Machine Learning

Classification Model

Present by Sangmin Bae

Contents

Part One

Logistic Regression

(Remind) Regression vs. Classification

회귀 (Regression)

1. 입력값 : 연속값(실수형), 이산값(범주형) 등 모두 가능
2. 출력값 : **연속값(실수형)**
3. 모델 형태 : 일반적인 함수 형태 (eg. $y = w_1 x + w_0$)

분류 (Classification)

1. 입력값 : 연속값(실수형), 이산값(범주형) 등 모두 가능
2. 출력값 : **이산값(범주형)**
3. 모델 형태 : 이진 분류라면 **시그모이드(sigmoid)** 함수, 다중 분류라면 **소프트맥스(softmax)** 함수 꼭 포함

Why not Linear Regression?

선형 회귀를 통한 분류

- $$\mathbf{Y} = \begin{cases} \mathbf{1\ if\ korean;} \\ \mathbf{2\ if\ american;} \\ \mathbf{3\ if\ japanese;} \end{cases} \qquad \mathbf{Y} = \begin{cases} \mathbf{1\ if\ american;} \\ \mathbf{2\ if\ korean;} \\ \mathbf{3\ if\ japanese;} \end{cases}$$

- 일반적인 회귀에서는 **라벨의 순서(크기)에 따라 결과가 달라짐**
- 다른 손실 함수나 모델이 필요함

(Remind) Sigmoid Function

시그모이드 함수 (sigmoid function)

- $\mathbf{y} = \frac{\mathbf{1}}{\mathbf{1+e^{-x}}}$
- $\mathrm{x} = 0$ 일때, 함수의 출력값은 0.5가 됨
- 함수의 출력값이 항상 0 이상 1 이하임

Logistic Regression (1)

오즈 (odds)

- **성공($y = 1$) 확률이 실패($y = 0$) 확률에 비해 몇 배 더 높은가**를 나타냄
- $\mathbf{odds} = \frac{\mathbf{p(y=1|x)}}{\mathbf{1-p(y=1|x)}}$

로짓 변환 (logit)

- 오즈에 로그를 취한 함수 형태
- 입력값 (p)의 범위가 [0,1] 일 때, $[-\infty, +\infty]$ 를 출력함
- $\mathbf{logit(p) = log(odds) = log\,\frac{p(y=1|x)}{1-p(y=1|x)}}$

Logistic Regression (2)

로지스틱 함수 (logistic function)

- **로짓 변환의 역함수**로 해석 가능
- $\mathrm{logit}(p) = \log(\mathrm{odds}) = \log\frac{p(y=1|x)}{1-p(y=1|x)} = w_0 + w_1x_1 + \cdots + w_Dx_D = w^TX$
- $\mathbf{p(y = 1|x) = \frac{e^{w^Tx}}{1+e^{w^Tx}} = \frac{1}{1+e^{-w^Tx}}}$
- 따라서 로지스틱 함수는 **선형 회귀와 sigmoid 함수의 결합임**

Logistic Regression (3)

로지스틱 회귀 (logistic regression)

- 로지스틱 회귀 모델은 **로지스틱 함수 형태의 회귀 모델임**
- $\mathbf{P(\hat{y} = 1|X) = \frac{1}{1+e^{-w^TX}}}$
- wX 의 값에 따라 예측값이 달라짐
  1. $\mathbf{w^TX > 0}$: 1로 분류
  2. $\mathbf{w^TX < 0}$: 0으로 분류
- 파라미터 w 의 최적값을 찾기 위한 **손실 함수**를 어떻게 정의할까?

Bayes' Theorem

베이즈 정리 (bayes' theorem)

- $\mathbf{P(w|X)} = \frac{\mathbf{P(X|w)P(w)}}{\mathbf{P(X)}} \propto \mathbf{P(X|w)P(w)}$
- **사후 확률** (posterior, $P(w|X)$) : 데이터가 주어졌을 때 가설에 대한 확률 분포(신뢰도)
- **우도 확률** (likelihood, $P(X|w)$) : 가설을 잘 모르지만 안다고 가정했을 경우, 주어진 데이터의 분포
- **사전 확률** (prior, $P(w)$) : 데이터를 보기 전, 일반적으로 알고 있는 가설의 확률
- 이 확률들을 통해 가설(모델의 파라미터)를 추정하는 방법으로 **MLE**와 **MAP** 두 가지가 있음

Maximum Likelihood Estimation (1)

우도 확률 (likelihood, $P(X|w)$)

- 모델 파라미터 값을 잘 모르지만 안다고 가정했을 경우, 주어진 데이터의 분포
- 따라서 우도 확률은 **모델의 파라미터 (w)에 대한 함수로 데이터의 분포를 표현**함
- 각 샘플이 i.i.d (independent and identical distributed)하다고 가정 후, 흔히 아는 **PDF (probability density function)의 곱으로 표현됨**
- Ex. 정규 분포를 따르는 데이터에 대한 우도 확률
  - w : μ (평균), σ (분산)
  - PDF : $\frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$
  - Likelihood : $\prod_i \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}\left(\frac{x_i-\mu}{\sigma}\right)^2}$

Maximum Likelihood Estimation (2)

최대 우도 추정법 (Maximum Likelihood Estimation, MLE)

- 현재의 데이터 분포가 나올 확률이 가장 높은 파라미터 == **우도 확률을 최대로 만드는 파라미터**
- $\hat{\mathbf{w}} = \underset{\mathbf{w}}{\arg\max}\, \mathbf{P}(\mathbf{X}|\mathbf{w})$
- 매우 간단한 파라미터 추정법이지만, **데이터에 따라 값이 민감하게 변화함**
- Ex. 앞면만 나온 동전 던지기

Maximum A Posterior

최대 사후 확률 (Maximum A Posterior, MAP)

- **데이터에 의존적인 MLE의 단점을 해결**하기 위해 사용되는 방법론
- 주어진 데이터에 대해 최대 확률을 가지는 파라미터를 찾는 방법
- $\hat{\mathbf{w}} = \underset{\mathbf{w}}{\arg\max}\, \mathrm{P}(\mathbf{w}|\mathbf{X})$
- 하지만, 사후 확률은 곧바로 계산이 불가능
- 베이즈 정리를 이용해 **사전 확률과 우도 확률의 곱으로 표현**
- 추정의 정확도는 **사전 확률의 정확도에 좌우됨**

MLE for Logistic Regression (1)

베르누이 분포 (Bernoulli)

- 베르누이 시행이란 **두 가지 결과값**만을 가지는 실험을 지칭
- 베르누이 시행에 따라 **0(실패) 또는 1(성공)**의 값을 대응시키는 확률변수를 베르누이 확률변수라 부름
- 이 확률변수의 분포를 베르누이 분포라 명명
- $\mathbf{P(Y = y_i) = p^{y_i}(1 - p)^{1-y_i}}$ (**파라미터는 p 하나**)
- $\mathbf{L = \prod_i p^{y_i}(1 - p)^{1-y_i}}$

MLE for Logistic Regression (2)

(Remind) 로지스틱 회귀 (logistic regression)

- 로지스틱 회귀 모델은 로지스틱 함수 형태의 회귀 모델임
- $\mathbf{P(\hat{y} = 1|X) = \frac{1}{1+e^{-w^T X}} = \sigma(w^T X)}$
- **파라미터 p 값이 $\sigma(w^T X)$ 인 베르누이 분포**로 해석 가능

로지스틱 회귀의 우도 함수

- $\mathbf{L = \prod_i \sigma(w^T X_i)^{y_i} \left(1 - \sigma(w^T X_i)\right)^{1-y_i}}$
- 로그 함수는 단조 증가 함수이므로 L 또는 **ln L** 를 최대로 만드는 w 는 동일함
- $\ln L = \sum_i y_i \ln\{\sigma(w^T X_i)\} + \sum_i (1 - y_i) \ln\{1 - \sigma(w^T X_i)\}$
- 로그 우도 함수를 최대화 == **-로그 우도 함수를 최소화**

MLE for Logistic Regression (3)

SGD for MLE

- 손실 함수로 $-\mathbf{ln\ L}$를 사용
- $-\ln L = -(\sum_i y_i \ln\{\sigma(w^T X_i)\} + \sum_i (1-y_i)\ \ln\{1-\sigma(w^T X_i)\})$

