# 머신런닝 개요

# Lecture 5: ML Algorithms

College of Information and Electronic Engineering

Kyung Hee Univeristy

Prof. Wonha Kim

(wonha@khu.ac.kr)

# Contents

- Supervised Learning

1. K-nearest neighbor (k-NN)
2. Maximum A Posteriori (MAP) Estimation
3. Regression

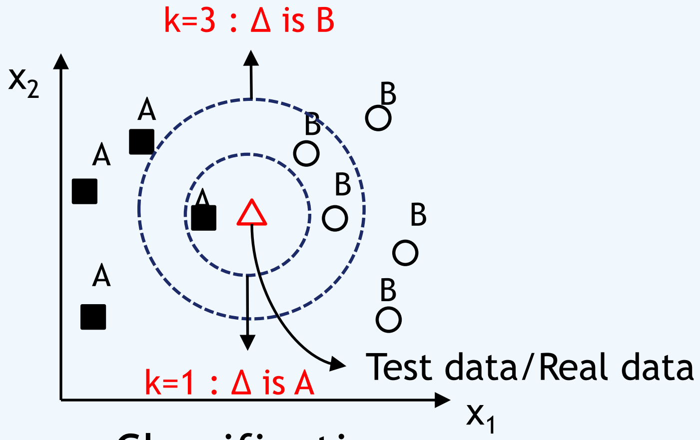
- Unsupervised Learning

4. K-mean

# 1. k-Nearest Neighbors(k-NN) (1/3) – Supervised Learning

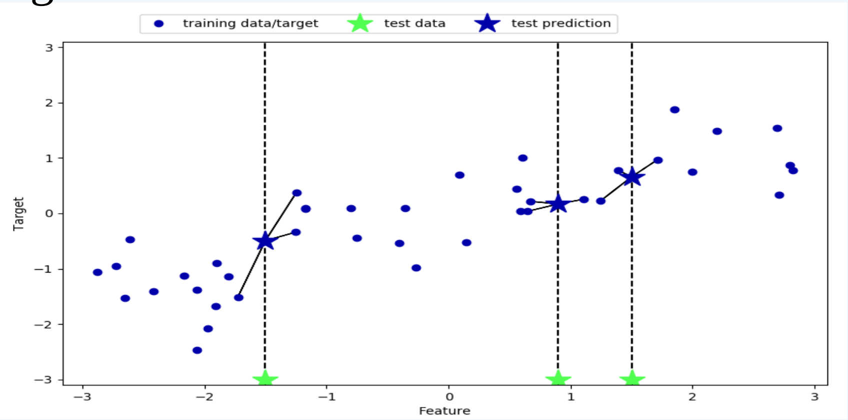
- Training set $D = \{(X_i, y_i)\}_{i=1}^{n}$, ($X_i$ : Feature vector, $y_i$ : Label/Target)
- Task : Assign the proper label to a query vector $X'$.
- Calculate distances from $X'$ to each training vector $X_i$ (i=1,…n): $d_i(X', X_i) = ||X' - X_i||$
- Select k nearest vectors. k should be odd to prevent ties.

- For classification, assign the majority label of k nearest vectors as the label of $X'$.




Classification

- For regression, assign average target value of k nearest vectors as the target value of $X'$.



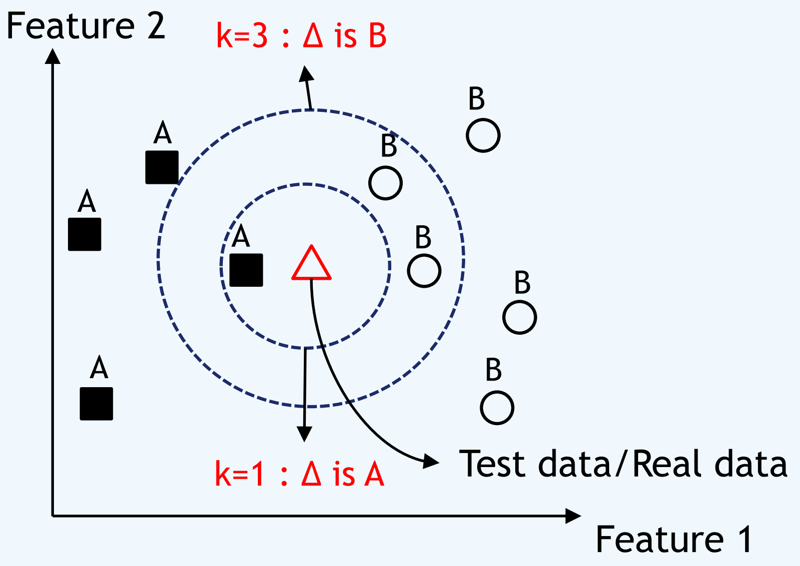

Regression

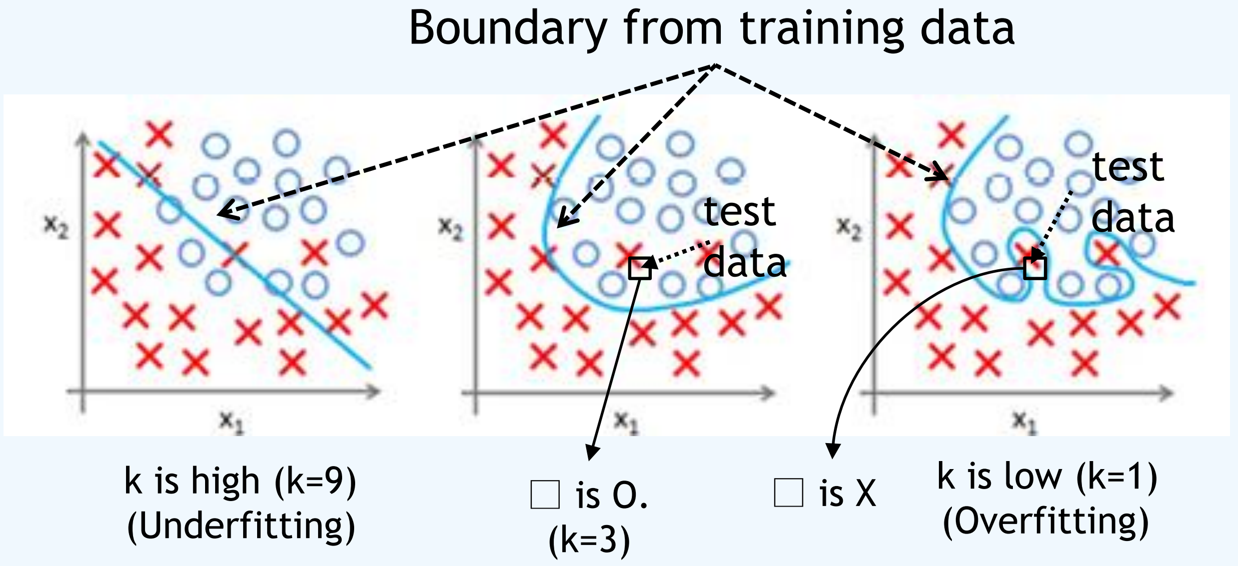
- Weight can be assigned as decreasing with distance so that nearer data contributes more.
  So, $d_i^{w}(X', X_i) = w \cdot ||X' - X_i||, w \propto 1/||X' - X_i||$

- Instance-based learning, Non-parametric method. (No use of any probability model)

