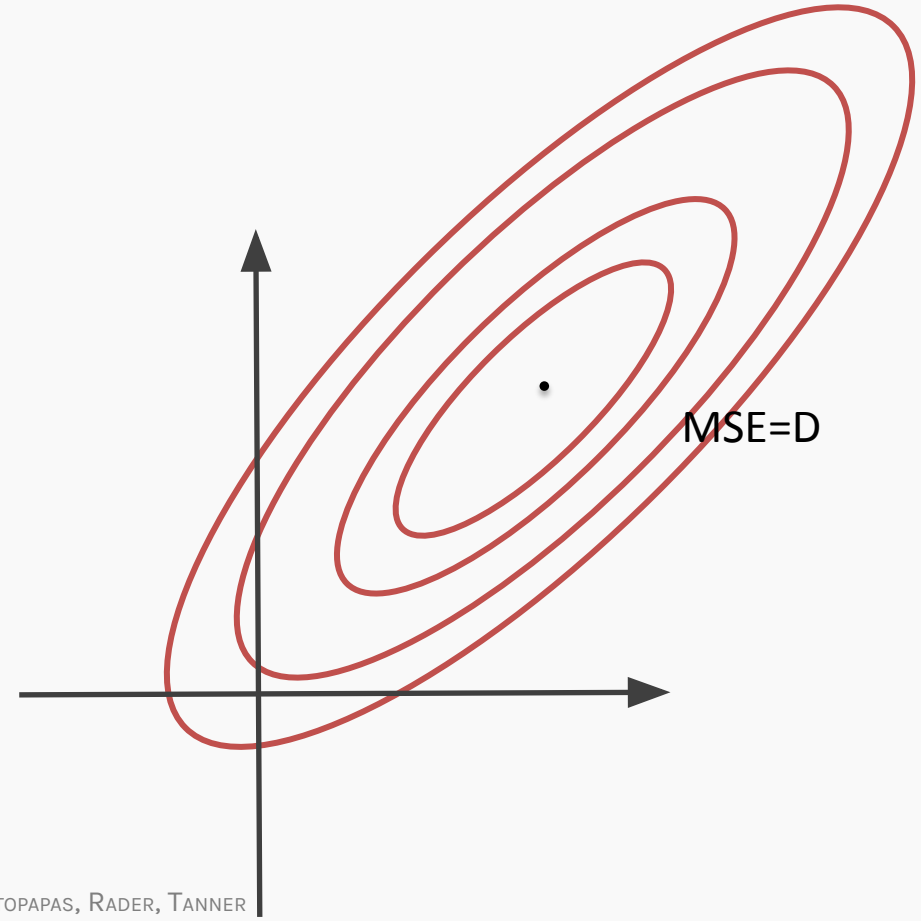
The Geometry of Regularization (LASSO)