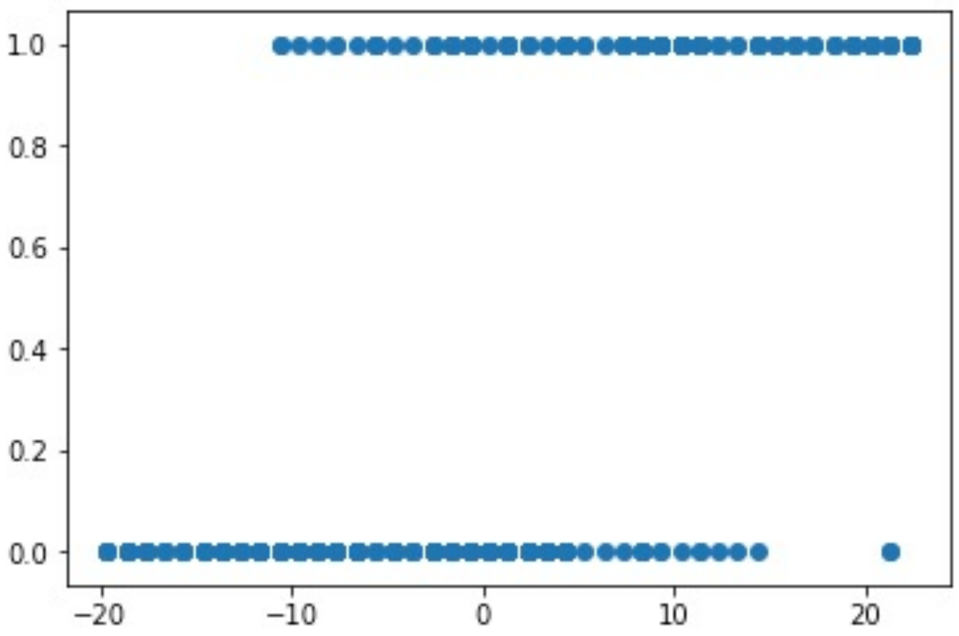
  $= -(\sum_i y_i\ (\ln\{\sigma(w^T X_i)\} - \ln\{1-\sigma(w^T X_i)\}) + \ \ln\{1-\sigma(w^T X_i)\})$

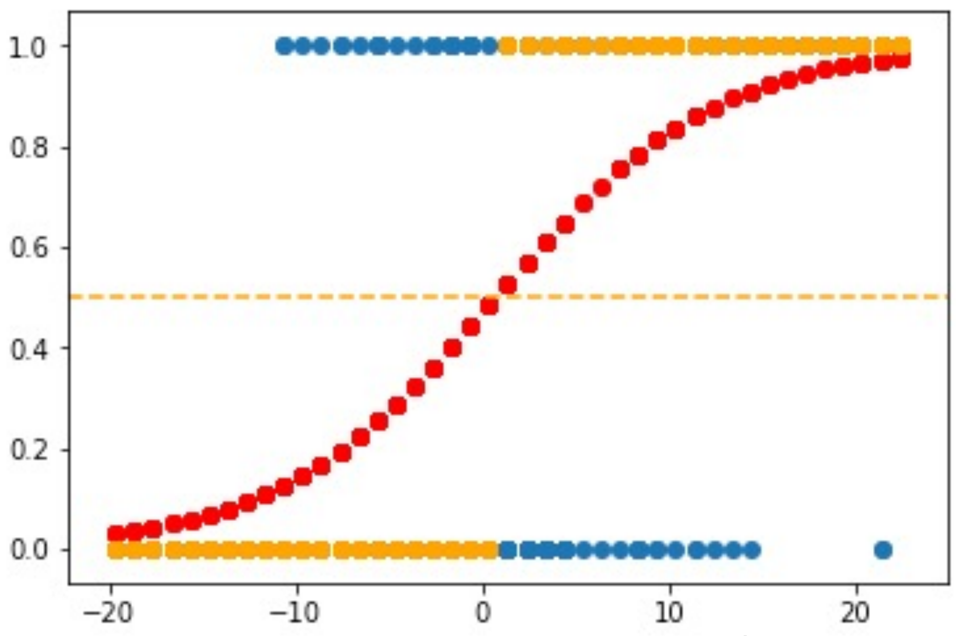
  $= -(\sum_i y_i w^T X_i + \ \ln\{1-\sigma(w^T X_i)\}) = \ \boldsymbol{-(\sum_i y_i w^T X_i -\ \ln\{1+e^{w^T X_i}\})}$

- $\mathbf{0} = \frac{\boldsymbol{\partial} \mathbf{lnL}}{\boldsymbol{\partial} \mathbf{w}} = \{\sum_i y_i X_i\} + \{\sum_i -X_i \frac{e^{w^T X_i}}{1+e^{w^T X_i}}\} = \sum_i X_i (y_i - P(y_i = 1 | X_i; w))$
- $w_{t+1} = w_t - lr \times \frac{\partial \ln L}{\partial w}$

Example for Logistic Regression

로지스틱 회귀 예시





Nonlinear Logistic Regression

비선형 로지스틱 회귀

- $P(\hat{y} = 1|X) = \frac{1}{1+e^{-w^T X}} = \sigma(w^T X)$
- 위의 선형 로지스틱 회귀에서 **$w^T X$를 비선형 회귀 함수로 변형**하면 됨

Evaluation Metrics (1)

오차 행렬 (confusion matrix)

- 성능 측정을 위해 예측값과 실제값을 비교한 표

		예측값	
		Positive	Negative
실제값	Positive	TP (True Positive)	FN (False Negative)
	Negative	FP (False Positive)	TN (True Negative)

Evaluation Metrics (2)

정확도 (accuracy)

- n 개의 데이터 샘플 중 예측에 성공한 샘플의 비율 ($\frac{TP+TN}{TP+FN+FP+TN}$)

예측값

실제값

	Positive	Negative
Positive	TP (True Positive)	FN (False Negative)
Negative	FP (False Positive)	TN (True Negative)

Evaluation Metrics (3)

정밀도 (precision)

- 모델이 Positive로 예측한 것 중 실제값 또한 Positive인 비율 ($\frac{TP}{TP+FP}$)

		예측값	
		Positive	Negative
실제값	Positive	TP (True Positive)	FN (False Negative)
	Negative	FP (False Positive)	TN (True Negative)

Evaluation Metrics (4)

재현도 (recall)

- 실제값이 Positive인 것 중 모델이 Positive로 예측한 비율 ($\frac{TP}{TP+FN}$)

		예측값	
		Positive	Negative
실제값	Positive	TP (True Positive)	FN (False Negative)
	Negative	FP (False Positive)	TN (True Negative)

Evaluation Metrics (5)

F1 Score

- 정밀도와 재현도의 **조화 평균** (역수의 산술평균의 역수, $\frac{2 \times \text{Precision} \times \textit{Recall}}{\textit{Precision} + \text{Recall}}$)

		예측값	
		Positive	Negative
실제값	Positive	TP (True Positive)	FN (False Negative)
	Negative	FP (False Positive)	TN (True Negative)

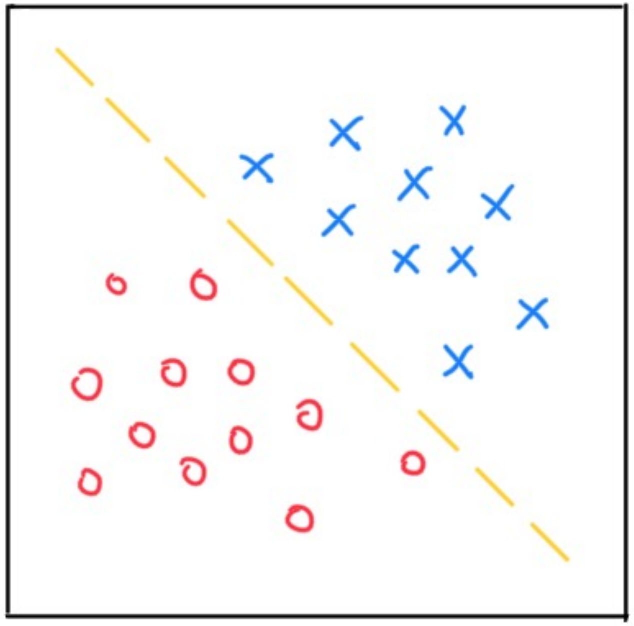
Part Two

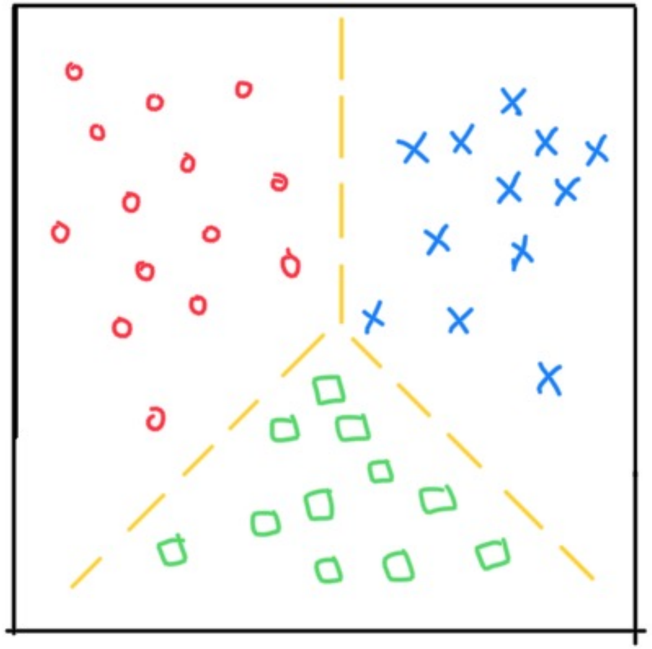
Softmax Regression

Multiclass Classification

다중 분류 모델 (multiclass classifier)

- 분류해야되는 **클래스가 여러 개**인 상황
- 로지스틱 회귀 모델에 대해 **다중 클래스로의 확장**이 필요함





Softmax Function

(Remind) 시그모이드 함수 (sigmoid function)

- 이진 분류 문제를 위한 비선형 함수
- $\mathbf{y} = \frac{\mathbf{1}}{\mathbf{1+e^{-x}}}$
- 함수의 출력값이 항상 0이상 1이하며, 중앙 출력값은 0.5임