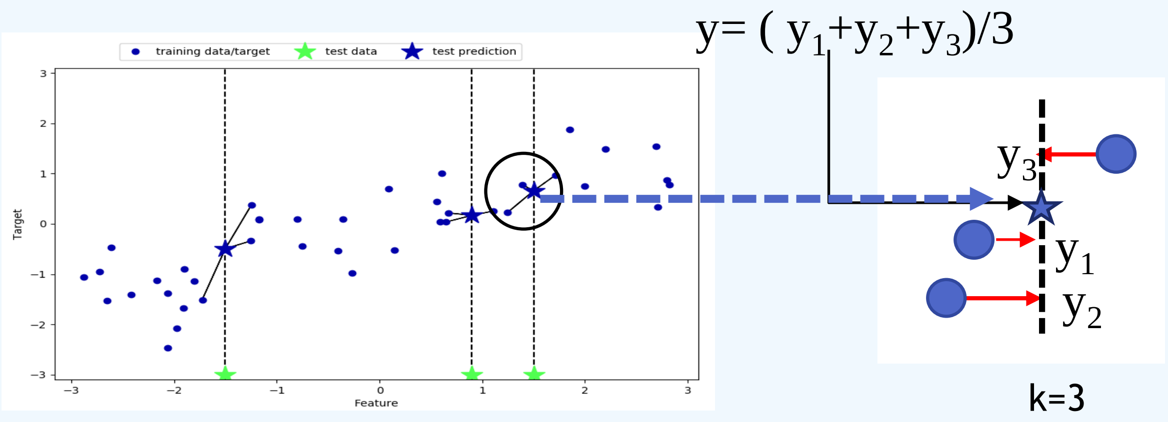
# 1. k-Nearest Neighbors (2/3) : Classification/Regression

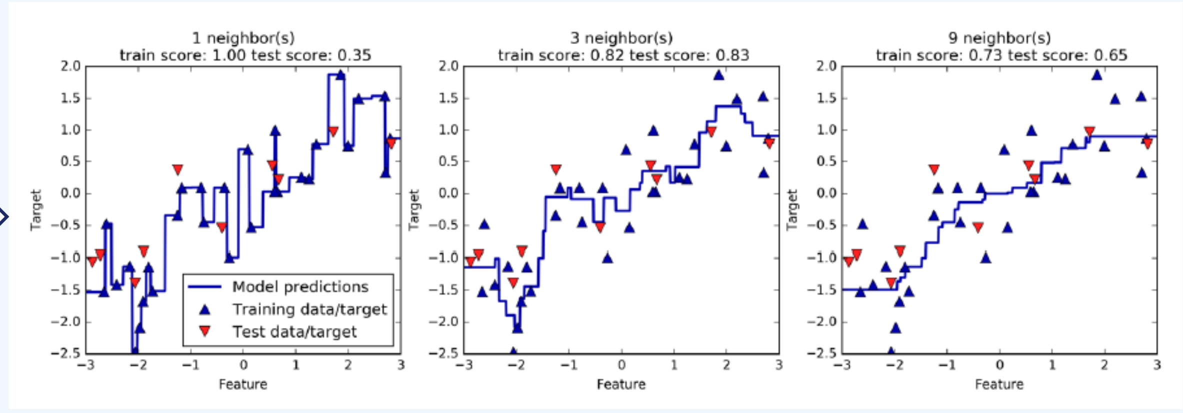
## Classification




## Regression




k=1, Overfitting

k=3

k=9, Underfitting

# 1. k-NN regression program (3/3)

```
import matplotlib.pyplot as plt
import mglearn
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor

# 데이터 셋을 만듭니다.
X, y = mglearn.datasets.make_wave(n_samples=40)

plt.plot(X, y, 'o')
plt.ylim(-3, 3)
plt.xlabel("Feature")
plt.ylabel("Target")
plt.show()

# k = 1 일 때의 예측
mglearn.plots.plot_knn_regression(n_neighbors=1)
plt.show()

# k = 3 일 때의 예측
mglearn.plots.plot_knn_regression(n_neighbors=3)
plt.show()

# wave 데이터 셋을 훈련 세트와 테스트 세트로 나눕니다.
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

# 이웃의 수를 3으로 하여 모델의 객체를 만듭니다.
reg = KNeighborsRegressor(n_neighbors=3)

# Training data와 Target을 사용하여 Model을 학습시킵니다.
reg.fit(X_train, y_train)

print("테스트 세트 예측:\n{}".format(reg.predict(X_test)))

print("테스트 세트 R^2: {:.2f}".format(reg.score(X_test, y_test)))
```

Cf) Introduction to Machine Learning with Python, p30~44

$$p(B|A)p(A) = p(A \cap B)$$
$$p(A|B)p(B) = p(A \cap B)$$
$$\Rightarrow p(A|B)p(B) = p(B|A)p(A) \Rightarrow p(A|B) = \frac{p(B|A)p(A)}{p(B)} = \frac{p(B|A)p(A)}{\sum_{\alpha \in A} p(B|A)p(A)}$$

$$P(y,x) = P(x|y)P(y) = P(x,y) = P(y|x)P(x) \Rightarrow P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

- $P(y|\mathbf{x}) = \frac{P(x|y)P(y)}{P(x)}$ $\Longleftrightarrow$ $\text{Posterior} = \frac{\text{Likelihood x Prior}}{\text{Evidence}}$

  – Posteriori Prob. P(y | x) 의 추정은 대부분의 경우 *x*의 공간 너무 크기 (너무 많기) 때문에 거의 불가능

  – Posteriori Prob. P(y | x)는 대부분은 Bayesian Theory를 이용하여 추정함.

  – Priori Prob. P(y)는 data로 부터 측정하여 구함. 즉, $P(y=i) = N_i/N$ = i의 개수/전체 샘플 개수

  – Likelihood P(**x**|y) ( L(y|**x**) ) 는 data의 observation를 통해서 구함. (Density Estimation method)

- MAP : x가 발생 했을 때, 가장 발생 가능한 y를 추정. $\Longleftrightarrow$ $y^* = \underset{y}{\operatorname{argmax}}\, P(y|\mathbf{x})$

# 2. Maximum A Posteriori (MAP) (2/9): Toy Example (1/2)

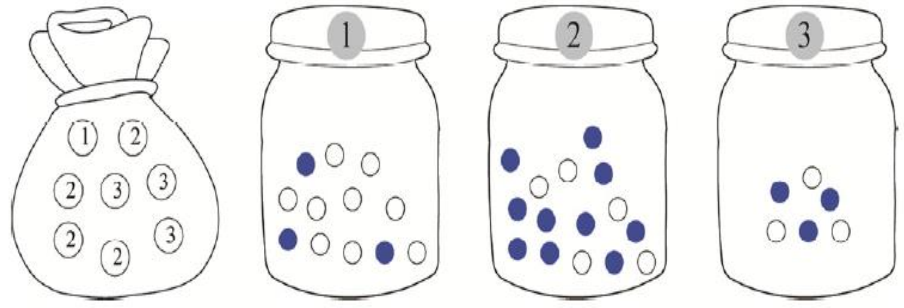


그림 2-15 확률 실험

-주머니에서 번호를 뽑은 다음, 번호에 따라 해당 병에서 공을 뽑고 색을 관찰함

- 꺼낸 카드와 공은 꺼낸 곳에 다시 넣음.