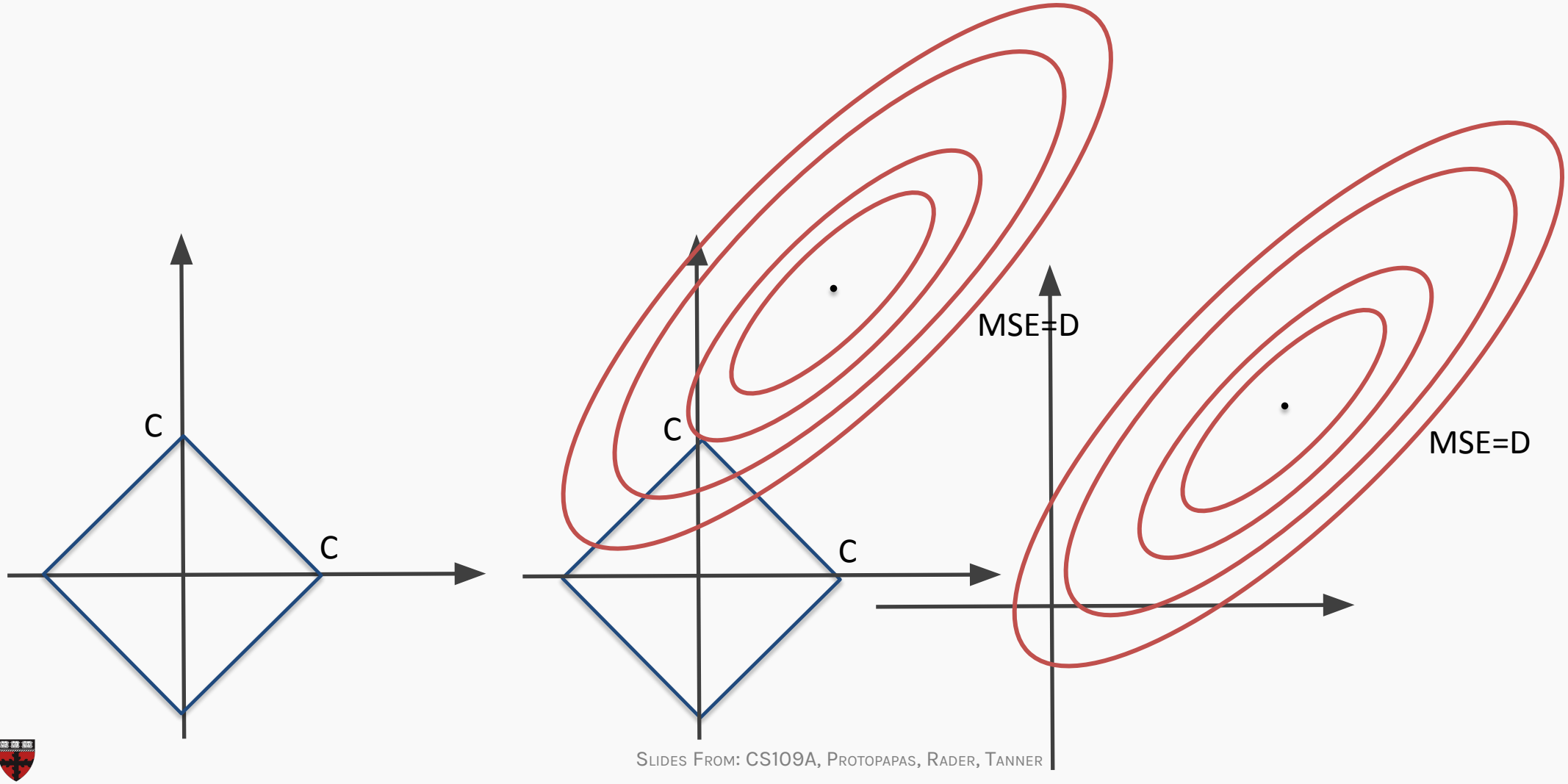
The Geometry of Regularization (LASSO)





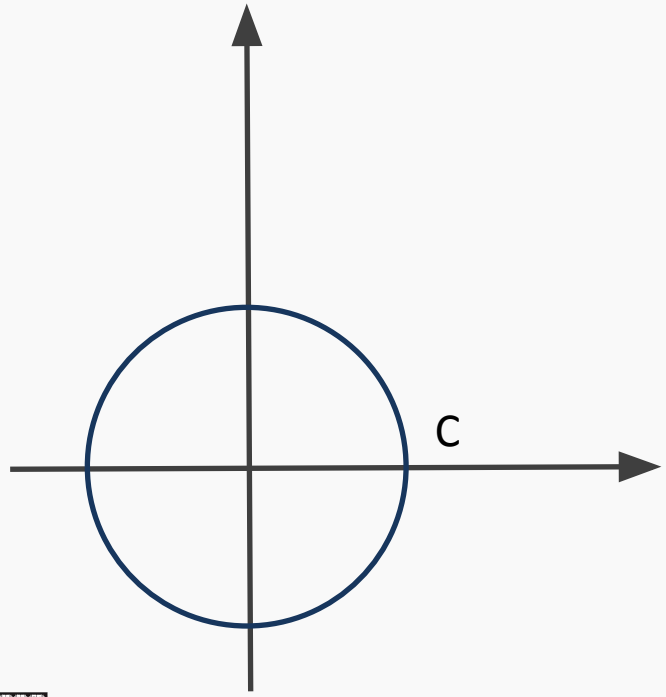
Slides From: CS109A, Protopapas, Rader, Tanner