(Remind) 소프트맥스 함수 (softmax function)

- **다중 분류 문제**를 위한 비선형 함수
- $\mathbf{y_i} = \frac{\mathbf{e^{x_i}}}{\sum_{k=1}^{K} e^{x_k}}$ ($\mathbf{k}$ 는 클래스 갯수)

Probability

시그모이드 함수와 소프트맥스 함수의 역할

- $y = \frac{1}{1+e^{-x}}$
- $y_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}}$ (k 는 클래스 갯수)
- 입력값을 **확률의 성질을 만족하는 결과값**으로 변환

확률의 성질

1. $0 \leq P(A) \leq 1$
2. $P(S) = 1$
3. $P(\emptyset) = 0$
4. $P(A^c) = 1 - P(A)$

Cross Entropy Loss

(Remind) 로지스틱 회귀의 우도 함수

- $\mathbf{L} = \prod_i \boldsymbol{\sigma}(\mathbf{w}^T\mathbf{X}_i)^{\mathbf{y}_i}\left(\mathbf{1} - \boldsymbol{\sigma}(\mathbf{w}^T\mathbf{X}_i)\right)^{\mathbf{1}-\mathbf{y}_i}$
- 이를 $\prod_i \mathbf{p}(\mathbf{y}_i = \mathbf{c}|\mathbf{X}_i)$ 로 해석 가능
- $\mathbf{y}_i$ 의 값이 0, 혹은 1 의 상황에서 **0~C** 의 상황으로 확장 필요

다중 분류 모델의 우도 함수

- $\prod_i p(y_i = c|X_i) = \prod_i \mathrm{softmax}\left(w^TX_i\right)^{y_i}$
- $\mathbf{L}_{\mathbf{CE}} = -\sum_i^n \mathbf{y}_i \ln\left(\mathbf{softmax}\left(\mathbf{w}^T\mathbf{X}_i\right)\right)$, where $\mathbf{y}_i = [\mathbf{0}, \mathbf{0}, \mathbf{1}, \cdots, \mathbf{0}]$

Part Three

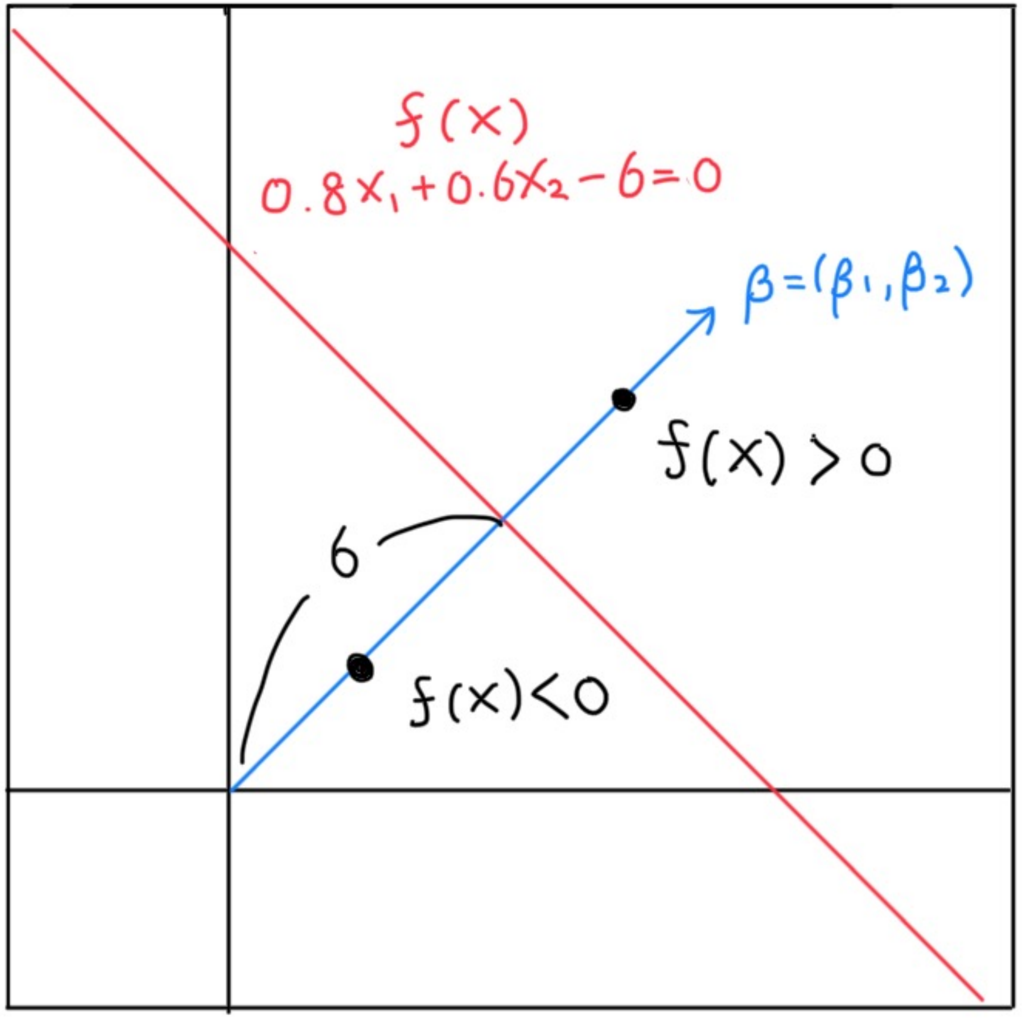
Support Vector Machine

Hyperplane (1)

Hyperplane

- p 차원에서의 hyperplane은 $\mathrm{p}-1$ 차원에서의 **평평한 어핀 공간**임
- $\boldsymbol{b}+\boldsymbol{w}_{\mathbf{1}}\mathbf{X}_{\mathbf{1}}+\boldsymbol{w}_{\mathbf{2}}\mathbf{X}_{\mathbf{2}}+\cdots+\boldsymbol{w}_{\mathbf{p}}\mathbf{X}_{\mathbf{p}}=\mathbf{b}+\langle\mathbf{w},\mathbf{X}\rangle=\mathbf{0}$
- $\mathbf{w}=(\boldsymbol{w}_{\mathbf{1}},\boldsymbol{w}_{\mathbf{2}},\cdots,\mathbf{w}_{\mathbf{p}})$의 **normal vector**는 hyperplane과 **orthogonal 방향**을 의미
- Ex. 2차원에서의 hyperplane은 선, 3차원에서는 면

Hyperplane (2)

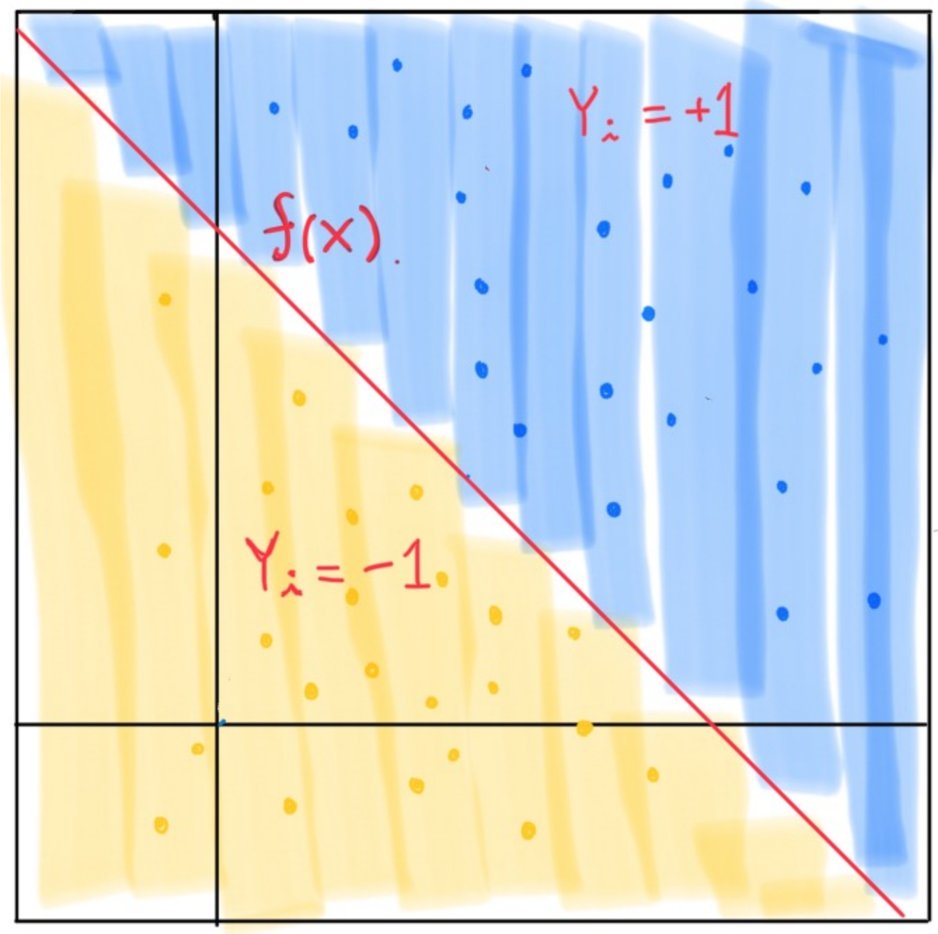


Hyperplane (3)