- 번호를 $y$, 공의 색을 $x$라는 확률변수로 표현하면 정의역은

  $y \in \{①, ②, ③\}$, $x \in \{파랑, 하양\}$

- ①번 카드를 뽑을 확률은 $P(y=①)=P(①)=1/8$

- 카드는 ①번, 공은 하양일 확률은 $P(y=①, x=하양)$

$$P(y = ①, x = 하양) = P(x = 하양 | y = ①)P(y = ①) = \frac{9}{12}\frac{1}{8} = \frac{3}{32}$$

- 하얀 공이 뽑힐 확률

$$P(하양) = P(하양|①)P(①) + P(하양|②)P(②) + P(하양|③)P(③)$$
$$= \frac{9}{12}\frac{1}{8} + \frac{5}{15}\frac{4}{8} + \frac{3}{6}\frac{3}{8} = \frac{43}{96}$$

- 합 규칙 (Marginalization)

$$P(x) = \sum_{y} P(y, x) = \sum_{y} P(x|y)P(y)$$

# 2. Maximum A Posteriori (MAP) (3/9): Toy Example (2/2)

- **하얀 공이 나왔을 때 [x=하양], 어느 병에서 나왔는지 추정하시오.** (**가장 발생 가능한** y=?)

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, P(y|x = \text{하양}) = \underset{y}{\operatorname{argmax}} \frac{P(x = \text{하양}|y)P(y)}{P(x = \text{하양})}$$

$$P(①|\text{하양}) = \frac{P(\text{하양}|①)P(①)}{P(\text{하양})} = \frac{\frac{9}{12}\frac{1}{8}}{\frac{43}{96}} = \frac{9}{43}$$

$$P(②|\text{하양}) = \frac{P(\text{하양}|②)P(②)}{P(\text{하양})} = \frac{\frac{5}{15}\frac{4}{8}}{\frac{43}{96}} = \frac{16}{43}$$

$$P(③|\text{하양}) = \frac{P(\text{하양}|③)P(③)}{P(\text{하양})} = \frac{\frac{3}{6}\frac{3}{8}}{\frac{43}{96}} = \frac{18}{43}$$

③번 병일 확률이 가장 높음

- **Bayesian rule의 Decision에 적용**

$$\overbrace{P(y|x)}^{\text{Posteriori Prob.}} = \frac{\overbrace{P(x|y)}^{\text{Likelihood}}\ \overbrace{P(y)}^{\text{Priori Prob.}}}{\underbrace{P(x)}_{\text{Evidence}}}$$

# 2. Maximum A Posteriori (MAP) (4/9) : Estimation

- Jang, Hana (A pro golfer)
  - **Evidence** : We *observe* that 1 of 10 players makes par at a hole. (P(x)): P(par) = 1/10.
  - **Prior** : Among 20 players, Jang hits one time. The prob. that Jan hits. (P(y)): P(Jang)=1/20.
  - **Likelihood** : Data shows that Jan Hana makes 6 pars at 7 hits. (P(x/y)) : P(par/Jang) = 6/7

- Prof. Kim, Wonha
  - **Evidence** : We *observe* that 1 of 10 players make par at a hole. : P(par) = 1/10.
  - **Prior** : Among 20 players, Prof. Kim played 7 times. The prob. of Kim's hitting. : P(Kim)=7/20.
  - **Likelihood** : *Data* shows that Kim makes the one par at 6 hits. : P(par/Kim) = 1/6.

- We know that either Kim or Jang makes par. Estimate who is he/she?
  - Prob. of Kim's hitting : P(Kim/Par) = P(Par/Kim) * P(Kim)/P(Par) =(1/6)*(7/20)/(1/10) = 7/12.
  - Prob. of Jang's hitting : P(Jang/Par) = P(Par/Jang) * P(Jang)/P(Par) =(6/7)*(1/20)/(1/10) = 6/14.

$\Rightarrow$ P(Par/Jang) > P(Par/Kim) : *Definitely, Jang plays much better than Prof. Kim.*

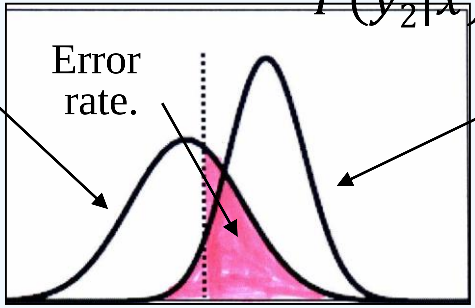
$\Rightarrow$ P(Kim) > P(Jang) : *Unfair. Prof. Kim plays 7 times more than Jang.*

$\Rightarrow$ P(Kim/Par) > P(Jang/Par) : *We must estimate that Prof. Kim makes the par.*

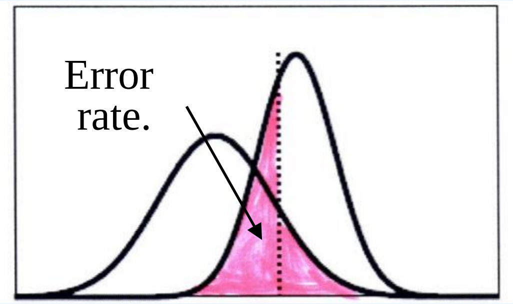
- Optimal Decision (Classification)

$$P(y_1|x) = \frac{P(x|y_1)P(y_1)}{P(x)}$$

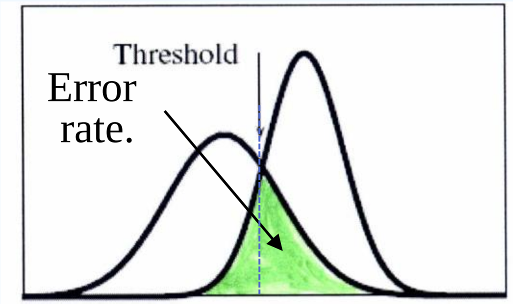
$$P(y_2|x) = \frac{P(x|y_2)P(y_2)}{P(x)}$$




Low threshold



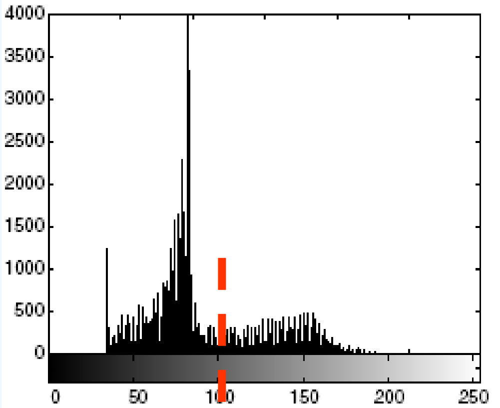

High threshold



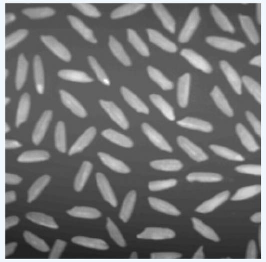

Optimal threshold

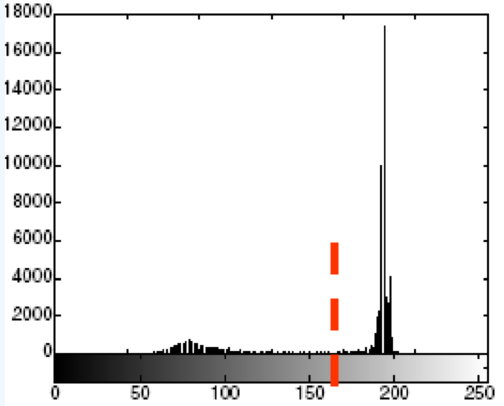
***Smallest Error rate***

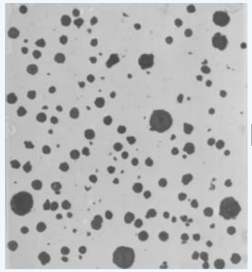
- EX: Segmentation
  - 두 물체의 밝기에 따른 확률 분포
  - 두 물체를 구분하는 최적의 밝기 threshold 설정하기