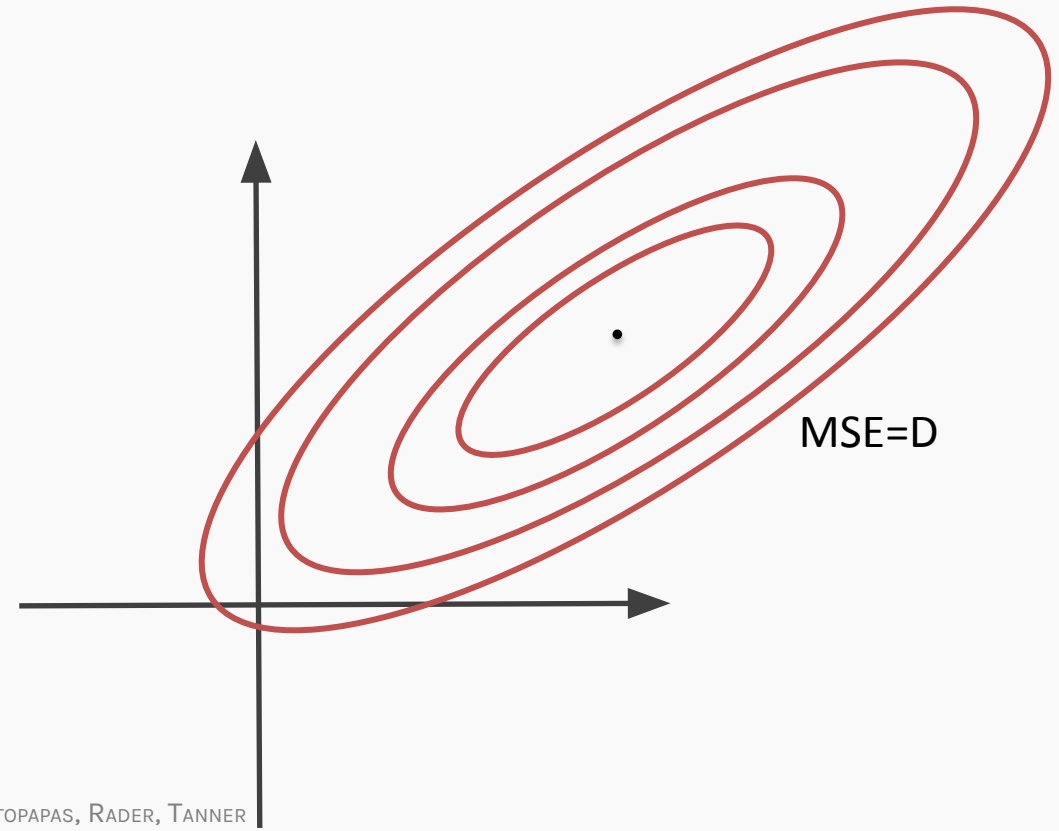
The Geometry of Regularization (LASSO)



Slides From: CS109A, Protopapas, Rader, Tanner

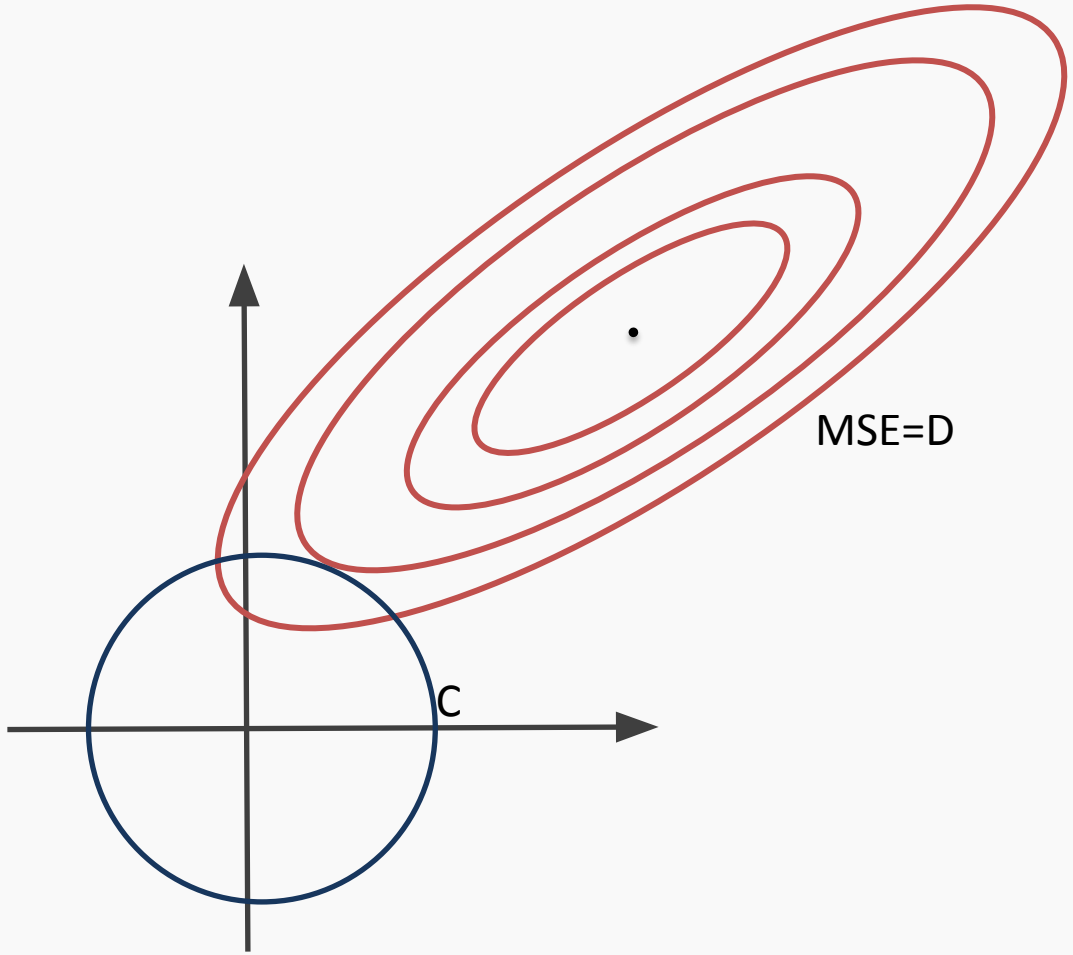
The Geometry of Regularization (Ridge)



