Linear Regression

Hanwool Jeong

hwjeong@kw.ac.kr

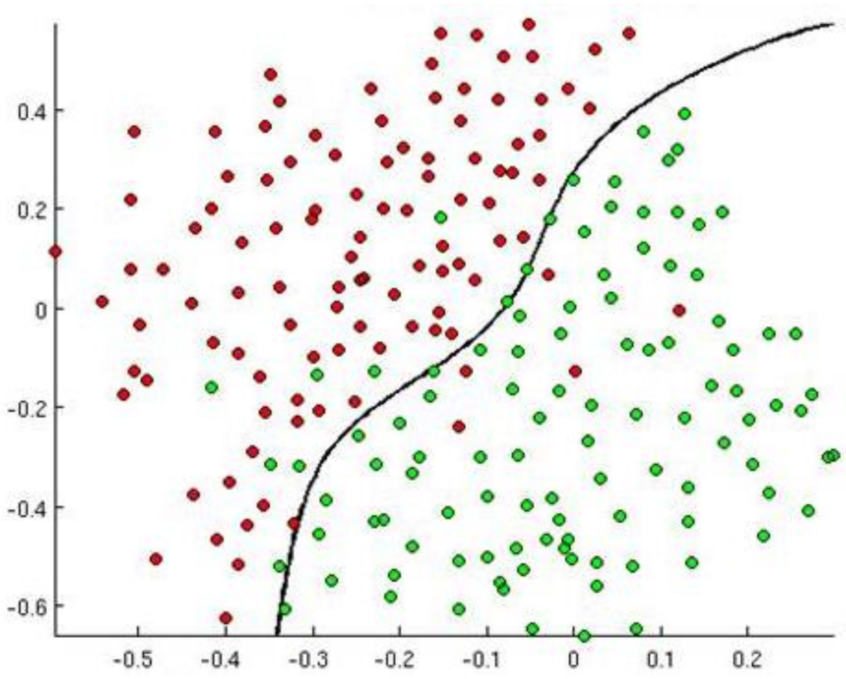
What We Will Learn

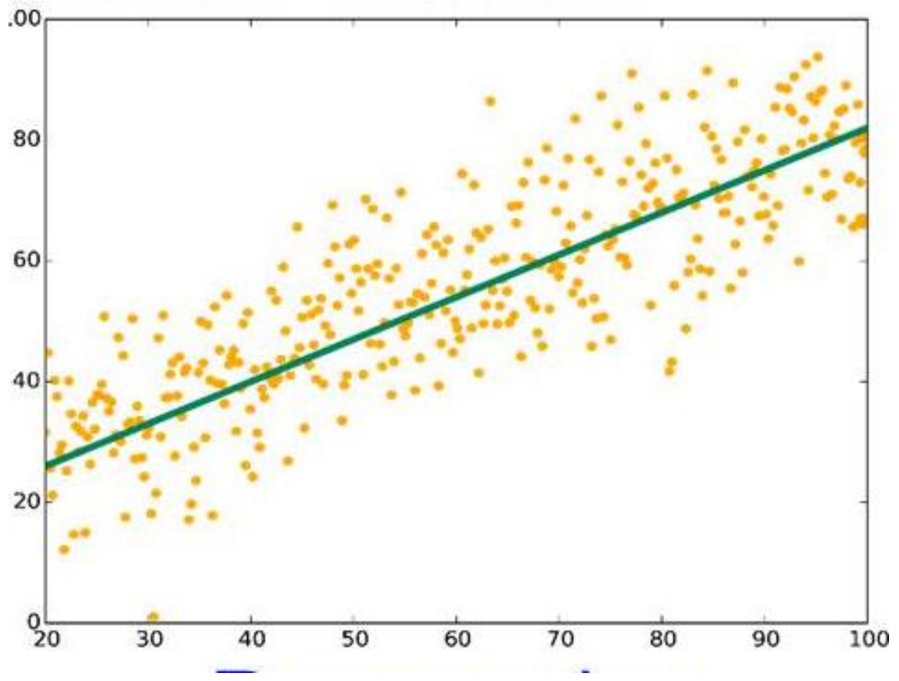
- First, focus on how **supervised learning** is realized!
  - You remember? **Training**, then **prediction**.
  - General procedure for "**training**"
  - How Probability/Linear Algebra/Calculus are used
- Terms used in ML
- Realizing ML with python codes

➔ Linear regression is good for start!

Revisit: Regression vs. Classification

- Regression is just like classification except the response variable is continuous.





Why We Need Matrix & Vector in ML

- In Iris flower example, the features are extracted in 4 variables: the length and the width of the sepals and petals, in centimeter

  ➔ This 4D data can be expressed with a vector

$$\xrightarrow{\textbf{Feature Extraction}} \quad \mathbf{x} = \begin{bmatrix} 8.1 \\ 2.8 \\ 5.0 \\ 1.1 \end{bmatrix}$$

- To contain multiple data vectors or process the vector, the matrix can be used.

Backgrounds for Linear Algebra; Matrix-Vector Multiplication

- The concept of weight? ➔ **Importance/Influence of each feature**
- Inner product?
- Usually feature vector is to be processed with a matrix A containing coefficient **(Why matrix? Not inner product?)**
- You know Norm? Transpose? Inverse Matrix? and Identity matrix?

Design Matrix; Expressing & Handling Many Feature Vectors

- Using subscribes, a number of feature vectors are expressed as:

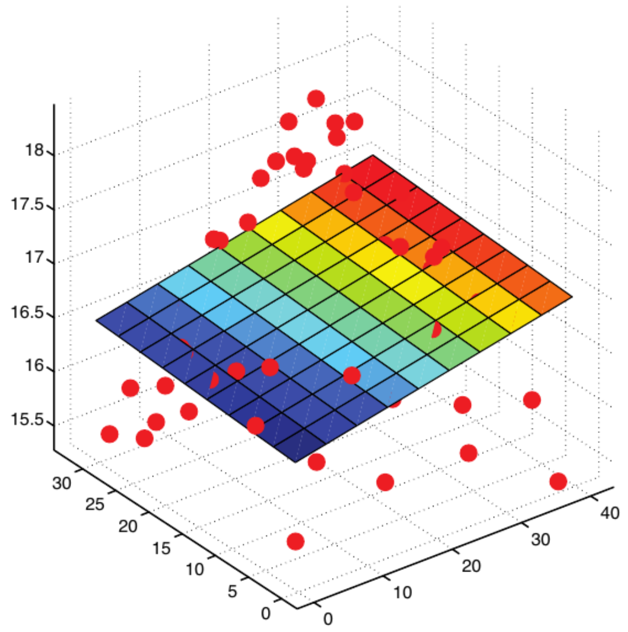
$$\mathbf{x}_1 = \begin{bmatrix} 8.1 \\ 2.8 \\ 5.0 \\ 1.1 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 3.2 \\ 1.5 \\ 1.2 \\ 0.5 \end{bmatrix}, \quad \mathbf{x}_3 = \begin{bmatrix} 5.1 \\ 2.6 \\ 1.3 \\ 0.2 \end{bmatrix}, \quad \mathbf{x}_4 = \begin{bmatrix} 4.1 \\ 3.2 \\ 2.3 \\ 1.1 \end{bmatrix}, \ldots , \quad \mathbf{x}_{1000} = \begin{bmatrix} 6.3 \\ 3.0 \\ 5.3 \\ 1.8 \end{bmatrix}$$