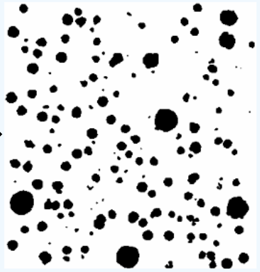



Rice image : rice.tif







# 2. MAP Estimation (6/9): Bayes Decision Rule (BDR)

- Bayesian Decision for two classes.
  - For observed data X, there are two classes, i.e, y=0, or y=1.
  - We decide y* in following way :

$$y^*(x) = \begin{cases} 0, & \text{if } P_{y|X}(0|X) \geq P_{y|X}(1|X) \\ 1, & \text{if } P_{y|X}(1|X) \geq P_{y|X}(0|X) \end{cases} \equiv \underset{y}{\operatorname{argmax}}\, P_{y|X}(y|X) = \operatorname{argmax} P_{X|y}(X|y)\; P_y(y)$$

- Bayesian Decision for n classes.
- For observed data X, there are n classes, i.e, y=1,…, n.
- $y^*=i$ if $P_{y|X}(y=i|X) > P_{y|X}(y=j|X)$ for $j \neq i$

$$\Longrightarrow\; y^* = \underset{y}{\operatorname{argmax}}\, P_{y|X}(y|X) = \underset{y}{\operatorname{argmax}}\, P_{X|y}(X|y)\; P_y(y)$$

# 2. MAP Estimation (7/9) :Decision

- Maximum Likelihood Estimation (MLE)

$$y^*(x) = \underset{y}{\operatorname{argmax}}\ P_{X|y}(x|y)$$

- Parameter w is known  and the optimal w should be founded .

- Maximum A Posteriori (MAP) Estimation .

$$y^*(x) = \underset{y}{\operatorname{argmax}}\ P_{y|X}(y|X)$$ : Not much used

$$y^*(x) = \underset{y}{\operatorname{argmax}}\ P_{X|y}(X|y)\ P_y(y)$$

$$y^*(x) = \underset{y}{\operatorname{argmax}}\ [\log(P_{X|y}(X|y)) + \log(P_y(y))]$$ : Most frequently used

- Data X are observed.
- Parameters y (or prior knowledge) are unknown and probabilistically presumed.
- If $P_y(w)$ is uniform, MLE =MAP.

- MAP for Multivariate Gaussian Classifier ( Model based learning)

 - The pdf of class i is a N-dimensional Gaussian with mean ui and covariance

$$P_{x|y}(X|y=i) = \frac{1}{\sqrt{(2\pi)^N|\Sigma_i|}} \exp\left\{-\frac{1}{2}(X-\mu_i)^T\Sigma_i^{-1}(X-\mu_i)\right\}$$

 -MAP Classifier

$$i^*(x) = \underset{i}{\operatorname{argmax}}\left[-\frac{1}{2}(X-\mu_i)^T\Sigma_i^{-1}(X-\mu_i) - \frac{1}{2}\log\{(2\pi)^N|\Sigma_i|\} + \log P_y(y=i)\right]$$

- Compare with K-NN.
  - K-NN : Non-model
  - Low complexity

- Design one dimensional Gaussian K-classifiers.

-Collect i$^{th}$ class dataset : $D^i = \{x_1^i, x_2^i, \ldots, x_n^i\}$ (data size for ith class is $n^i$.) where i=1,..., k.

-Estimate the Gaussian parameters:

$$\hat{u}_i = \frac{1}{n^i}\sum_{j=1}^{n^i} x_j^i, \quad \hat{\Sigma}_i = \frac{1}{n^i}\sum_{j=1}^{n^i}(x_j^i - \hat{u}_i)^2, \quad \hat{P}_w(y=i) = \frac{n^i}{S}$$ where S is the total sample size.

-MAP Classifier

$$i^*(x) = \underset{y}{\operatorname{argmax}}\left[-\frac{1}{2}(x-\hat{u}_i)^T\hat{\Sigma}_i^{-1}(x-\hat{u}_i) - \frac{1}{2}\log\{2\pi|\hat{\Sigma}_i|\} + \log\hat{P}_y(y=i)\right]$$

- Data x 가 주어졌을 때, x 를 발생 시킬 확률을 최대로 하는  매개변수 $\boldsymbol{\Theta}$ 를 추정하는 문제

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmax}}\, P(\mathbb{X}|\Theta)$$

- 수학적 계산 편의를 위해 Log 함수를 사용 할 수 있음.

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmax}}\, \log P(\mathbb{X}|\Theta)$$

- X = [ $\mathbf{x}_1, \mathbf{x}_2, ,,, , \mathbf{x}_n$]에서 $\mathbf{x}_i$ (i=1,,,,n) 가 independent and identical distribution (iid) 일 때 .

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmax}}\, \log P(\mathbb{X}|\Theta) = \underset{\Theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log P(\mathbf{x}_i|\Theta)$$

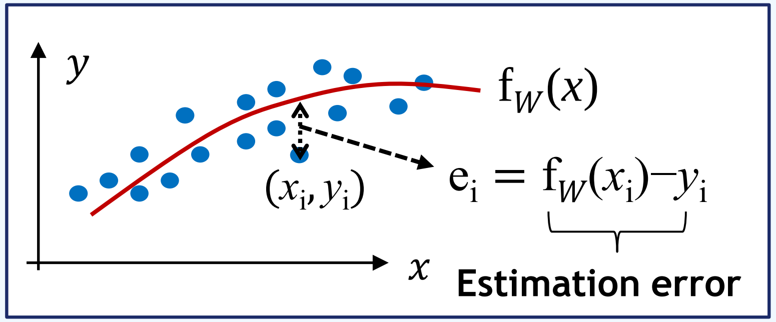
# 3 Machine Learning Model (1/7): Regression

- **Feature Data set : X** = $[x_1, x_2 \cdots x_n]^T$, **Target value set : Y** = $[y_1, y_2 \cdots y_n]^T$
- **Polynomial predicting the target value :** $\boldsymbol{f_w(x)}$ **=> For an input feature** $\boldsymbol{x_i}$ **,** $\boldsymbol{f_w(x_i)}$ **predicts its target get value** $y_i$**.**

$$\boldsymbol{f_W}(x_i) = w_0 + w_1 x_i + w_2 {x_i}^2 \cdots + w_p {x_i}^p = \sum_{\boldsymbol{k=0}}^{\boldsymbol{p}} w_k {x_i}^k = \mathbf{x}_\mathrm{i}{}^T \cdot \boldsymbol{W} \text{ where } \begin{array}{l} \mathbf{x}_\mathrm{i} = [1, x_i, {x_i}^2, \cdots, {x_i}^p]^T \\ \boldsymbol{W} = [w_0, w_1, w_2, \cdots, w_p] \end{array}$$

Bias

- Objective (Cost) function :




$$J(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} |e_i|^2 = \frac{1}{n}\sum_{i=1}^{n} (f_W(x_i) - y_i)^2$$