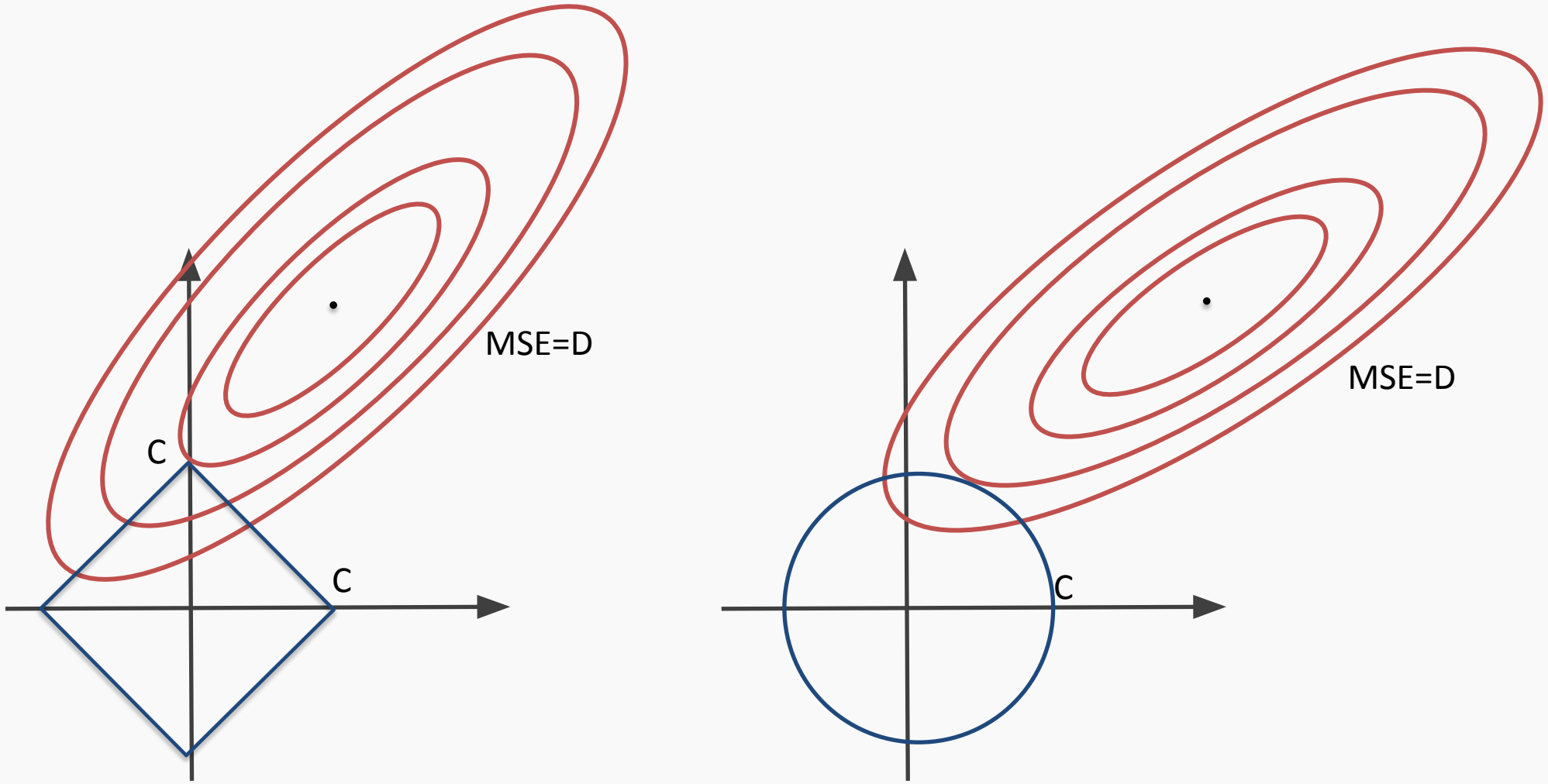


Slides From: CS109A, Protopapas, Rader, Tanner

The Geometry of Regularization (Ridge)