- **N×D design matrix** containing *N* feature vectors, ***X*** is defined as

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^{\mathrm{T}} \\ \boldsymbol{x}_2^{\mathrm{T}} \\ \boldsymbol{x}_3^{\mathrm{T}} \\ \boldsymbol{x}_4^{\mathrm{T}} \\ \cdots \\ \boldsymbol{x}_{1000}^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} 8.1 & 2.8 & 5.0 & 1.1 \\ 3.2 & 1.5 & 1.2 & 0.5 \\ 5.1 & 2.6 & 1.3 & 0.2 \\ 4.1 & 3.2 & 2.3 & 5.3 \\ .. & .. & .. & .. \\ 6.3 & 3.0 & 5.3 & 1.8 \end{bmatrix}$$

Application for Linear Regression

- Midterm score vs. working hour
- Rental price of house vs. area
- Unemployment rate vs. age



What we should determine is …

Linear Regression Model

- Linear regression model can be defined as

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots + \theta_D x_D$$

  - $\hat{y}$: predicted value
  - D : the number of feature (dimension)
  - x_i : ith feature value
  - θ_j : jth model parameter

- Or with vector expression using inner product,

$$\hat{y} = h_\theta(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x}$$

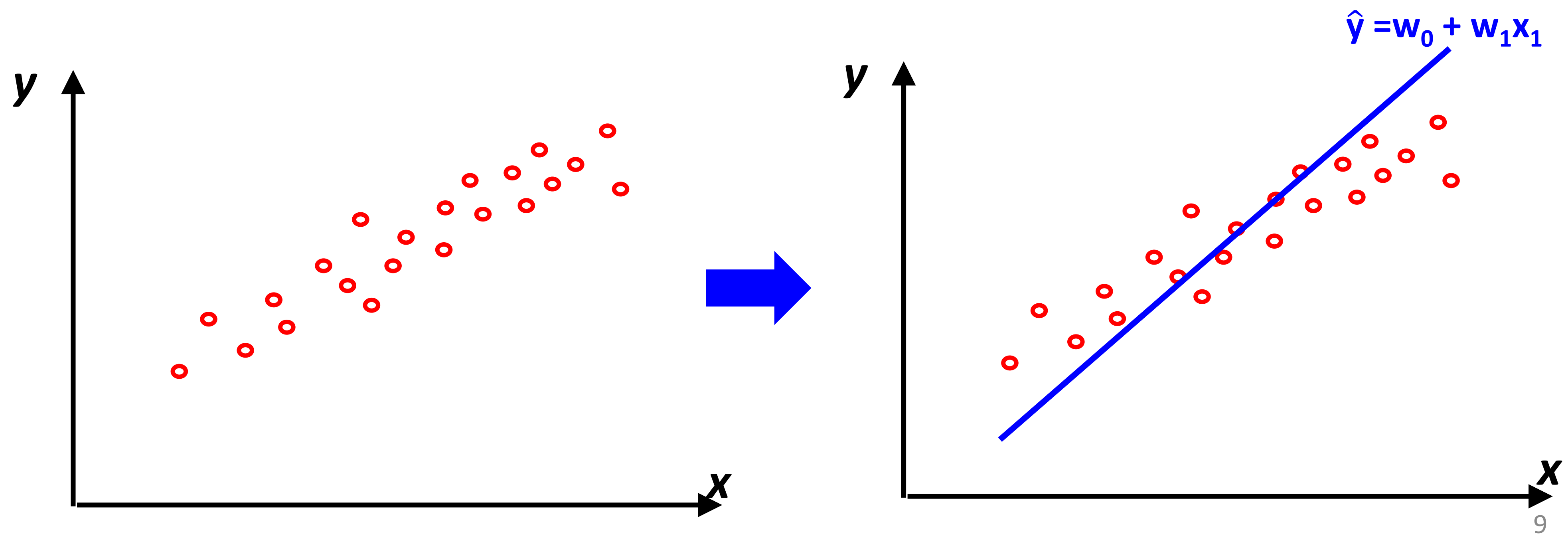
  - $x_0 = 1$
  - h_θ : **hypothesis** function with model parameter vector θ

D=1 Example

- We will use the vector w for the parameter vector θ to stand for "weight." Then the best fit curve can be expressed:

  $\hat{y} = h_w(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} = w_0 + w_1x_1$ ➔ You should determine ***w***

- How?



Residual Sum of Squares

- Residual sum of squares is defined by

$$\mathrm{RSS}(\mathbf{w}) \triangleq \sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2$$

- Or mean of RSS, the mean square (MSE) is

$$\mathrm{MSE}(\boldsymbol{w}) = \frac{\mathrm{RSS}(\boldsymbol{w})}{N}$$

- We will use RSS as the loss function or cost function, which evaluates how well something fits the data.
- RSS or MSE is just one of many different loss (or cost) functions

Finding the Optimal Weight

- To find MLE, or the optimal value of **w** MSE should be minimized.
- To find the minimum of MSE, the gradient of MSE is derived.

$$\mathrm{MSE}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \boldsymbol{w}^T\boldsymbol{x}_i)^2 \Rightarrow \nabla MSE(\boldsymbol{w}) = -\frac{2}{N}\sum_{i=1}^{N}(y_i - \boldsymbol{w}^T\boldsymbol{x}_i)\,\boldsymbol{x}_i = \vec{\mathbf{0}}$$

$$\sum_{i=1}^{N}(y_i - \boldsymbol{w}^T\boldsymbol{x}_i)\,\boldsymbol{x}_i = \sum_{i=1}^{N} y_i\boldsymbol{x_i} - \sum_{i=1}^{N}(\boldsymbol{w}^T\boldsymbol{x_i})\boldsymbol{x_i} = \vec{\mathbf{0}}$$

$$\sum_{i=1}^{N}(\boldsymbol{w}^T\boldsymbol{x_i})\boldsymbol{x_i} = \sum_{i=1}^{N} y_i\boldsymbol{x_i}$$

- With the definition of N-dimensional vector **y** = $[y_1, y_2, .. y_N]^T$ and the design matrix X which is predefined ,

$$\mathbf{X}^T\mathbf{X}\mathbf{w} = \mathbf{X}^T\mathbf{y}$$ **: Normal equation**

Finding the Optimal Weight (cont'd)

- The corresponding solution $\widehat{w}$ to the normal equation is called ordinary least squares or OLS solution.

$$\mathbf{X}^T\mathbf{X}\mathbf{w} = \mathbf{X}^T\mathbf{y} \quad \Rightarrow \quad \hat{\mathbf{w}}_{OLS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

- With larger D, N or higher order regression (instead of linear) make it highly challenging to derive $\widehat{w}_{\text{OLS}}$.
  ➔ In other words, computational complexity is increased.
- Is there any numerical way to find it?