Separating Hyperplane

- $f(X) = b + \mathbf{w_1 X_1} + w_2\mathbf{X_2} + \cdots + \mathbf{w_p X_p}$
- $\mathbf{f(X) > 0}$ 인 점들과 $\mathbf{f(X) < 0}$ 인 점들을 **분류**
- $\mathbf{Y_i \cdot f(X_i) > 0}$ **for all i**

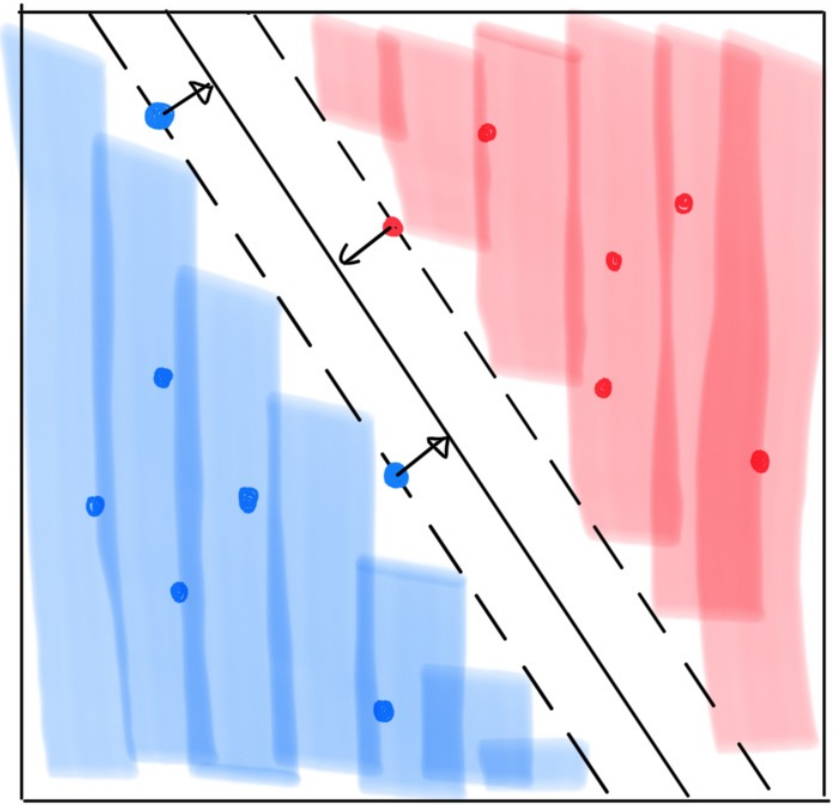
  $\mathbf{f(X) = 0}$은 separating hyperplane을 의미



Maximal Margin Classifier (1)

Maximal Margin Classifier

- 모든 separating hyperplane 중에서, **이진 클래스 사이의 gap 또는 margin을 최대**로 만들어주는 것을 찾음
- 결정 경계에 영향을 미치는 샘플들을 **서포트 벡터 (support vector)**라고 부름
- $$\max_{b, w_1, \cdots, w_p} M$$
  $$\text{subject to } M_i \geq M \text{ for all } i = 1, \cdots, N$$



Maximal Margin Classifier (2)

Maximal Margin Classifier

- $\max_{w,b} \; \mathbf{M}$

  $\textbf{subject to } \boldsymbol{M_i} \geq \mathbf{M} \textbf{ for all i} = \mathbf{1}, \cdots, \mathbf{N}$

Maximal Margin Classifier (3)

Maximal Margin Classifier

- $$\max_{w,b} \mathrm{M}$$

  $$\text{subject to } \frac{y_i(b+w_1x_{i1}+\cdots+w_px_{ip})}{\|w\|} \geq \mathrm{M} \text{ for all } i = 1, \cdots, N$$

Maximal Margin Classifier (4)

Maximal Margin Classifier

- $$\max_{w,b} \frac{\hat{M}}{\|w\|}$$

  $$\text{subject to } y_i\left(b + w_1 x_{i1} + \cdots + w_p x_{ip}\right) \geq \hat{M} = M\|w\|$$

  $$\text{for all } i = 1, \cdots, N$$

Maximal Margin Classifier (5)

Maximal Margin Classifier

- $$\max_{w,b} \frac{1}{\|w\|}$$
  $$\text{subject to } y_i(b + w_1 x_{i1} + \cdots + w_p x_{ip}) \geq 1$$
  $$\text{for all } i = 1, \cdots, N$$

Maximal Margin Classifier (6)

Maximal Margin Classifier

- $\min_{w,b} \frac{\|w\|^2}{2}$

  $\text{subject to } y_i\left(b + w_1 x_{i1} + \cdots + w_p x_{ip}\right) \geq 1$

  $\text{for all } i = 1, \cdots, N$

Lagrange Multiplier Method (1)

라그랑주 승수법 (lagrange multiplier method)

- 제약식이 있는 최적화 문제를 **라그랑주 승수 항을 추가**해, **제약이 없는 문제로** 바꾸는 방법

- **원초 문제 (primal problem)**

$$\min_{\mathbf{x}} \mathbf{c}^{\mathrm{T}}\mathbf{x}$$

$$\text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{G}\mathbf{x} \leq \mathbf{h}$$

- 라그랑주 승수 벡터 $\boldsymbol{u}$와 $\boldsymbol{v} \geq \mathbf{0}$를 도입해 라그랑주 함수 $\boldsymbol{L}$을 만듦

$$\mathbf{L}(\mathbf{x}, \mathbf{u}, \mathbf{v}) = \mathbf{c}^{\mathrm{T}}\mathbf{x} + \mathbf{u}^{\mathrm{T}}(\mathbf{A}\mathbf{x} - \mathbf{b}) + \mathbf{v}^{\mathrm{T}}(\mathbf{G}\mathbf{x} - \mathbf{h}) \leq \mathbf{c}^{\mathrm{T}}\mathbf{x}$$

Lagrange Multiplier Method (2)

라그랑주 승수법 (lagrange multiplier method)

- $f^* \geq \min_{x} L(x, u, v) = \min_{x} \mathbf{c}^T\mathbf{x} + \mathbf{u}^T(\mathbf{Ax} - \mathbf{b}) + \mathbf{v}^T(\mathbf{Gx} - \mathbf{h}) = g(u, v)$
- $g(u, v)$를 라그랑지 듀얼 함수라고 부름
- 편미분의 결과가 0이 되는 지점에서 최소값을 가짐

$$\frac{\partial \mathbf{L}}{\partial \mathbf{x}} = \mathbf{c}^T + \mathbf{u}^T\mathbf{A} + \mathbf{v}^T\mathbf{G} = \mathbf{0}$$

$$\mathbf{c} = -\mathbf{A}^T\mathbf{u} - \mathbf{G}^T\mathbf{v}$$

- $\mathbf{L}(\mathbf{x}, \mathbf{u}, \mathbf{v}) = \mathbf{c}^T\mathbf{x} + \mathbf{u}^T(\mathbf{Ax} - \mathbf{b}) + \mathbf{v}^T(\mathbf{Gx} - \mathbf{h})$

$$= \left(-\mathbf{A}^T\mathbf{u} - \mathbf{G}^T\mathbf{v}\right)^T\mathbf{x} + \mathbf{u}^T(\mathbf{Ax} - \mathbf{b}) + \mathbf{v}^T(\mathbf{Gx} - \mathbf{h})$$

$$= -\mathbf{u}^T\mathbf{b} - \mathbf{v}^T\mathbf{h} = \mathbf{g}(\mathbf{u}, \mathbf{v})$$

Lagrange Multiplier Method (3)

Duality gap

- $f^* \geq \min_x L(x, u, v) = g(u, v)$
- $g(u, v)$를 최대화하는 것은 원초 문제의 최적값과 가까워지는 것
- 둘 사이의 gap이 존재하면 weak dual, 존재하지 않으면 strong dual ($f^* = g(u, v)$)

라그랑주 듀얼 문제 (lagrange dual problem)

- 최소화 문제를 최대화 문제로 바꾸어 풂
- $\max_{\mathbf{u},\mathbf{v}} -\mathbf{u}^{\mathrm{T}}\mathbf{b} - \mathbf{v}^{\mathrm{T}}\mathbf{h}$

  subject to $-\mathbf{A}^{\mathrm{T}}\mathbf{u} - \mathbf{G}^{\mathrm{T}}\mathbf{v} = \mathbf{c},\ \mathbf{v} \geq \mathbf{0}$

Lagrange Multiplier Method (4)

Slater's condition

- $$\min_{x} f(x)$$
  $$\text{subject to } h_i(x) \le 0,\ i = 1, \cdots, m$$
  $$l_j(x) = 0,\ j = 1, \cdots, r$$
- 조건 1: Primal problem이 convex
  (i.e., f and $h_1, \cdots, h_m$ are convex, $l_1, \cdots, l_r$ are affine)
- 조건 2: There exists at least one strictly feasible $x \in \mathbb{R}^n$
  (i.e., $h_1(x) < 0, \cdots, h_m(x) < 0$ and $l_1(x) = 0, \cdots, l_r(x) = 0$)
- 조건 1과 2를 만족하면 **strong duality를 만족함**

KKT Condition

KKT 조건 (Karush-Kuhn-Tucker condition)

- strong duality의 문제(Slater's condition 만족)에서는 다음의 명제를 만족함
- $\boldsymbol{x}^*$ and $\boldsymbol{u}^*, \boldsymbol{v}^*$ are primal and dual solutions