The Geometry of Regularization



Examples

In [ ]:
```
from sklearn.linear_model import Lasso
```

In [22]:
```
lasso_regression = Lasso(alpha=1.0, fit_intercept=True)
lasso_regression.fit(np.vstack((X_train, X_val)), np.hstack((y_train, y_val)))

print('Lasso regression model:\n {} + {}^T . x'.format(lasso_regression.intercept_, lasso_regression.coe
```

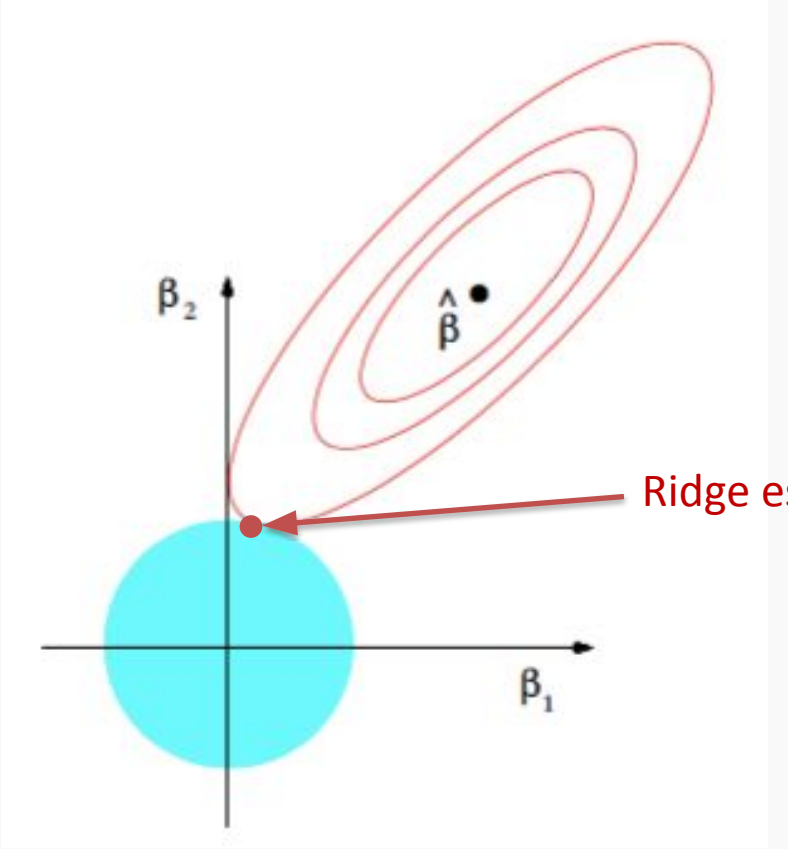
```
Lasso regression model:
 10.424895873901445 + [ 0.24482603  3.48164594  1.84836859 -0.06864603 -0.         -0.
 -0.02249766 -0.          0.          0.          0.          ]^T . x
```

In [ ]:
```
from sklearn.linear_model import Ridge
```

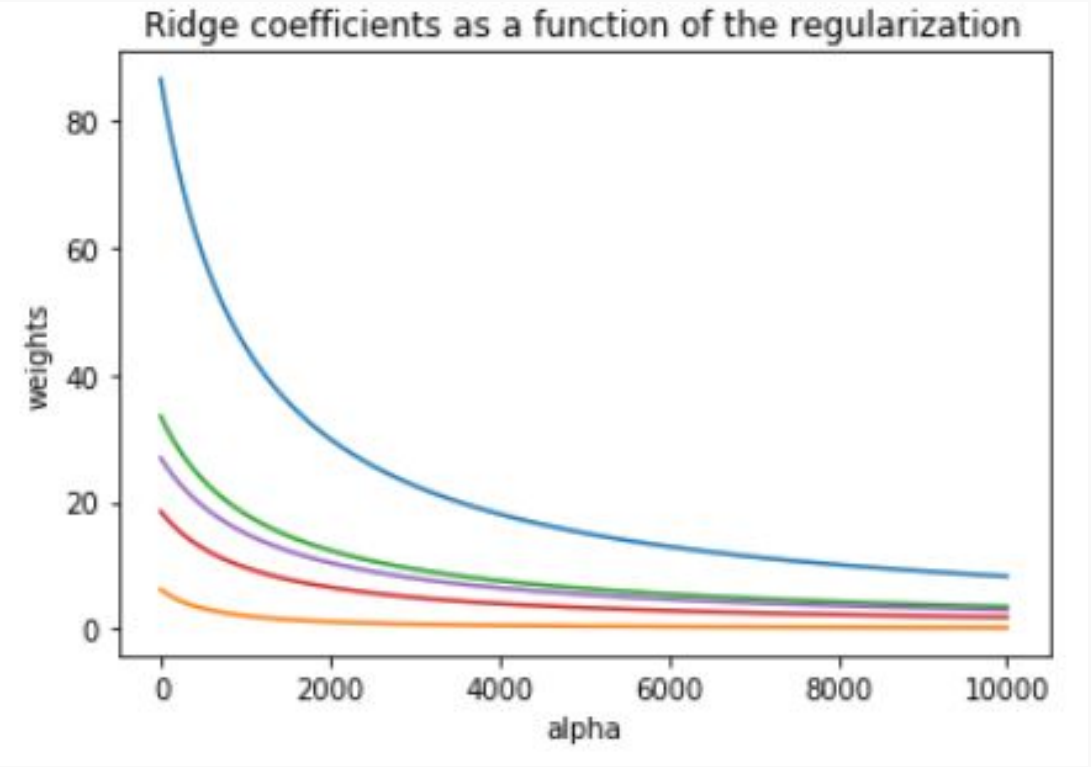
In [20]:
```
X_train = train[all_predictors].values
X_val = validation[all_predictors].values
X_test = test[all_predictors].values

ridge_regression = Ridge(alpha=1.0, fit_intercept=True)
ridge_regression.fit(np.vstack((X_train, X_val)), np.hstack((y_train, y_val)))

print('Ridge regression model:\n {} + {}^T . x'.format(ridge_regression.intercept_, ridge_regression.coe
```

```
Ridge regression model:
 -525.7662550875951 + [ 0.24007312  8.42566029  2.04098593 -0.04449172 -0.01227935  0.41902475
 -0.50397312 -4.47065168  4.99834262  0.          0.          0.29892679]^T . x
```