Revisit the Procedure to Find $\hat{\boldsymbol{w}}_{\boldsymbol{OLS}}$

- Finding $\hat{\boldsymbol{w}}_{\text{OLS}}$ starts with

$$\text{Minimizing } \mathrm{MSE}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \boldsymbol{w}^T\boldsymbol{x}_i)^2$$

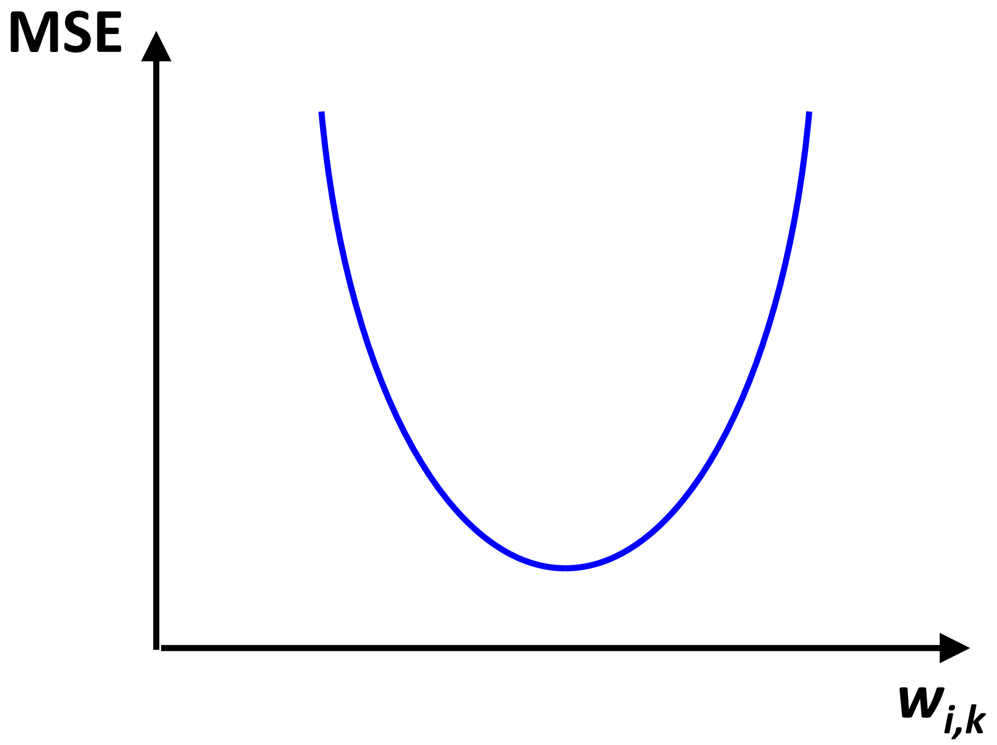
or

$$\nabla MSE(\boldsymbol{w}) = -\frac{2}{N}\sum_{i=1}^{N}(y_i - \boldsymbol{w}^T\boldsymbol{x}_i)\,\boldsymbol{x}_i = \overrightarrow{\mathbf{0}}$$

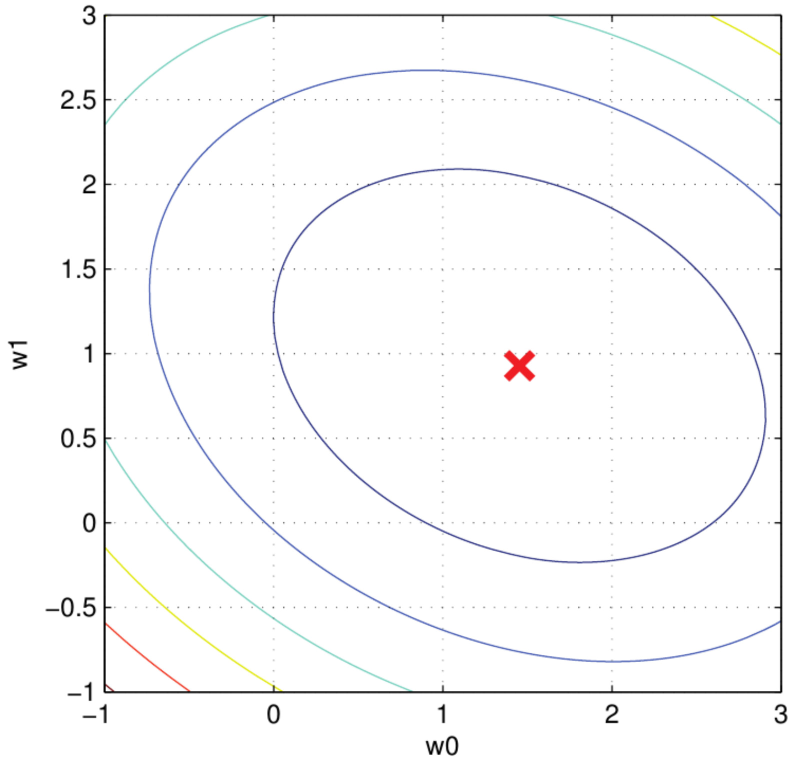
- Note that is ∇MSE is the vector, what dose the value of each component mean?
- It directs how **w** should be changed to end up with $\nabla MSE(\boldsymbol{w})$ = 0.

Gradient Descent Method

- $\mathrm{MSE}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \boldsymbol{w}^T\boldsymbol{x}_i)^2$
- $w_{\text{next}} = w_{\text{present}} - \eta\nabla MSE(\boldsymbol{w})$ ← η : learning rate