  $\Leftrightarrow$ $\boldsymbol{x}^*$ and $\boldsymbol{u}^*, \boldsymbol{v}^*$ satisfy the KKT conditions
- Stationarity 조건:

  $0 \in \partial\left(f(x) + \sum_{i=1}^{m} u_i h_i(x) + \sum_{j=1}^{r} v_j l_j(x)\right)$
- Complementary slackness 조건:

  $u_i \cdot h_i(x) = 0 \;\; for\ all\ i$

Optimization for Maximal Margin Classifier (1)

(Remind) Maximal Margin Classifier

- $\min_{w,b} \frac{\|w\|^2}{2}$

  subject to $y_i(b + w_1 x_{i1} + \cdots + w_p x_{ip}) \geq 1$

  for all $i = 1, \cdots, N$

Dual form

- $\max_\alpha \min_{w,b} L(w, b, \alpha_i) = \max_\alpha \min_{w,b} \frac{\|w\|^2}{2} - \sum_{i=1}^{N} \alpha_i (y_i(w \cdot x_i + b) - 1)$

  subject to $\alpha_i \geq 0, \ i = 1, \cdots, N$

Optimization for Maximal Margin Classifier (2)

Dual form

- $\max_{\alpha} \min_{w,b} L(w, b, \alpha_i) = \max_{\alpha} \min_{w,b} \frac{\|w\|^2}{2} - \sum_{i=1}^{N} \alpha_i(y_i(w \cdot x_i + b) - 1)$
  subject to $\alpha_i \geq 0$ for all $i = 1, \cdots, N$
- By **Stationary condition** (in KKT condition),

$$\frac{\partial L(w, b, \alpha_i)}{\partial w} = 0, \qquad \frac{\partial L(w, b, \alpha_i)}{\partial b} = 0$$

- 따라서, $w = \sum_{i=1}^{N} \alpha_i y_i x_i , \sum_{i=1}^{N} \alpha_i y_i = 0$

Optimization for Maximal Margin Classifier (3)

Dual form

- $\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$

  subject to $\alpha_i \geq 0, \ \sum_{i=1}^{N} \alpha_i y_i = 0$ for $i = 1, \cdots, N$
- By **Complementary slackness condition** (in KKT condition),

$$\alpha_i(y_i(w^T x_i + b) - 1) = 0 \quad \text{for all } i$$

- 따라서, α_i 또는 $y_i(w^T x_i + b) - 1$ 둘 중 하나는 반드시 0임
- 결정 경계에 영향을 미치는 관측치들은 오직 **support vector** 뿐임
- 그래서 **서포트 벡터 머신 (support vector machine)**이라 부름

Optimization for Maximal Margin Classifier (4)

Dual form

- $\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$

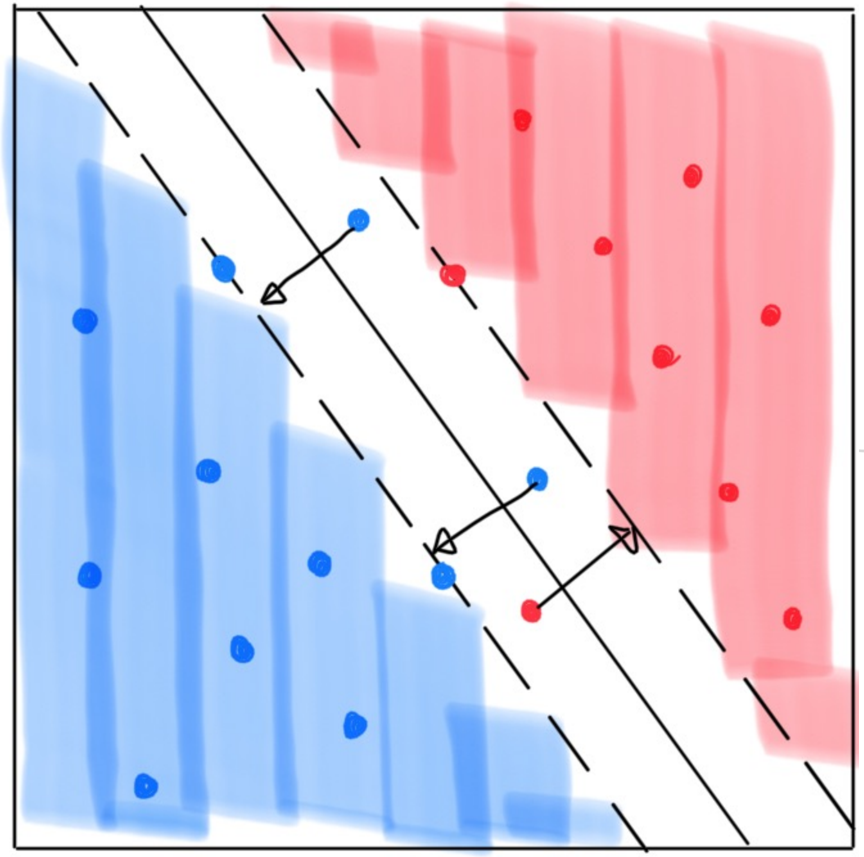
  $\text{subject to } \alpha_i \geq 0, \ \sum_{i=1}^{N} \alpha_i y_i = 0 \ \text{ for } i = 1, \cdots, N$
- 이는 **2차 계획법** (quadratic programming)을 통해 풀이 가능
- α_i 를 통해 w, b 를 계산함

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i , \ b = \frac{1}{N_{SV}} \sum_{i=1}^{N_{SV}} (y_i - w^T x_i)$$

Soft Margin Machine (1)

Non-separable data and Noisy data

- 선형 경계로는 완벽히 나눌 수 없는 경우
- 혹은 나눌 수 있지만 noisy 샘플때문에 비효율적인 경계가 형성되는 경우
- Slack variable ϵ_i 를 도입해 해결
- 이를 soft margin classifier로 부름



Soft Margin Machine (2)

Soft Margin Machine

- $$\min_{w,b,\epsilon} \frac{\|w\|^2}{2} + C\sum_{i=1}^{N} \epsilon_i$$

  $$\text{subject to } y_i(b + wx_i) \geq 1 - \epsilon_i,\ \ \epsilon_i \geq 0$$

  $$\text{for all } i = 1, \cdots, N$$

Soft Margin Machine (3)

Soft Margin Machine

- $\max_{\alpha,\beta} \min_{w,b,\epsilon} \frac{\|w\|^2}{2} + C\sum_{i=1}^{N}\epsilon_i - \sum_{i=1}^{N}\alpha_i[y_i(w\cdot x_i + b) - 1 + \epsilon_i] - \sum_{i=1}^{N}\beta_i\epsilon_i$

  subject to $\alpha_i \geq 0, \beta_i \geq 0$ for all $i = 1, \cdots, N$
- By **Stationary condition** (in KKT condition),

$$\frac{\partial L}{\partial w} = 0, \qquad \frac{\partial L}{\partial b} = 0, \qquad \frac{\partial L}{\partial \epsilon_i} = 0$$

- 따라서, $w = \sum_{i=1}^{N}\alpha_i y_i x_i$, $\sum_{i=1}^{N}\alpha_i y_i = 0$, $\alpha_i = C - \beta_i$

Soft Margin Machine (4)

Soft Margin Machine

- $\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$

  subject to $C \geq \alpha_i \geq 0$, $\sum_{i=1}^{N} \alpha_i y_i = 0$ for $i = 1, \cdots, N$
- By Complementary slackness condition, 두 가지의 Support Vector가 있음

  1. Unbounded SV: $C > \alpha_i > 0$ 의 margin 위에 있는 샘플

  2. Bounded SV: $\alpha_i = C$ $(\epsilon_i \neq 0)$ 의 margin 안에 있는 샘플
- $w = \sum_{i=1}^{N} \alpha_i y_i x_i$, $b = \frac{1}{N_{USV}} \sum_{i=1}^{N_{USV}} (y_i - w^T x_i)$

Soft Margin Machine (5)