Ridge visualized

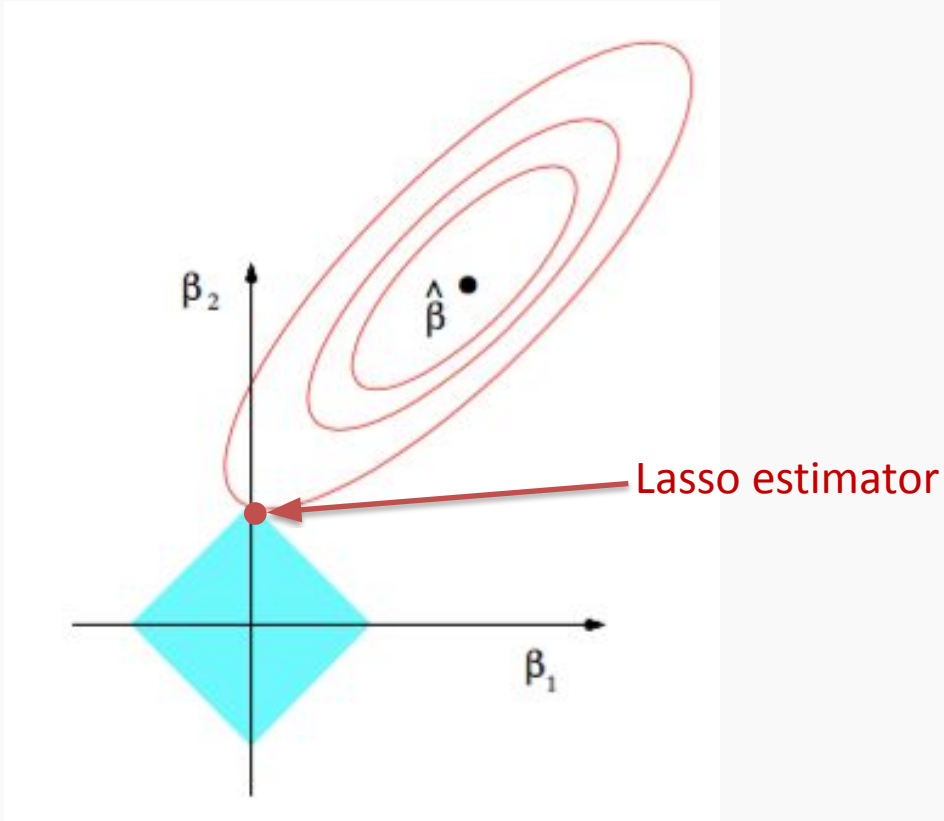


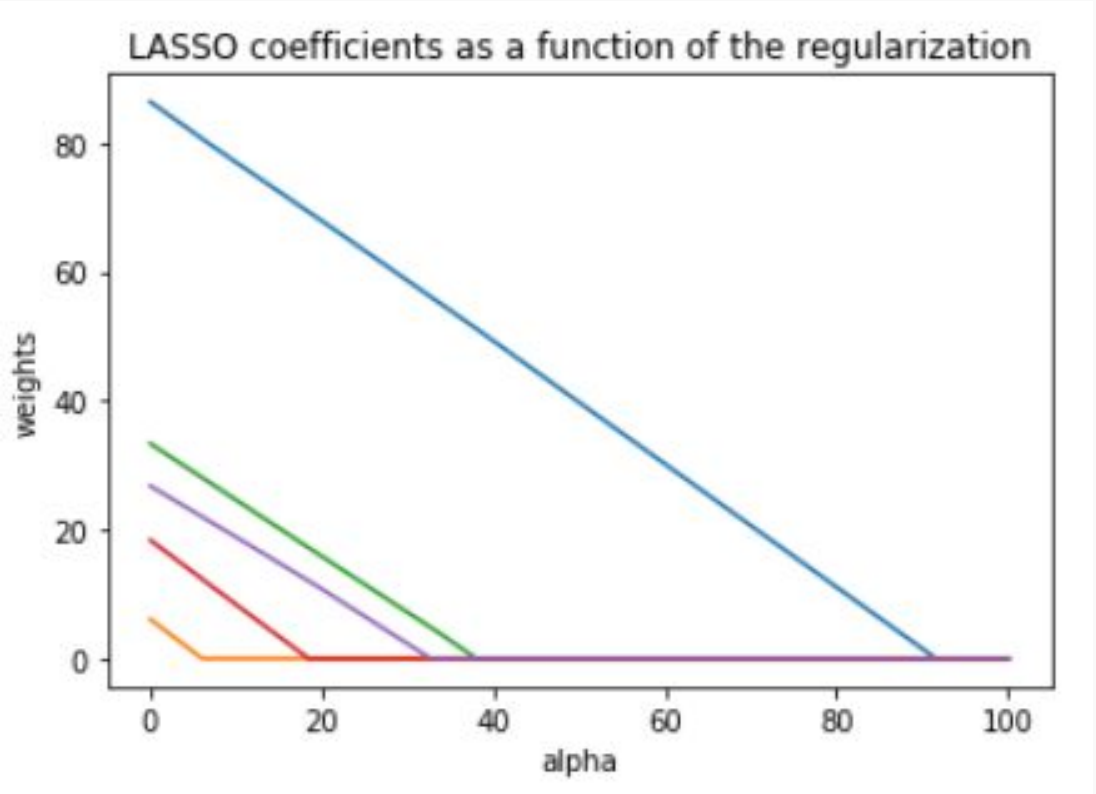
The ridge estimator is where the constraint and the loss intersect.



The values of the coefficients decrease as lambda increases, but they are not nullified.