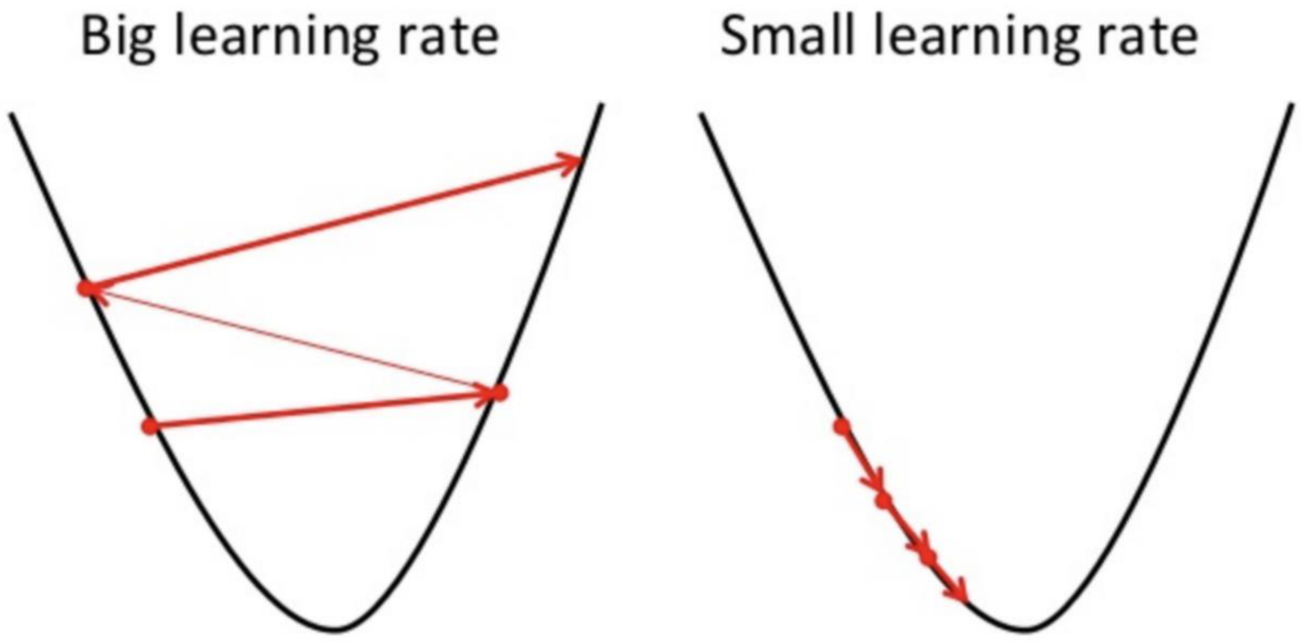


Contour of MSE vs. (w0, w1)



Effect of Learning Rate

- The general strategy is to start with a relatively large learning rate and make it smaller with each step ➔ learning schedule

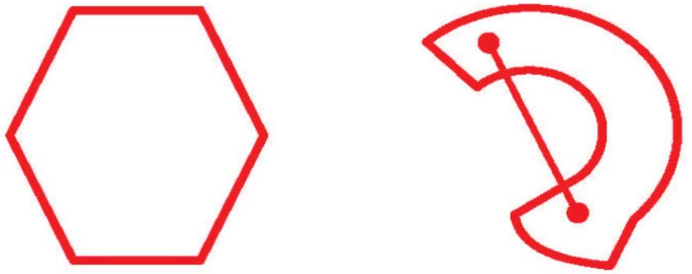


Convexity

- We say a set S is convex if any θ, θ' ∈ S, we have

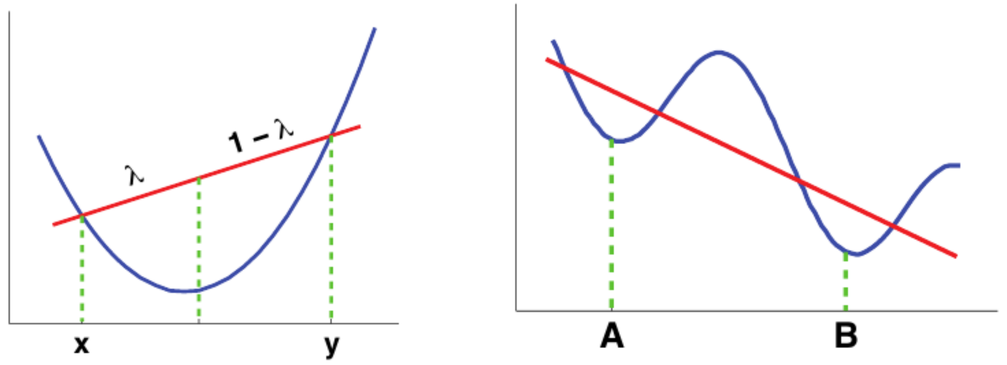
  $$\lambda\theta + (1-\lambda)\theta' \in S, \forall \lambda \in [0,1]$$

- If we draw a line from θ to θ', all points on the line lie inside the set.



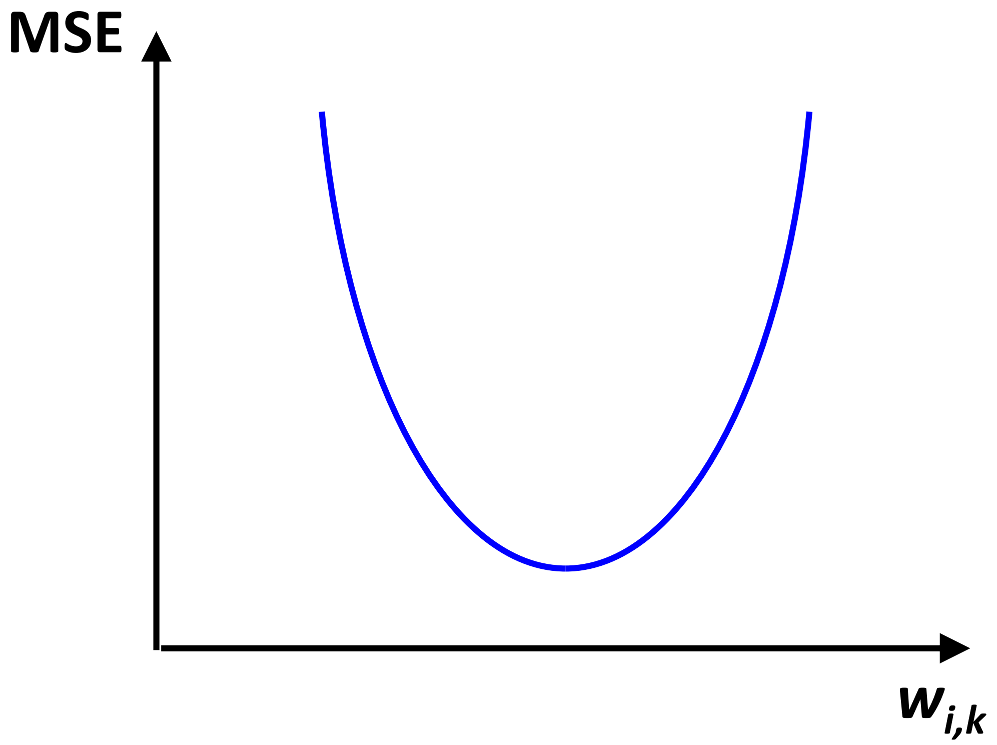
- Function f(θ) is called if it is defined on a convex set S, and for any θ, θ' ∈ S and any λ ∈ [0,1] we have