Hyperparameter C in SVM

- **C의 값이 커지면:**

  1. ϵ의 영향이 커져 0으로 보내며, 마진의 폭이 줄어듦

  2. Support vector 수가 줄어들어, **적은 샘플로 결정 경계를 찾음**

  3. **Bias ↓, Variance ↑**

- **C의 값이 작으면:**

  1. 마진의 폭이 커짐

  2. 모든 샘플이 support vector가 됨

  3. **Bias ↑, Variance ↓**

Classification for SVM

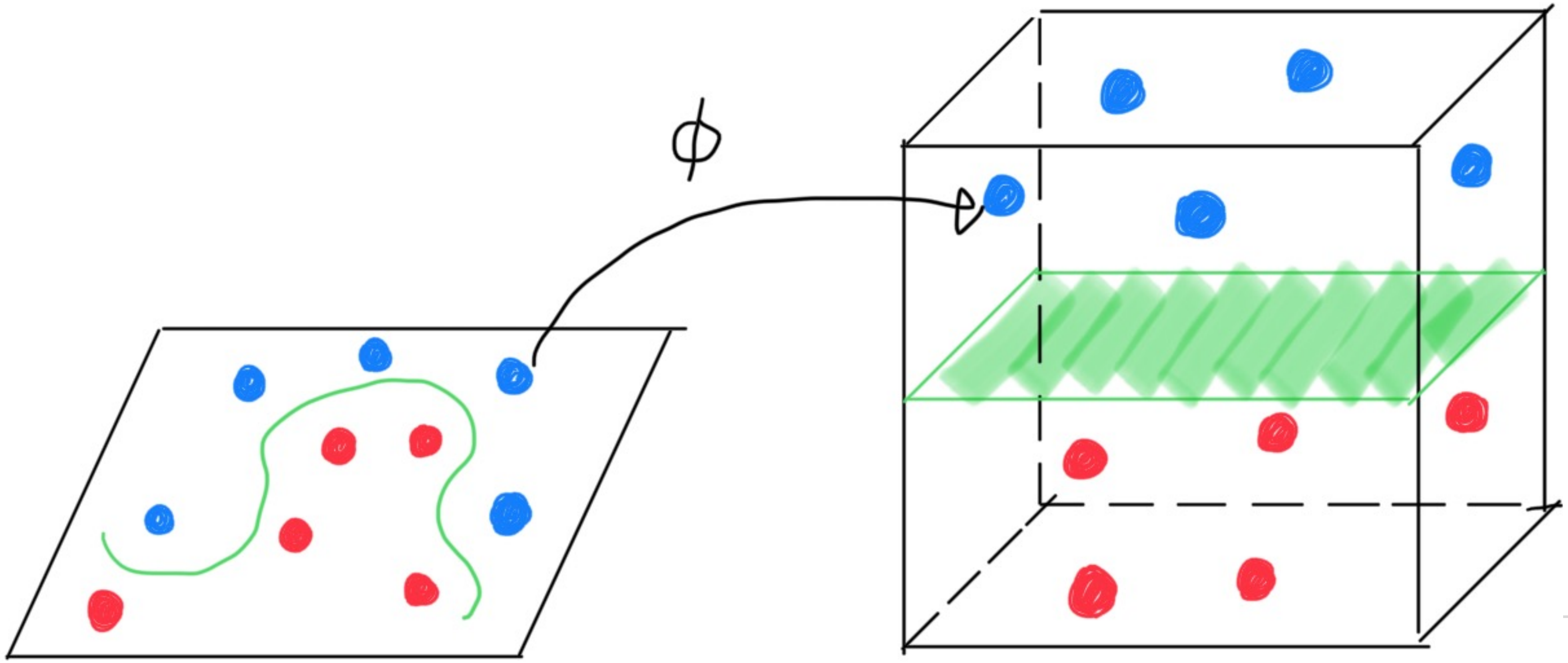
Classification for new data

- 지금까진 주어진 데이터로 결정 경계 hyperplane을 찾는 과정
- $\mathbf{f}(\mathbf{x}) = \mathbf{w}^{\mathbf{T}}\mathbf{x} + \mathbf{b} = \sum_{\mathbf{i} \in \mathbf{S}} \boldsymbol{\alpha}_{\mathbf{i}} \mathbf{y}_{\mathbf{i}} \mathbf{x}_{\mathbf{i}}^{\mathbf{T}} \mathbf{x} + \mathbf{b}$
- 예측값이 0보다 크면 $\hat{y} = 1$로, 0보다 작으면 $\hat{y} = -1$로 분류
- Support vector만 사용되기 때문에 연산량이 많지 않음

Nonlinear SVM (1)

Mapping function

- 비선형 구조의 데이터를 높은 차원으로 이동시켜 그 공간에서 분류하고자 함
- Input space에 존재하는 데이터를 feature space로 옮겨주는 mapping (ϕ)



Nonlinear SVM (2)

Kernel trick

- Mapping 함수 도입시 연산량이 대폭 증가함 ($\boldsymbol{\phi}(\mathbf{x_i})^T\boldsymbol{\phi}(\mathbf{x_j})$)
- 고차원 맵핑과 내적을 한번에 해결하기 위해 커널(Kernel) 도입
- Ex. $\mathrm{K}\left(\mathrm{x_i}, \mathrm{x_j}\right) = \phi(\mathrm{x_i})^T\phi\left(\mathrm{x_j}\right) = (\mathrm{Ax_i})^T\left(\mathrm{Ax_j}\right) = \mathrm{x_i}^T\mathrm{A}^T\mathrm{Ax_j}$ (m차원에서 n차원으로)
- Mercer's Theorem

  1. $\mathrm{K}(\mathrm{x_i}, \mathrm{x_i})$의 경우 **항상 0 이상의 값**을 지님 (positive semi-definite matrix 조건)

  2. $\mathbf{K}\left(\mathbf{x_i}, \mathbf{x}_j\right) = \mathbf{K}(\mathbf{x_j}, \mathbf{x_i})$ (대칭 행렬 조건)

  두 조건을 만족시, $\mathrm{K}\left(\mathrm{x_i}, \mathrm{x_j}\right) = \phi(\mathrm{x_i})^T\phi\left(\mathrm{x_j}\right)$를 만족하는 어떤 ϕ 가 존재함

Nonlinear SVM (2)

Kernel trick

- 조건을 만족하는 임의의 함수를 모두 커널 함수로 사용 가능
- Linear: $K(x_i, x_j) = x_i^T x_j$
- Polynomial: $K(x_i, x_j) = (x_i^T x_j + c)^d$
- Gaussian: $K(x_i, x_j) = \exp\left\{\frac{-\|x_i - x_j\|_2^2}{2\sigma^2}\right\}$
- Radial: $K(x_i, x_j) = \exp\{-\gamma \sum_{k=1}^{p} (x_{ik} - x_{jk})^2\}$

Nonlinear SVM (3)

Kernel trick

- Ex. $\mathbf{K}\left(\mathbf{x_i}, \mathbf{x_j}\right) = (\mathbf{x_i^T x_j})^{\mathbf{2}}$
- $\mathrm{K}(\langle \mathrm{x}_1, \mathrm{x}_2 \rangle, \langle \mathrm{z}_1, \mathrm{z}_2 \rangle) = (\mathrm{x} \cdot \mathrm{z})^2 = (\mathrm{x}_1 \mathrm{z}_1 + \mathrm{x}_2 \mathrm{z}_2)^2$

  $= \left\langle \mathbf{x_1^2}, \sqrt{\mathbf{2}}\mathbf{x_1 x_2},, \mathbf{x_2^2} \right\rangle \cdot \left\langle \mathrm{z}_1^2, \sqrt{2}\mathrm{z}_1 \mathrm{z}_2,, \mathrm{z}_2^2 \right\rangle = \boldsymbol{\phi}(\mathbf{x_1}, \mathbf{x_2}) \cdot \phi(\mathrm{z}_1, \mathrm{z}_2)$

Nonlinear SVM (4)

SVM with kernel trick

- $\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j y_i y_j K(x_i, x_j)$

  subject to $\alpha_i \geq 0,\ \sum_{i=1}^{N} \alpha_i y_i = 0$ for $i = 1, \cdots, N$

- $w = \sum_{i=1}^{N} \alpha_i y_i \phi(x_i),\ b = \frac{1}{N_{SV}} \sum_{i=1}^{N_{SV}} (y_i - (\sum_{j=1}^{N_{SV}} \alpha_j y_j K(x_i, x_j)))$
- 분류시 커널 함수가 사용되므로, mapping function에 대한 정의가 필요 없음

  $f(\phi(x)) = w^T\phi(x) + b = \sum_{i\in S} \alpha_i y_i \phi(x_i)^T \phi(x) + b = \sum_{i\in S} \alpha_i y_i K(x_i, x) + b$

Multiclass SVM

SVM for multiclass

- OVA (One versus All):

  K개의 2-class SVM 을 학습하며 각자의 class와 나머지 클래스로 나눔

  $\hat{f}_k(x)$의 값이 가장 큰 클래스로 분류

- OVO (One versus One):

  $\binom{K}{2}$ 개의 pairwise classifier($\hat{f}_{kl}(x)$)를 학습

  Pairwise competition을 가장 많이 이긴 클래스로 분류

SVM vs. Logistic Regression

SVM vs. LR

- 클래스가 거의 separable하면, SVM > LR
- 아닐 경우, LR(with ridge penalty) == SVM
- 확률값을 측정하고 싶으면, LR을 사용
- Nonlinear boundary에는, kernel SVM이 계산적인 면에서 더 좋음