LASSO visualized

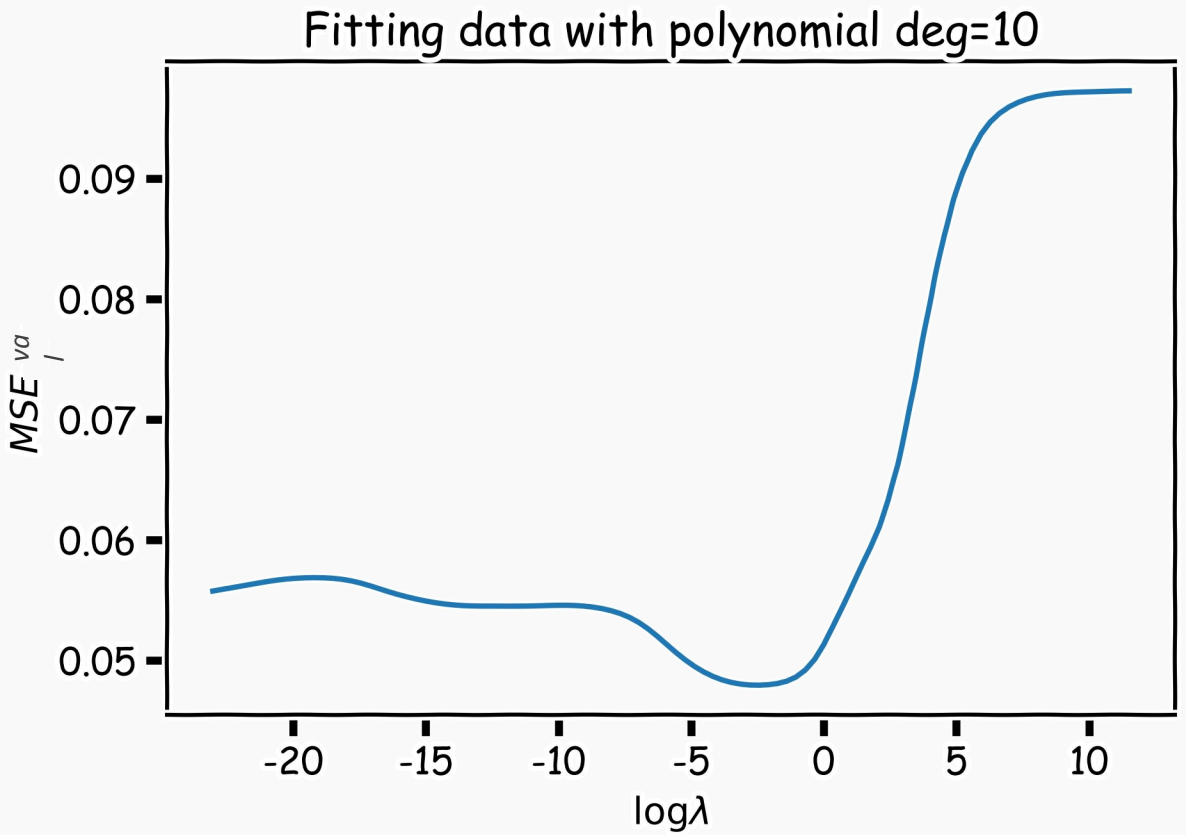




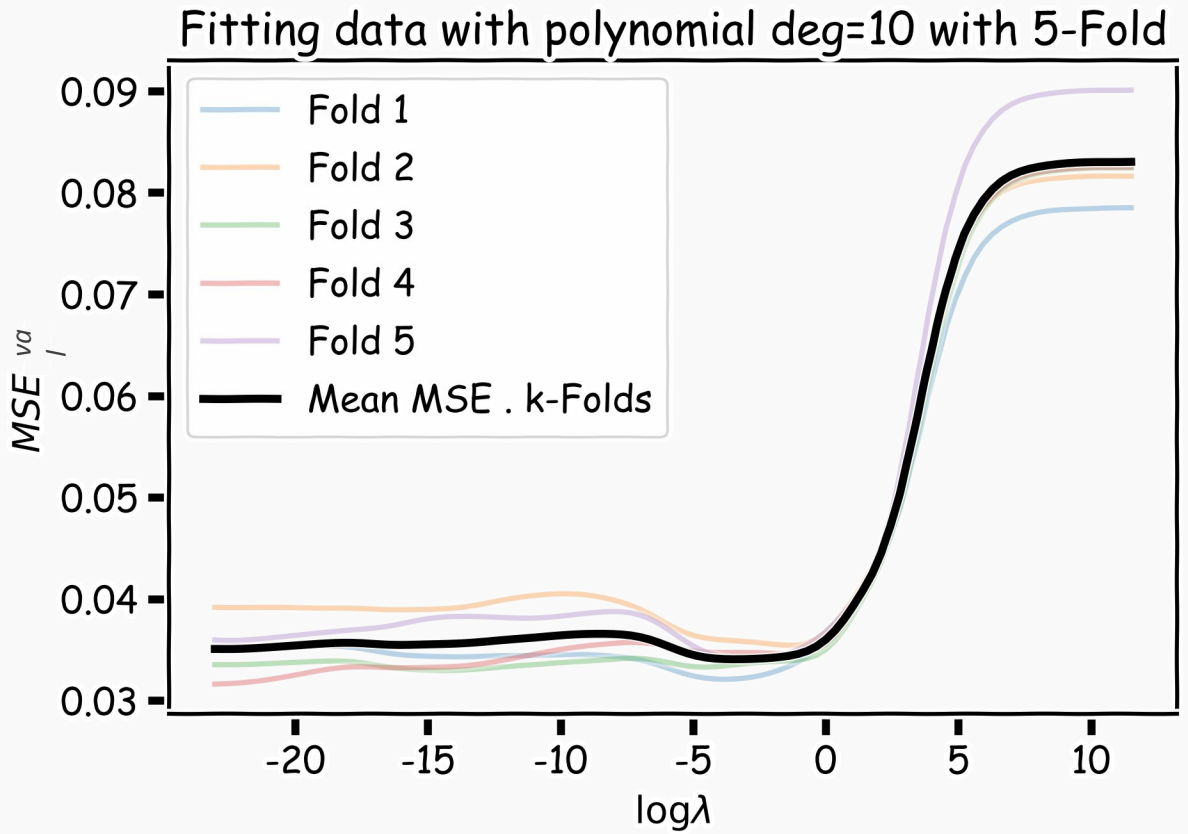
The Lasso estimator tends to zero out parameters as the OLS loss can easily intersect with the constraint on one of the axis.

The values of the coefficients decrease as lambda increases, and are nullified fast.

Ridge regularization with **validation** only



Ridge regularization with **validation** only: step by step



Variable Selection as Regularization

Since LASSO regression tend to produce zero estimates for a number of model parameters - we say that LASSO solutions are **sparse** - we consider LASSO to be a method for variable selection.

Many prefer using LASSO for variable selection (as well as for suppressing extreme parameter values) rather than stepwise selection, as LASSO avoids the statistic problems that arises in stepwise selection.

Question: What are the pros and cons of the two approaches?