**(Mean Square Error (MSE))**

$$= (\mathbf{X_n} \cdot \boldsymbol{W} - \boldsymbol{Y})^T (\mathbf{X_n} \cdot W - \boldsymbol{Y})$$

$$\text{where } \mathbf{X_n} = \begin{bmatrix} \mathbf{x_1}^\mathrm{T} \\ \vdots \\ \mathbf{x}_{\boldsymbol{n}}^\mathrm{T} \end{bmatrix} = \begin{bmatrix} 1, & x_1 & {x_1}^2 & \cdots & {x_1}^p \\ & & \vdots & & \\ 1, & x_n & {x_n}^2 & \cdots & {x_n}^p \end{bmatrix}$$

- ML Training : Decide parameters so that the parameter set $\boldsymbol{W}$ minimizes MSE $J(\boldsymbol{W})$.

$$\widehat{\boldsymbol{W}} = \underset{\boldsymbol{W}}{\mathrm{argmin}}\ J(\boldsymbol{W})$$

$$J(\boldsymbol{W}) = (\mathbf{X_n} \cdot \boldsymbol{W} - \boldsymbol{Y})^\mathrm{T}(\mathbf{X_n} \cdot \boldsymbol{W} - \boldsymbol{Y}) = \boldsymbol{W}^\mathrm{T}\mathbf{X_n}^\mathrm{T}\mathbf{X}\boldsymbol{W} - 2(\mathbf{X_n}\boldsymbol{W})^\mathrm{T}\boldsymbol{Y} + \boldsymbol{Y}^\mathrm{T}\boldsymbol{Y}$$

$$\frac{1}{\partial \boldsymbol{W}} J(\boldsymbol{W}) = 2\mathbf{X_n}^T\mathbf{X_n}\boldsymbol{W} - 2\mathbf{X_n}^T\boldsymbol{Y} = \mathbf{0} \Rightarrow \widehat{\mathbf{W}} = (\mathbf{X_n}^\mathrm{T} \cdot \mathbf{X_n})^{-1} \cdot \mathbf{X_n}^\mathrm{T} \cdot \mathbf{Y}$$

- Feature vectors and target values are

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 2 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} 0 \\ 2 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

* Note that feature values are {1,1,2,2,3}.

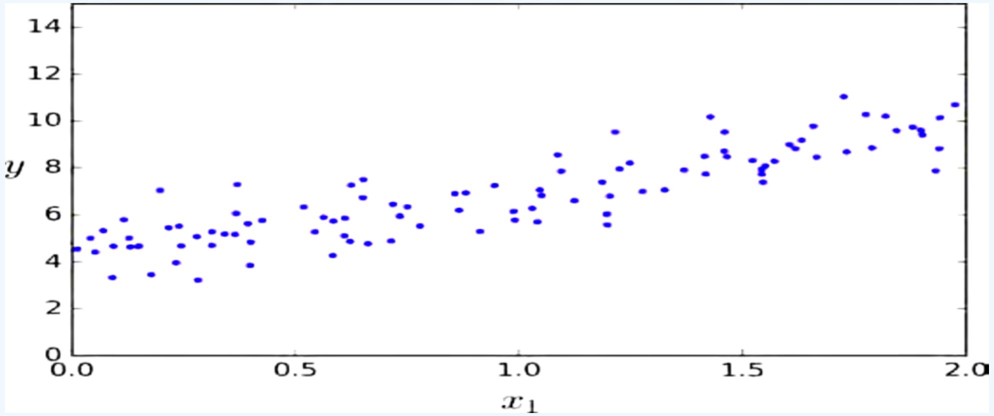
- Linear Regression Model is $y = b + a\,x$.
- Optimal weights.

$$[b\ a]^T = (X_n{}^T \cdot X_n)^{-1} \cdot X_n{}^T \cdot \boldsymbol{y} = [-0.5\ 1.5]^T$$

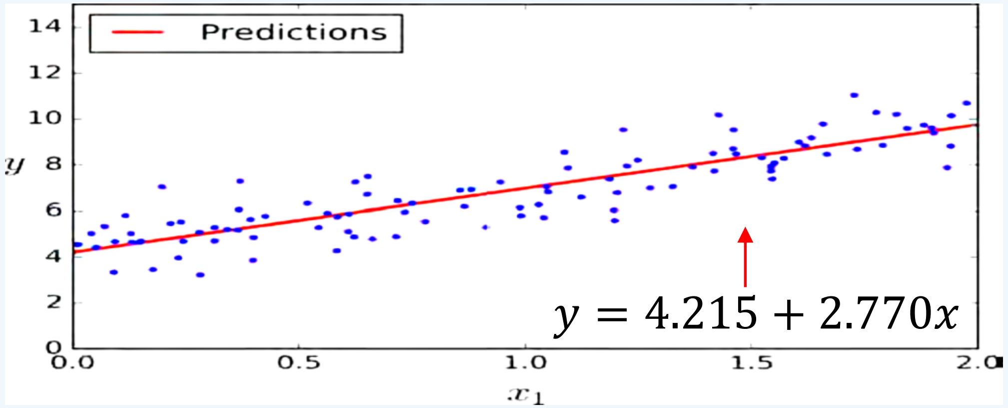
$$y = -0.5 + 1.5x$$

- Random data.

$$x = 2 \times \text{random\_noise}$$
$$y = 4 + 3 \times x + \text{random\_noise}$$




$$[b\ a]^T = (X_n{}^T \cdot X_n)^{-1} \cdot X_n{}^T \cdot \boldsymbol{y} = [4.215\ \ 2.770]^T$$




Oreilly, “Hands On Machine Learning with Scikit Learn and TensorFlow”, 108pp

- Multi dimension / Polynomial Regression Model
  - d-features: Training Set $\mathbb{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, $\mathbb{Y} = \{y_1, y_2, \cdots, y_n\}$ where $\mathbf{x}_i = (1, x_{i1}, x_{i2}, \cdots, x_{id})^{\mathrm{T}}$

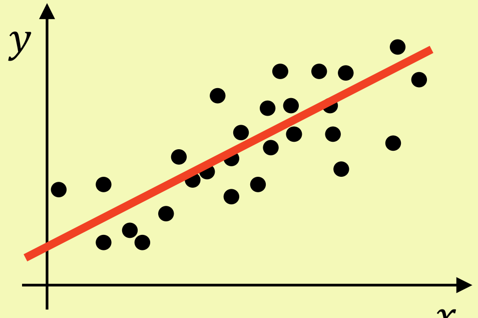
*Bias term*

$$\hat{y} = \theta_0 + \theta_1 x_{11} + \theta_2 x_{21}^2 + \cdots + \theta_p x_{n1}^p + \theta_{p+1} x_{12} + \cdots + \theta_{2p} x_{n2}^p + \cdots + \theta_{(d-1)p} x_{1d} + \cdots + \theta_{dp} x_{nd}^p$$

$$= \begin{bmatrix} 1 & x_{11} & x_{21}^2 & \cdots & x_{n1}^p & x_{21} & x_{21}^2 \cdots & x_{2n}^p \cdots & x_{1d} & \cdots & x_{nd}^p \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \vdots \\ \theta_{dp} \end{bmatrix} = \underline{J_\theta(\mathbf{x}) = \theta^T \cdot \mathbf{x}}$$

=> Optimal solution also is same with one-dimension model.