Part Four

Decision Tree

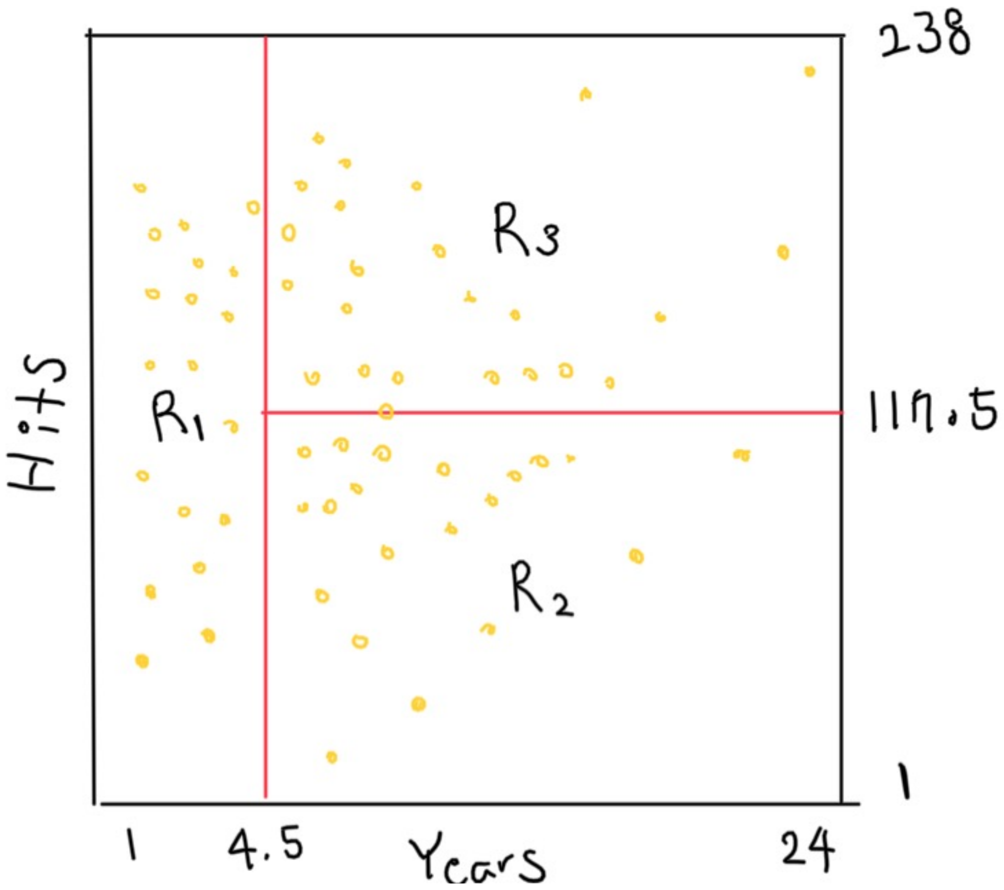
Tree-based Methods

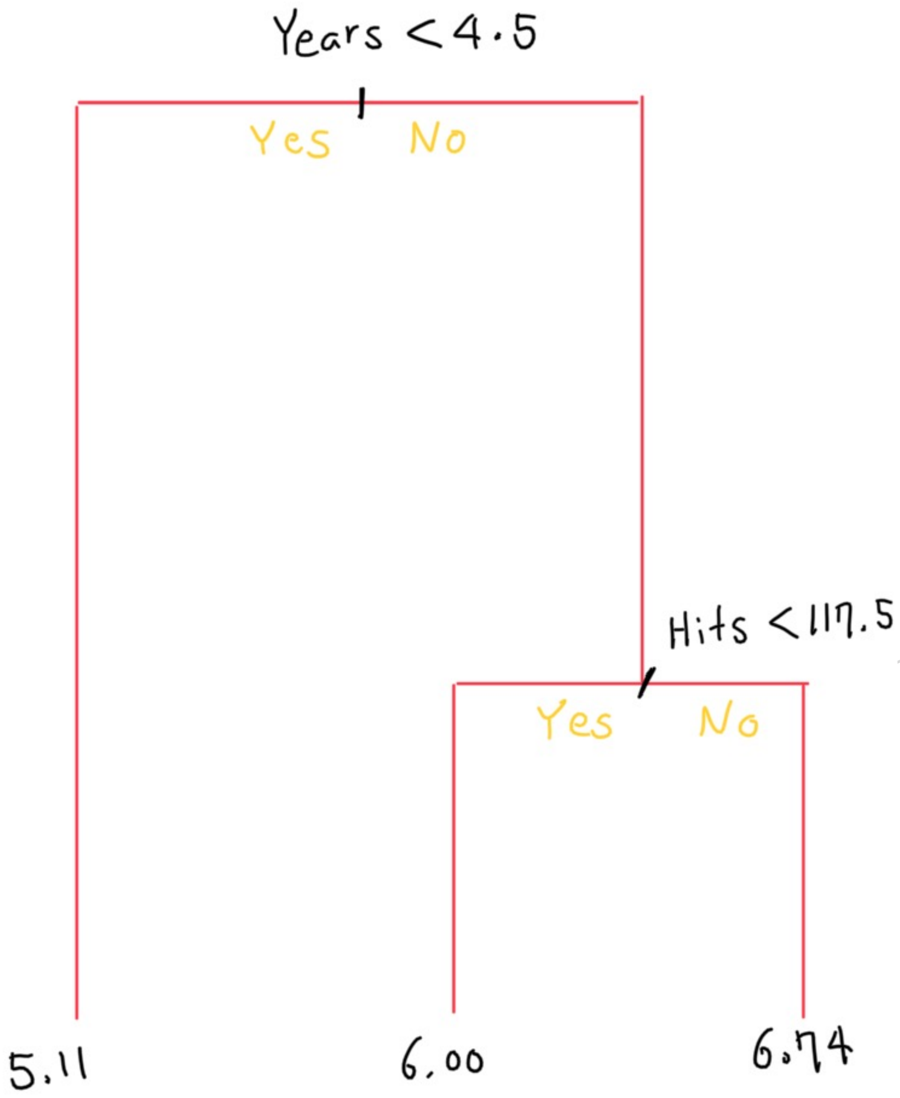
Tree-based methods

- 예측을 위해 여러 region으로 stratifying or segmenting 하는 방법론
- **회귀와 분류 모두에서 사용 가능함**

Tree-based Methods

Baseball salary data

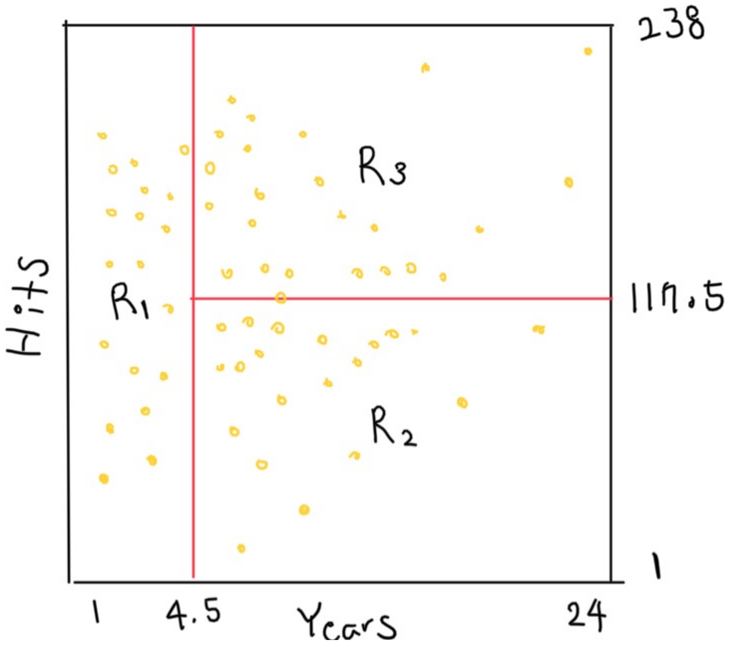


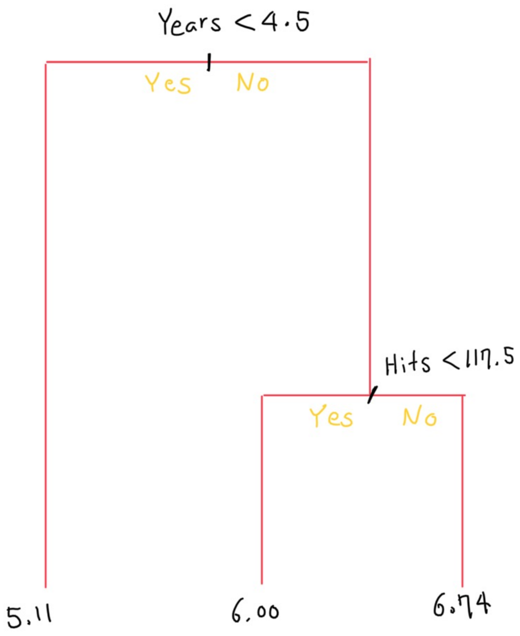


Terminology for Trees

용어 정리

- Terminal nodes: 결정 트리로 나뉘어진 Region, leaf라고도 표현
- 결정 트리는 terminal node가 아래에 오도록 **upside down 형식**으로 그림
- **Internal nodes**: predictor space가 나뉘어지는 부분





Regression with Decision Tree

회귀 결정 트리

- 결정 트리는 predictor space를 보통 사각형 또는 **box** 형태로 나눔
- 이는 예측 모델의 **간단성과 해석의 용이함**을 위한 것
- 회귀 결정 트리는 RSS(residual sum of squares)를 최소화하는 boxes $R_1, \cdots, R_J$를 찾는 것이 목적임

$$\sum_{j=1}^{J} \sum_{x_i \in R_j} \left(y_i - \bar{y}_{R_j} \right)^2$$

Greedy Algorithm (1)

Greedy tree-building

- $\mathbf{R_1}$(전체 input space)를 시작으로 다음의 과정을 반복함
  1. RSS를 최대로 감소시키는 $\mathrm{R_k}$ $(\mathrm{with}\ \mathrm{X_j} < \mathrm{s})$ 를 찾음

$$\sum_{\mathbf{m=1}}^{|\mathbf{T}|} \sum_{\mathbf{x_i} \in \mathbf{R_m}} \left(\mathbf{y_i} - \bar{\mathbf{y}}_{\mathbf{R_m}}\right)^{\mathbf{2}}$$