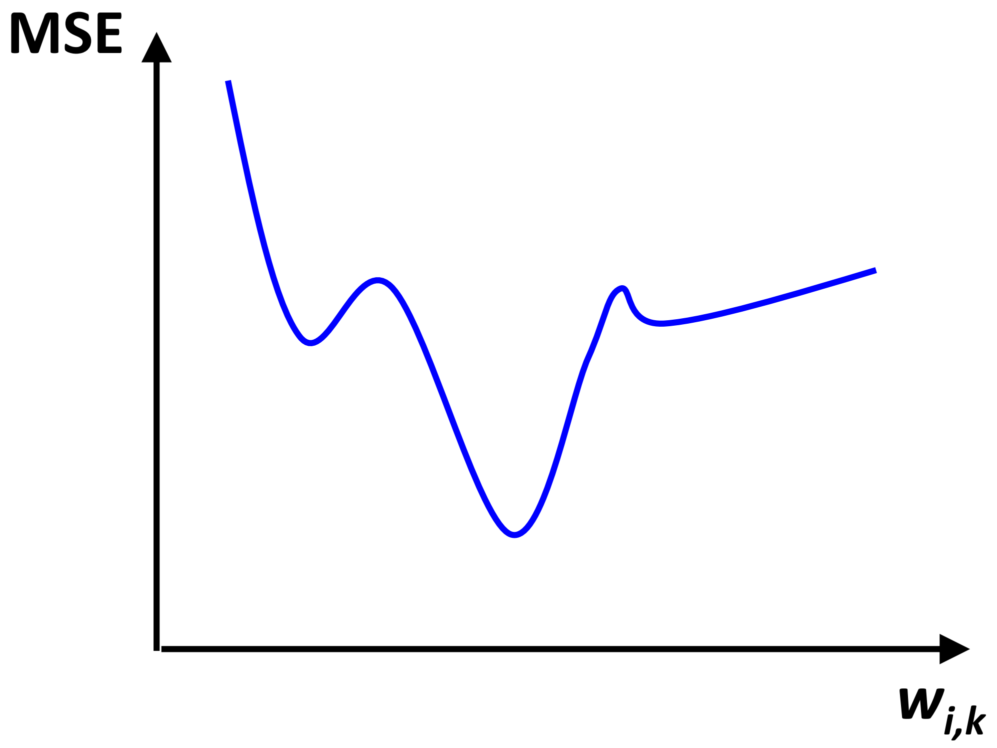
$$f(\lambda\boldsymbol{\theta} + (1-\lambda)\boldsymbol{\theta}') \leq \lambda f(\boldsymbol{\theta}) + (1-\lambda) f(\boldsymbol{\theta}')$$



Effect of Convexity

- We can always find the global optimal MLE





Gradient Descent Method vs. Stochastic Gradient Descent

- What if big data?

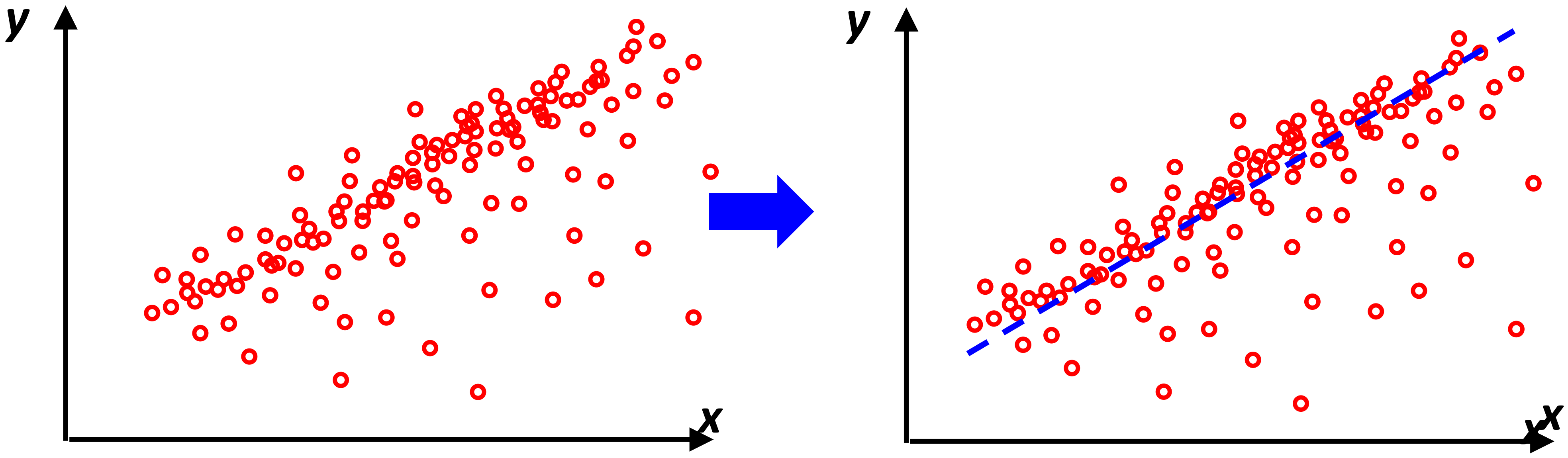
$$\mathrm{MSE}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \boldsymbol{w}^T\boldsymbol{x}_i)^2$$

$$w_{\text{next}} = w_{\text{present}} - \eta \nabla MSE(\boldsymbol{w})$$

- Stochastic Gradient Decent: Only using randomly picked **one** x_i per step
- Mini batch Gradient Decent

Applying MLE for D=1 Example w/ Considering Probability Density

- You should determine a probability density function that fits best to the given data. ➔ **We can do this by MLE. How?**
- Note that we are now apply MLE for the **training** phase, not **prediction** phase



Linear Regression w/ Gaussian Distribution Likelihood

- Linear regression is a model of form w/ Gaussian distribution,

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y|\mathbf{w}^T\mathbf{x}, \sigma^2)$$

- What we should determine is model or **θ for a given dataset** $\mathcal{D}$

p(θ/$\mathcal{D}$) vs. p($\mathcal{D}$/θ)?