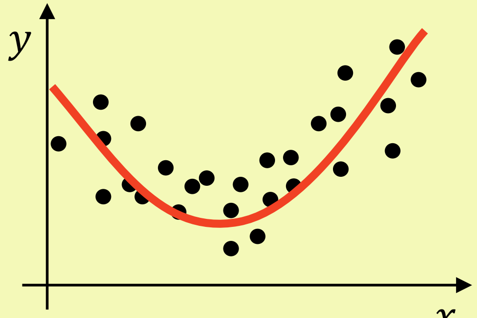
- Single feature /Linear




$\boldsymbol{\theta} = [\theta_0, \theta_1]$

$\mathbf{x} = [1, x_1]$

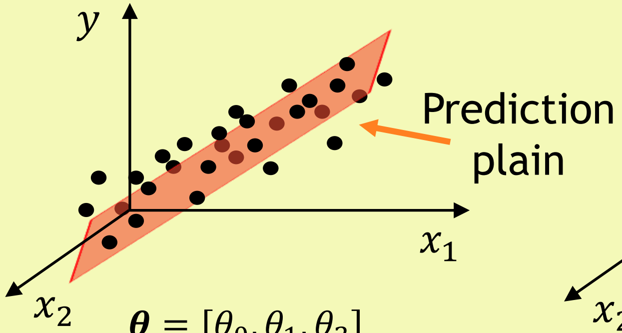
- Single feature /Non-Linear




$\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2]$

$\mathbf{x} = [1, x_1, x_1^2]$

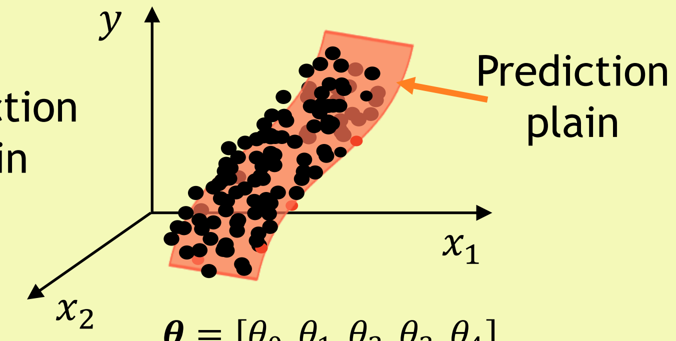
- Multiple features / Linear




$\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2]$

$\mathbf{x} = [1, x_1, x_2]$

- Multiple features / Non-Linear

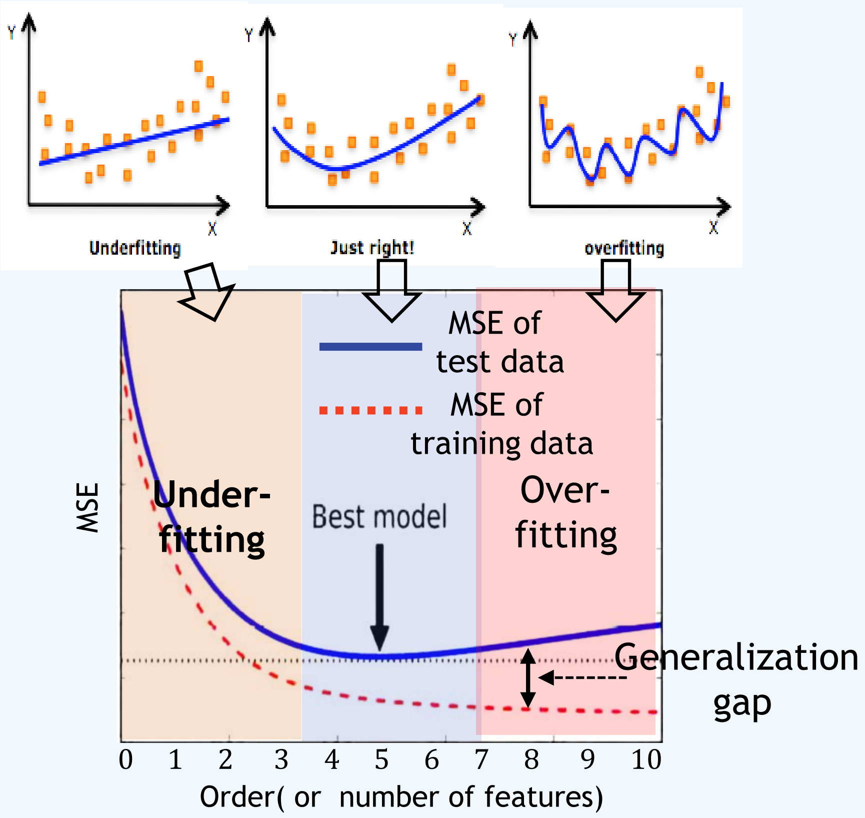



$\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2, \theta_3, \theta_4]$

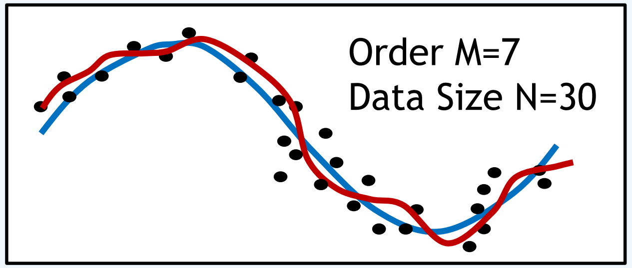
$\mathbf{x} = [1, x_1, x_1^2, x_2, x_2^2]$

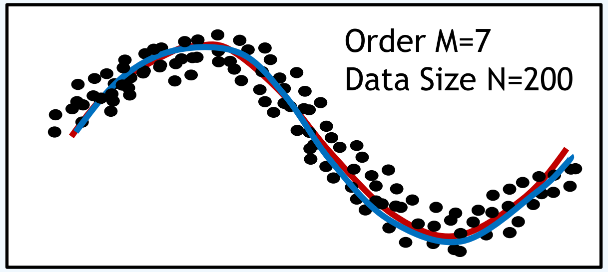
# 4. Regression(4/7) : Regularization

- Low order or Small features : Underfitting
  - A regression model with lower order ( or small features ) cannot fit to the training and test data.
  - Biased error
- Higher Order or Many features : Overfitting
  - A regression model with higher order ( or many features ) can reduce MSE for training data, but increase MSE for test data
  - Variance error




- The more data generally produce the more appropriate regression estimator.




However, increasing data is difficult in real applications

Regularization

# 4. Regression(5/7) : Regularization

- **Tikhonov Regularization**

$$\underbrace{J_{regularized}(\boldsymbol{\Theta}; \mathbb{X}, \mathbb{Y})}_{\text{규제를 적용한 목적함수}} = \underbrace{J(\boldsymbol{\Theta}; \mathbb{X}, \mathbb{Y})}_{\text{목적함수}} + \lambda \underbrace{R(\boldsymbol{\Theta})}_{\text{규제 항}}$$

- Smooth function : degree of inverse $\boldsymbol{\Theta}$ smooth
  - 차수가 높을 수로  커져서 penalty가 강화됨
  - 즉, 4 함수는 12차 함수 보다 smooth 정도가 적음
  - Data와 무관하고 model의 형태에 따라서 다름

- **L2 norm regulation :**

- For online learning

$$J_{regularized}(\boldsymbol{\Theta}; \mathbb{X}, \mathbb{Y}) = J(\boldsymbol{\Theta}; \mathbb{X}, \mathbb{Y}) + \lambda \|\boldsymbol{\Theta}\|_2^2$$

$$\nabla J_{regularized}(\boldsymbol{\Theta}; \mathbb{X}, \mathbb{Y}) = \nabla J(\boldsymbol{\Theta}; \mathbb{X}, \mathbb{Y}) + 2\lambda\boldsymbol{\Theta}$$

$$\begin{aligned}\boldsymbol{\Theta} &= \boldsymbol{\Theta} - \rho \nabla J_{regularized}(\boldsymbol{\Theta}; \mathbb{X}, \mathbb{Y}) \\ &= \boldsymbol{\Theta} - \rho(\nabla J(\boldsymbol{\Theta}; \mathbb{X}, \mathbb{Y}) + 2\lambda\boldsymbol{\Theta}) \\ &= (1 - 2\rho\lambda)\boldsymbol{\Theta} - \rho\nabla J(\boldsymbol{\Theta}; \mathbb{X}, \mathbb{Y})\end{aligned}$$

$$\Downarrow$$

$$\boldsymbol{\Theta} = (1 - 2\rho\lambda)\boldsymbol{\Theta} - \rho\nabla J$$

- For batch learning

$$J_{regularized}(\mathbf{w}) = (\mathbf{Xw} - \mathbf{y})^{\mathrm{T}}(\mathbf{Xw} - \mathbf{y}) + \lambda\|\mathbf{w}\|_2^2$$

ox문제 나옴
도중에 추가하면 절대 안됨

$$\frac{\partial J_{regularized}}{\partial \mathbf{w}} = \mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{w} - \mathbf{X}^{\mathrm{T}}\mathbf{y} + 2\lambda\mathbf{w} = \mathbf{0}$$

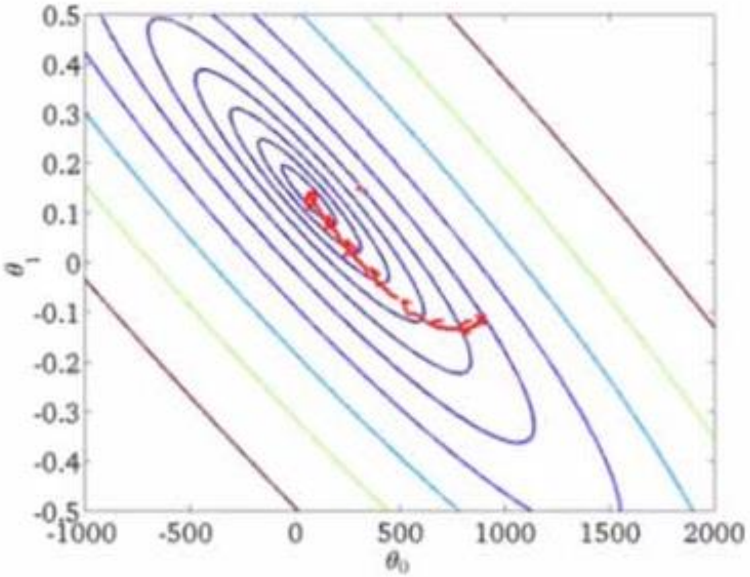
$$\Rightarrow \quad (\mathbf{X}^{\mathrm{T}}\mathbf{X} + 2\lambda\mathbf{I})\mathbf{w} = \mathbf{X}^{\mathrm{T}}\mathbf{y}$$

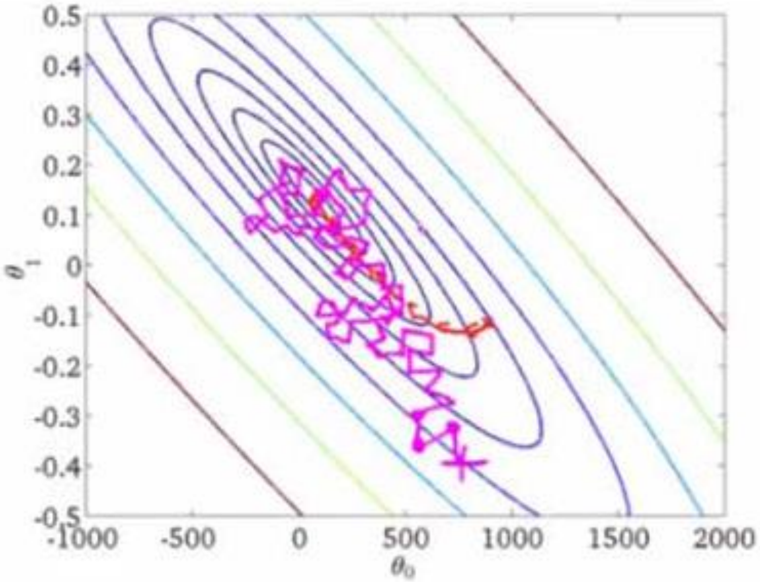
$$\Downarrow$$

$$\hat{\mathbf{w}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + 2\lambda\mathbf{I})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

# 4. Regression (6/7): On-line vs Batch regression

| Batch Gradient Descent | Stochastic Gradient Descent |
|---|---|
| • Gently converges to the (local) minimum<br>• Very slow<br>• Intractable for datasets that don't fit in memory<br>• No online learning | • Faster<br>• Online learning<br>• Heavy fluctuation<br>• Capability to jump to new (potentially better local minima)<br>• Complicated convergence (overshooting) |






From Kwangil Kim's lecture note

**예제 5-1** **리지 회귀**

훈련집합 $\mathbb{X} = \{\mathbf{x}_1 = \begin{pmatrix}1\\1\end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix}2\\3\end{pmatrix}, \mathbf{x}_3 = \begin{pmatrix}3\\3\end{pmatrix}\}$, $\mathbb{Y} = \{y_1 = 3.0,\ y_2 = 7.0, y_3 = 8.8\}$이 주어졌다고 가정하자. 특징 벡터가 2차원이므로 $d$=2이고 샘플이 3개이므로 $n$=3이다. 훈련집합으로 설계행렬 **X**와 레이블 행렬 **y**를 다음과 같이 쓸 수 있다.

$$\mathbf{X} = \begin{pmatrix}1 & 1\\2 & 3\\3 & 3\end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix}3.0\\7.0\\8.8\end{pmatrix}$$

이 값들을 식 (5.29)에 대입하여 다음과 같이 $\hat{\mathbf{w}}$을 구할 수 있다. 이때 $\lambda = 0.25$라 가정하자.

$$\hat{\mathbf{w}} = \left(\begin{pmatrix}1 & 2 & 3\\1 & 3 & 3\end{pmatrix}\begin{pmatrix}1 & 1\\2 & 3\\3 & 3\end{pmatrix} + \begin{pmatrix}0.5 & 0\\0 & 0.5\end{pmatrix}\right)^{-1}\begin{pmatrix}1 & 2 & 3\\1 & 3 & 3\end{pmatrix}\begin{pmatrix}3.0\\7.0\\8.8\end{pmatrix} = \begin{pmatrix}1.4916\\1.3607\end{pmatrix}$$