- A common way to estimate the parameters of a statistical model **(=θ)** is to compute the **MLE (why?)**, which is defined as

$$\hat{\boldsymbol{\theta}} \triangleq \arg\max_{\boldsymbol{\theta}} \log p(\mathcal{D}|\boldsymbol{\theta})$$

- It is common to assume the training examples are independent and identically distributed, meaning log-likelihood is

$$\ell(\boldsymbol{\theta}) \triangleq \log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \boldsymbol{\theta})$$

Deriving MLE

- Inserting the definition of the Gaussian into the above, we find that the log likelihood is given by

$$\begin{aligned}\ell(\boldsymbol{\theta}) &= \sum_{i=1}^{N} \log\left[\left(\frac{1}{2\pi\sigma^2}\right)^{\frac{1}{2}} \exp\left(-\frac{1}{2\sigma^2}(y_i - \mathbf{w}^T\mathbf{x}_i)^2\right)\right] \\ &= \frac{-1}{2\sigma^2} RSS(\mathbf{w}) - \frac{N}{2}\log(2\pi\sigma^2)\end{aligned}$$

- This leads to the same results previously obtained as, which is also called as least squares

$$\mathbf{X}^T\mathbf{X}\mathbf{w} = \mathbf{X}^T\mathbf{y} \quad \Rightarrow \quad \hat{\mathbf{w}}_{OLS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

- **What is important is the overall procedure, not the result of this case.**

Focusing on Noise or Error Distribution

- RSS can be expressed if we define $\epsilon_i = (y_i - \mathbf{w}^T \mathbf{x}_i)$

$$\text{RSS}(\mathbf{w}) = ||\boldsymbol{\epsilon}||_2^2 = \sum_{i=1}^{N} \epsilon_i^2$$

- It can be said as the noise in regression models follow zero-mean Gaussian distribution with constant variance σ:

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2),$$

General Procedure For Training Is

Maximize **$p(\theta/D)$**

Revisit Minimizing MSE Approach for Comparing it w/ MLE or MAP

- **Minimizing MSE** is to minimize the following

$$\mathrm{MSE}(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \boldsymbol{w}^T\boldsymbol{x}_i)^2$$

θ change ➔ MSE change

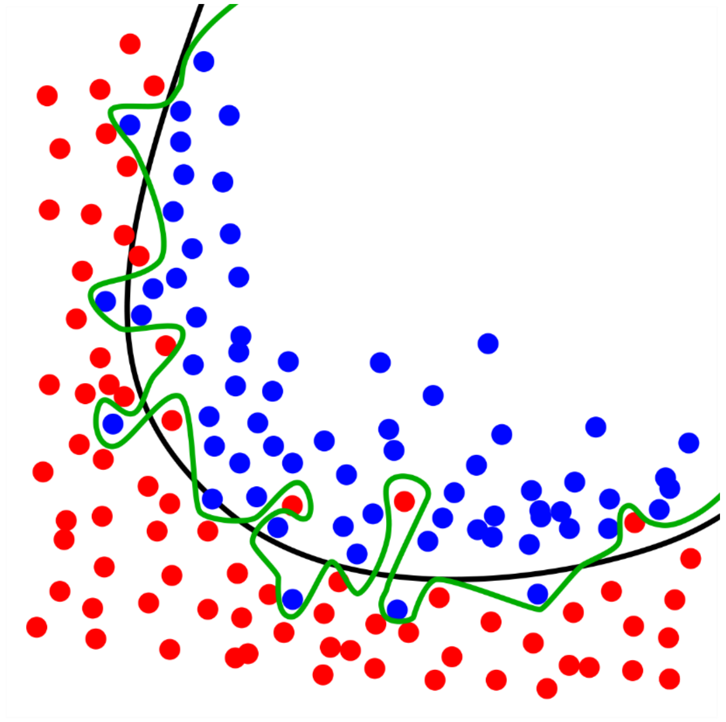
- MAP is to **maximize p(θ)** for given $\mathcal{D}$ and
  MLE is to **maximize p($\mathcal{D}$),** that is, to find θ that maximizes P($\mathcal{D}$):

θ change ➔ P($\mathcal{D}$) change

- For now, you can roughly say that Error↓ is equivalent to Probability ↑

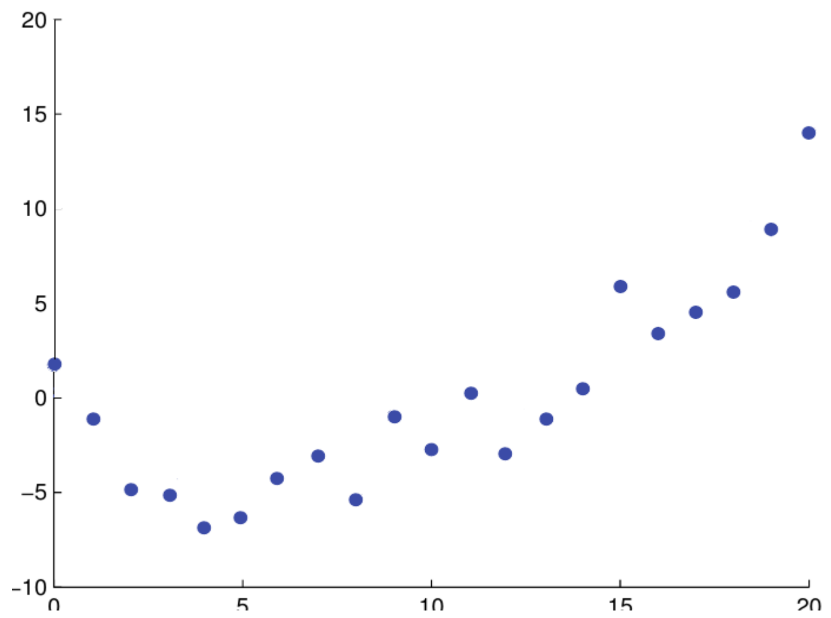
You Remember Overfitting?

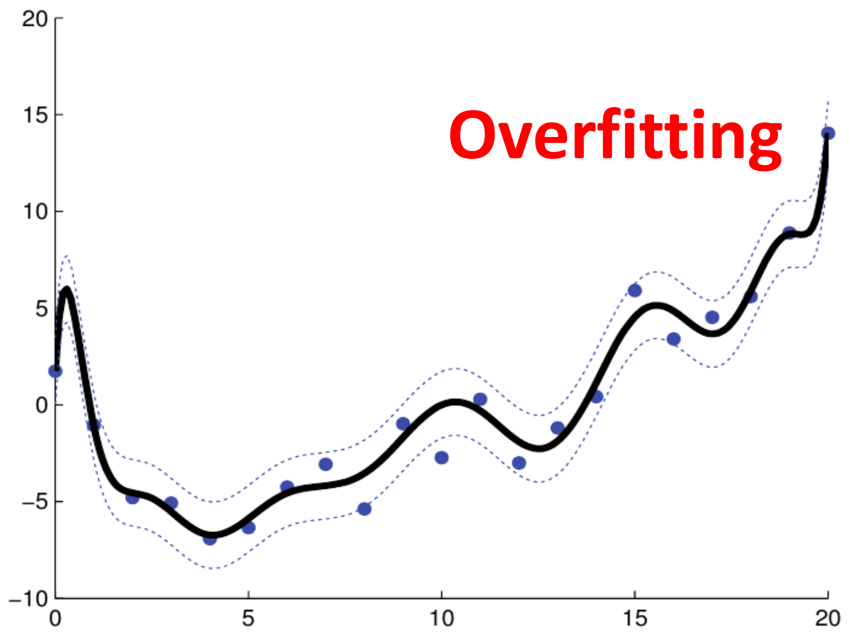
- It is the phenomenon that a predictive model (or ML) is too closely fit to only the training set.
- It is limitation of MLE, compared to MAP. You remember why?