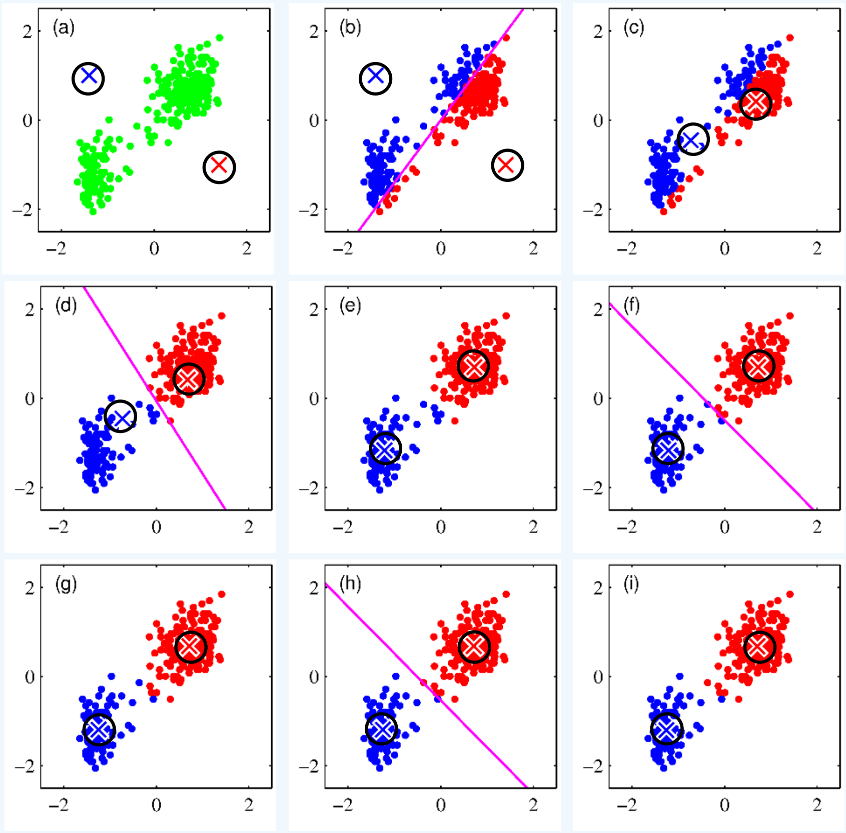
따라서 하이퍼 평면은 $y = 1.4916x_1 + 1.3607x_2$이다. 새로운 샘플로 $\mathbf{x} = (5 \quad 4)^{\mathrm{T}}$가 입력되면 식 (5.30)을 이용하여 12.9009를 예측한다.

- For d-features (dimension) and n- observation (or data) set $[\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$, partition the n observation into $\mathrm{k}(\leq n)$ sets $\mathbf{S} = \{S_1, S_2, \cdots, S_k\}$ so as to minimize the within-cluster sum of squares(WCSS) (Wikipedia)

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

$\boldsymbol{\mu}_i$ : is the mean of elements in $S_i$.

K(=2)-mean



- Algorithms

: Give an initial set of $k$ means $\boldsymbol{u}_1^{(1)}, \cdots, \boldsymbol{u}_k^{(1)}$

: Do iteration until $||\boldsymbol{u}_i^{(t+1)} - \boldsymbol{u}_i^{(t)}|| > \varepsilon$

(1) Assign each observation to the cluster whose mean has the least distance.

$$i^* = \underset{1 \leq i \leq k}{\mathrm{argmin}} \, ||\mathbf{x}_j - u_i|| \Rightarrow \mathbf{x}_j \to S_{i^*} \quad (j = 1, \ldots, N)$$

(2) Update the means to be the centroids of observations in the new clusters

$$u_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \mathbf{x}_j$$

$\left|S_i^{(t)}\right|$ is the number of data in class $S_i^{(t)}$

# 4. K-mean Clustering (2/3) :  Constructing clusters

- Mean Splitting (for deciding proper k).

1. Set k=1, calculate $u^{(i)}$.
2. Splitting the mean for k=2  by perturbing randomly.

$$\mathbf{u}_1^{(2)} \leftarrow \mathbf{u}^{(1)}, \quad \mathbf{u}_2^{(2)} \leftarrow (1+\epsilon)\mathbf{u}^{(1)} \qquad (\epsilon <<1)$$

3. Run k=2 means.

4. Deleting empty clusters

  - Check the number of elements in each cluster.
  - Delete the cluster having too small elements
  - Split the cluster having the most elements.

4. Splitting the means for k=4.

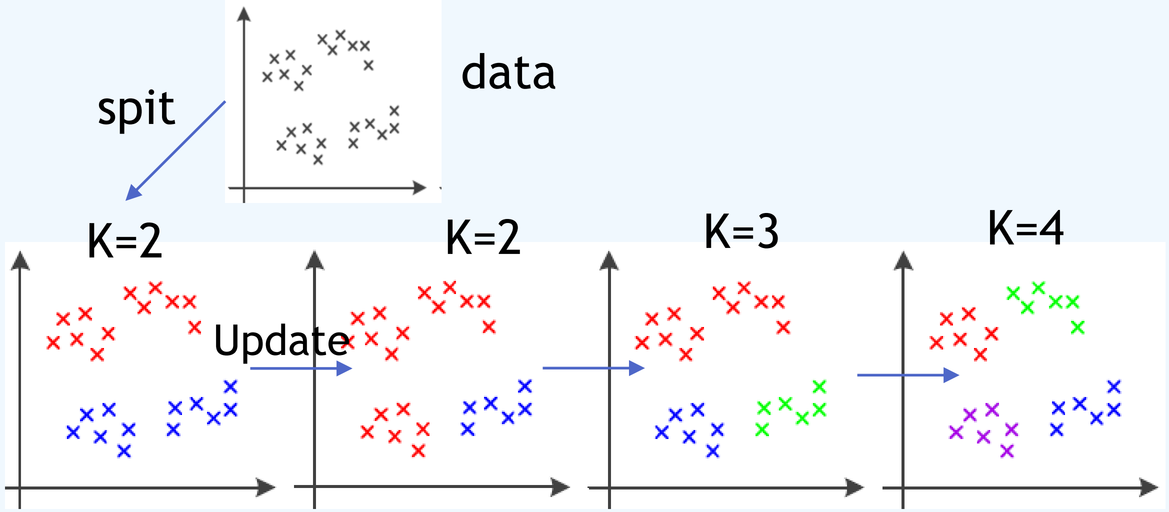
$$\mathbf{u}_1^{(4)} \leftarrow \mathbf{u}_1^{(2)}, \quad \mathbf{u}_2^{(4)} \leftarrow (1+\epsilon)\mathbf{u}_1^{(2)}$$

$$\mathbf{u}_3^{(4)} \leftarrow \mathbf{u}_2^{(2)}, \quad \mathbf{u}_4^{(4)} \leftarrow (1+\epsilon)\mathbf{u}_2^{(2)}$$

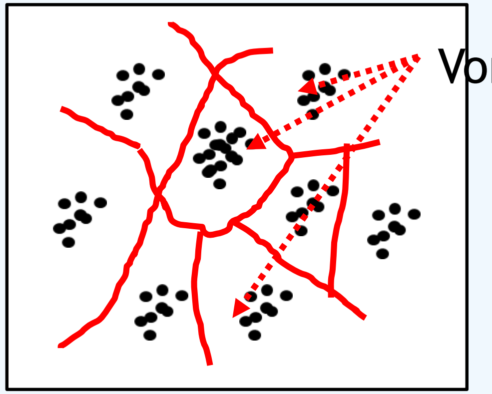
5. Repeat splitting until finding the proper k.

  (* Random initialization of means are not good.)

- Clustering with k=4




- Voronoi Cell



Voronoi Cells: $\mathbf{S} = \{S_1, S_2, \cdots, S_k\}$

# 4. K-mean Clustering (3/3) : Application

- Image segmentation
  - The Image Data form is I(x,y) =[R, G, B].
  - K-mean clustering let neighbored and similar color pixels be a set.