Overfitting in Linear Regression
How Does the Fit Look Like?

- Suppose that you can use Degree 14 polynomial fit to the following N =21 data.





- Here, the resultant $\boldsymbol{w}_{\mathrm{OLS}}$ is

{1, 6.560, -36.934, -109.255, 543.452, 1022.561, -3046.224, -3768.013, 8524.540, 6607.897, -12640.058, -5530.188, 9479.730, 1774.639, -2821.526}

Revisit Posterior Probability in Bayesian (Generative) Learning

- Generative Classifier in posterior probability

$$\overset{\text{Posterior}}{p(y=c\mid \boldsymbol{x})} = \frac{p(y=c,\boldsymbol{x})}{p(x)} = \frac{p(\boldsymbol{x}|y=c)p(y=c)}{p(x)} \propto \overset{\text{Likelihood}}{p(\boldsymbol{x}|y=c)}\overset{\text{Prior}}{p(y=c)}$$

- Through **training**, we find out where the data originates from, or, the hidden data generator in the form of probability distrib.
- Given data D, we can find most probable $\boldsymbol{\theta}$, by maximizing

$$\overset{\text{Posterior}}{p(\boldsymbol{\theta}\mid D)} = \frac{p(\boldsymbol{\theta},D)}{p(D)} = \frac{p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(D)} \propto \overset{\text{Likelihood}}{p(D|\boldsymbol{\theta})} \times \overset{\text{Prior}}{p(\boldsymbol{\theta})}$$

- What we have done is to maximize p(D/θ), that is, MLE, or performed MAP with constant p(θ) Why?
  ➔ With abundant data, it is not problem.

Revisit Overfitting Issue in MLE

- Overfitting

- MLE is

$$\hat{\boldsymbol{\theta}} \triangleq \arg\max_{\boldsymbol{\theta}} \log p(\mathcal{D}|\boldsymbol{\theta})$$

- MAP is

Overfitting? We Can Adopt MAP!

- Posterior formulation in regression training:

$$\underbrace{p(\boldsymbol{w} \mid D)}_{\text{Posterior}} \propto \underbrace{p(D|\boldsymbol{w})}_{\text{Likelihood}} \times \underbrace{p(\boldsymbol{w})}_{\text{Prior}}$$

- So far, we assumed $p(\boldsymbol{w})$ is uniform, i.e., w follows uniform probability distribution.
- To reduce the “wiggle” in ***w***, we can assume the prior p(***w***) follows zero-mean Gaussian distribution as

$$p(\mathbf{w}) = \prod_j \mathcal{N}(w_j | 0, \tau^2)$$

- where $1/\tau^2$ controls the strength of prior

Performing MAP

- We should find w that maximizes $p(\boldsymbol{w} \mid D) \propto \; p(D|\boldsymbol{w}) \; \times$ $p(\boldsymbol{w})$ where

$p(D|\boldsymbol{w}) = \mathcal{N}(\boldsymbol{y}|\mathrm{w}_0 + \boldsymbol{wTx}, \sigma^2)$ and $p(\mathbf{w}) = \prod_j \mathcal{N}(w_j|0, \tau^2)$

- This leads to maximize log likelihood:

$$\underset{\mathbf{w}}{\operatorname{argmax}} \sum_{i=1}^{N} \log \mathcal{N}(y_i|w_0 + \mathbf{w}^T \mathbf{x}_i, \sigma^2) + \sum_{j=1}^{D} \log \mathcal{N}(w_j|0, \tau^2)$$

- That is equivalent to minimize

$$J(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - (w_0 + \mathbf{w}^T \mathbf{x}_i))^2 + \lambda ||\mathbf{w}||_2^2$$

where

$$\lambda \triangleq \sigma^2/\tau^2 \text{ and } ||\mathbf{w}||_2^2 = \sum_j w_j^2 = \mathbf{w}^T \mathbf{w}$$

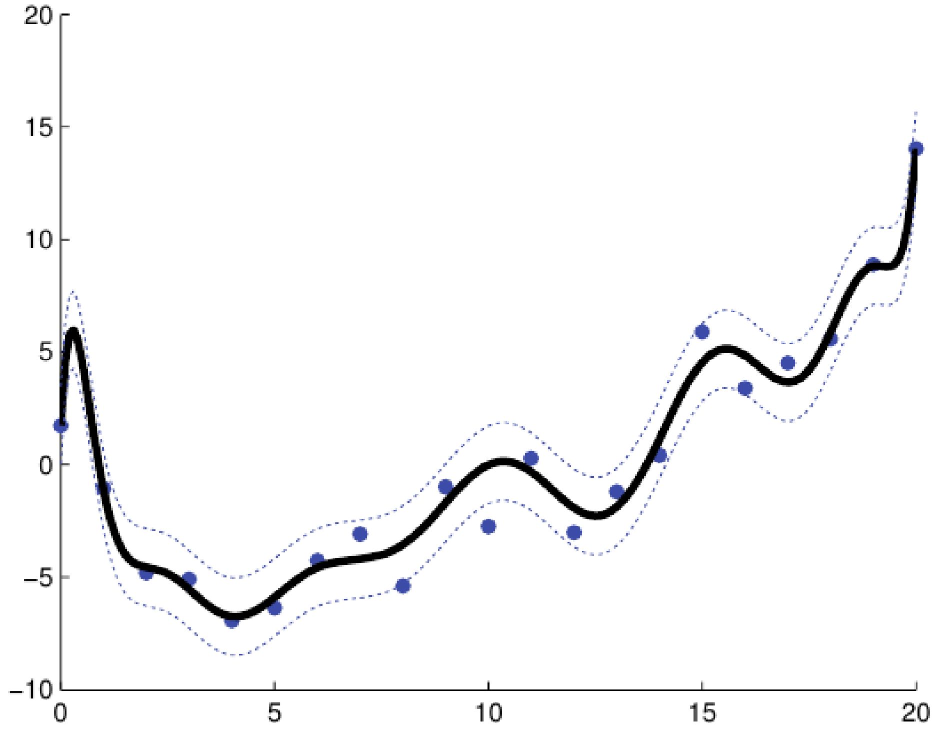
Optimal w with Regularization

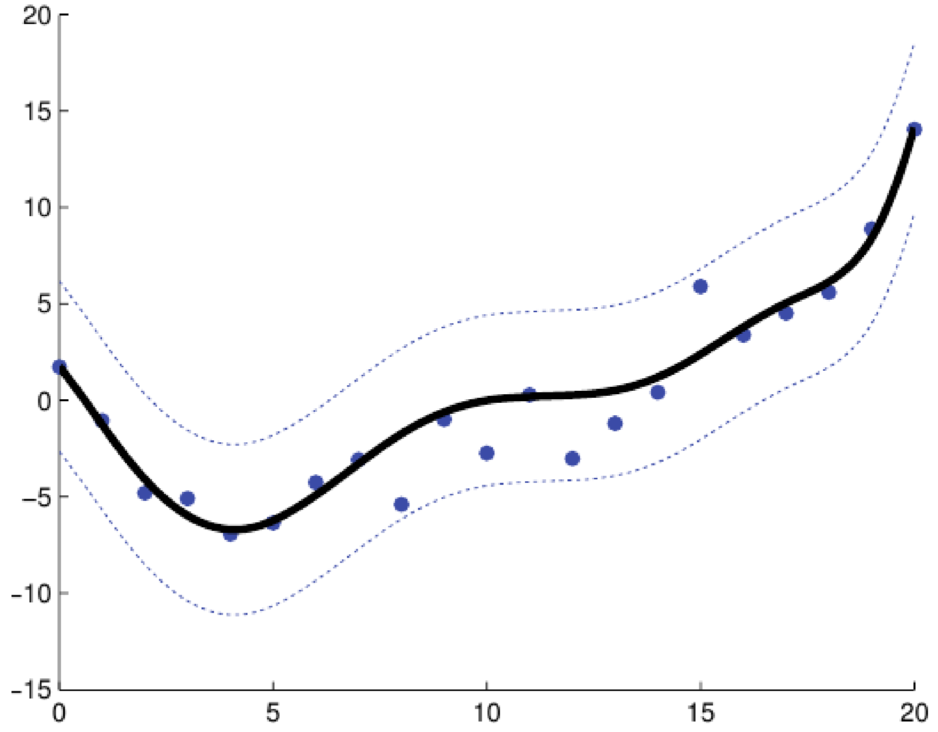
- The resultant optimal ***w*** is

$$\hat{\mathbf{w}}_{ridge} = (\lambda \mathbf{I}_D + \mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

- This technique is known **as ridge regression**, or **penalized least squares**.
- Adding a Gaussian prior to the parameters to encourage them to be small is called **regularization** or **weight decay**.
- Note that the offset term w_0 is not regularized, since this just affects the height of the function, not its complexity.
- By penalizing the sum of the magnitudes of the weights, we ensure the function is simple.

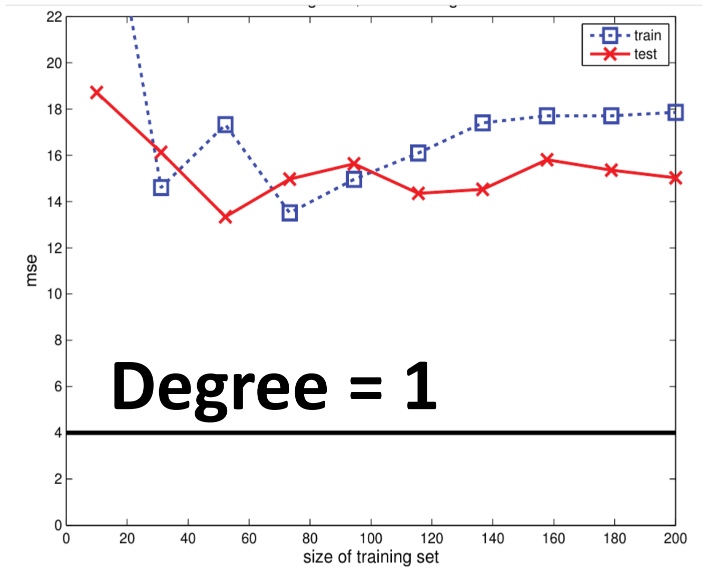
Results of Regularization

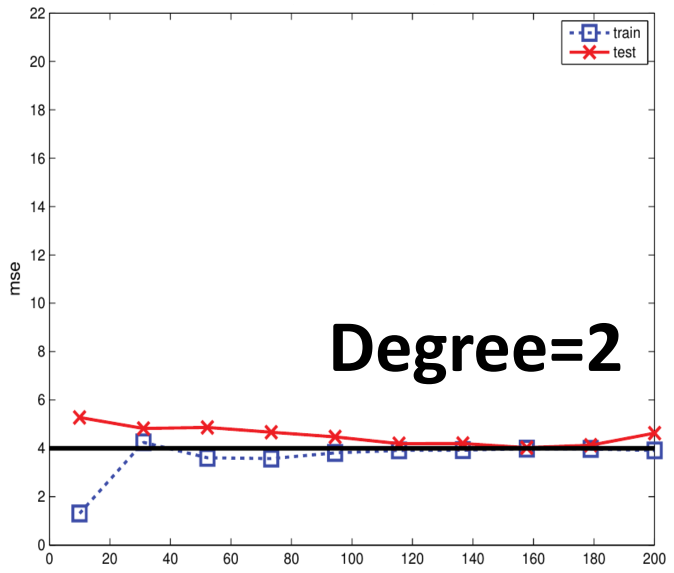


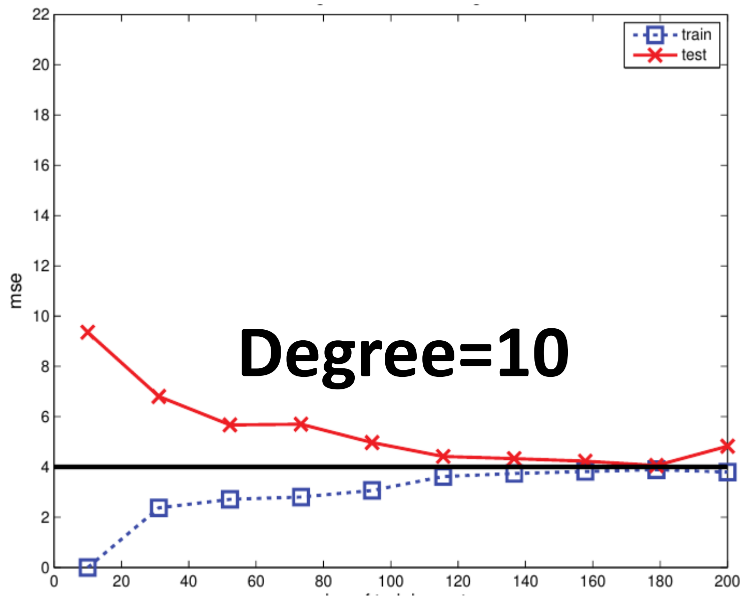


Regularization Effect of big data

- Regularization is the most common way to avoid overfitting. However, another effective approach — which is not always available — is to use lots of data ➔ Why?







Summary

- Linear regression through w/o probability density
- Linear regression through MLE with Gaussian likelihood
- Linear regression through MLE with Laplace likelihood handling outlier
- Linear regression through MAP with Gaussian likelihood & Gaussian prior