  2. Splitting point $\mathbf{s}$를 기준으로 region을 새롭게 정의함

Greedy Algorithm (2)

Greedy tree-building

- Greedy 방식은 **일정 기준(각 region에 5개 이하의 샘플) 만족 시 멈춤**
- 모든 가능한 region을 고려하는 것은 **계산상 불가능함**
- 따라서 Top-down, greedy 방법론(recursive binary splitting)을 사용함.
- **root에서부터 leaves까지** 트리를 생성하기 때문에 top-down이라고 불림
- Greedy라 불리는 이유는, 이전이나 이후 상황이 아닌 딱 **현재 상황에서의 best split**을 행하기 때문임

Greedy Algorithm (3)

Greedy tree-building

- Terminal node의 수가 많아질수록,
  1. RSS 값이 0으로 수렴
  2. Over-fitting 이슈 발생 (Bias ↓, Variance ↑)
  3. 모델 학습을 위한 computation cost 증가

Preventing Over-fitting

Over-fitting 방지 idea

- **교차 검증 (cross validation)**을 통해 optimal subtree 찾음

  -> 하지만 경우의 수가 너무 많기 때문에, over-fitting을 완벽히 막기 힘듬
- RSS 값이 **일정 threshold 만큼 떨어지지 않으면** tree 성장을 멈춤

  -> 다음 성장에서 큰 RSS drop이 일어날 수도 있음

Cost Complexity Pruning

가지치기(pruning) 손실 함수

- 기존의 손실 함수 RSS에 가지치기를 위한 정규화 항을 추가함

$$\mathbf{minimize} \sum_{R_m \in T} \sum_{x_i \in R_m} \left(y_i - \bar{y}_{R_m}\right)^2 + \alpha|T|$$

- $|T|$: # of terminal node
- $\alpha = \infty$ 라면, **null 트리** 생성 (한개의 leaf만으로 구성된 트리)
- $\alpha = 0$ 라면, **full 트리** 생성
- α 하이퍼 파라미터는 교차 검증을 통해 구할 수 있음

Classification with Decision Tree

분류 결정 트리

- 회귀 결정 트리와 매우 유사하지만, RSS의 손실 함수 사용 불가
- 평균값이 아닌 Majority vote를 통해 예측 (i.e., **각 region의 가장 많은 클래스를 뽑음**)
- 새로운 분류 손실 함수가 필요

Classification Loss Function (1)

Misclassification rate (classification error rate)

- Region 안의 샘플 중에서 **most common class에 포함되지 않은 샘플의 수를 계산**
- 두 가지 형태의 손실 함수로 표현 가능

  1. **minimize** $\sum_{m=1}^{|T|} \sum_{x_i \in R_m} I(y_i \neq \hat{y}_{R_m})$

  2. **minimize** $1 - \max_{k} \hat{p}_{mk}$

     $\hat{p}_{mk}$: m-th region에서 k-th class에 해당하는 학습 데이터의 비율

- 하지만 트리 성장에 있어서 충분히 민감하지 못한 단점

Classification Loss Function (2)

Gini index

- K개 클래스의 분산에 대한 관측치
- **minimize** $\sum_{m=1}^{|T|} q_m \sum_{k=1}^{K} \hat{p}_{mk}(1-\hat{p}_{mk})$

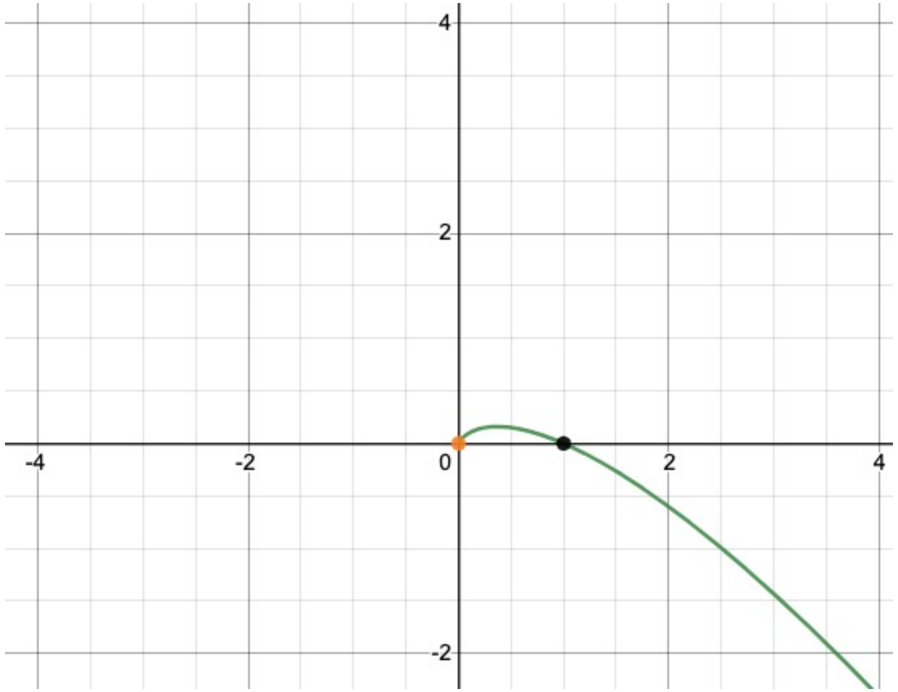
  $\hat{p}_{mk}$: m−th region에서 k−th class에 해당하는 학습 데이터의 비율

  q_m: 전체 데이터 개수에 대한 region R_m에 있는 샘플 수의 비율
- $\hat{p}_{mk}$이 모두 0 아니면 1에 수렴할수록 좋아짐
- Gini index 값이 작으면 single 클래스가 node를 장악한 상황이므로, **node purity**에 대한 관측치로도 해석 가능

Classification Loss Function (3)

Cross-entropy

- Gini index와 매우 유사한 손실 함수클래스의 분산에 대한 관측치
- **minimize** $-\sum_{m=1}^{|T|} q_m \sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk})$



Pros and Cons for Decision Tree

Pros:

- 모델에 대한 해석과 설명이 쉬움
- 인간의 의사 결정과 매우 비슷한 형태의 모델임
- 시각적으로 보여주기 편리하며, 비전문가도 쉽게 이해 가능

Cons:

- 다른 회귀, 분류 모델에 비해 예측 성능이 일반적으로 떨어짐
- 하지만 이는 많은 수의 결정 트리의 결과를 종합하는 **Ensemble 학습(e.g., Bagging, Boosting)**으로 보완 가능함

Part Five

Linear Discriminant Analysis

Bayes' Classifier (1)

베이즈 분류기 (bayes' classifier)

- $\mathbf{P}(Y = k|\mathbf{X} = \mathbf{x}) = \frac{\mathbf{P}(\mathbf{X} = \mathbf{x}|Y = k)\mathbf{P}(\mathbf{Y}=\mathbf{k})}{\mathbf{P}(\mathbf{X}=\mathbf{x})}$

$$= \frac{P(X = x|Y = k)P(Y = k)}{\sum_{i=1}^{K} P(X = x|Y = i)P(Y = i)}$$

- 사후 확률값을 가장 크게 만들어주는 클래스로 분류하는 것
- $arg \max_{1 \le k \le K} P(Y = k|X = x)$

Bayes' Classifier (2)

베이즈 분류기 (bayes' classifier)

- $\mathrm{P}(Y = k|\mathrm{X} = \mathrm{x}) = \dfrac{\mathrm{P}(X = \mathrm{x}|Y = k)\mathrm{P}(\mathrm{Y{=}k})}{\sum_{l=1}^{K} P(X = x|Y = l)P(Y{=}l)}$
- 사전 확률 $\boldsymbol{\pi_k}$: 데이터가 k번째 클래스에서 뽑혔을 확률
- Density function : $\boldsymbol{f_k(x)} = P(X = x|Y = k)$

- (Remind) 우도 확률 $P(X|Y)$: 각 샘플이 i.i.d 할 때, PDF (probability density function)의 곱과 동일

Bayes' Classifier (3)

베이즈 분류기 (bayes' classifier)

- $$\mathrm{P}(Y=k|\mathrm{X}=\mathrm{x})=\frac{\mathrm{P}(X=\mathrm{x}|Y=k)\mathrm{P(Y{=}k)}}{\sum_{l=1}^{K}P(X=x|Y=l)P(Y=l)}$$

$$=\frac{\pi_k f_k(x)}{\sum_{l=1}^{K}\pi_l f_l(x)}$$

Linear Discriminant Analysis (1)

선형 판별 분석 (linear discriminant analysis)

- 다음의 두 가지 가정을 사용함

1. Density function이 Normal 혹은 Gaussian density를 따른다

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma_k}\right)^2}$$

2. $\sigma_k = \sigma \quad for\ all\ k$

- $$\boldsymbol{p_k(x) = P(Y = k|X = x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma}\right)^2}}{\sum_{i=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu_l}{\sigma}\right)^2}}}$$

Linear Discriminant Analysis (2)

판별 함수 (discriminant function)

- 데이터 $X = x$ 를 분류하기 위해 판별 함수를 정의해야 함
- 판별 함수는 $p_k(x)$에 대한 값을 뱉는 함수
- $arg \max_{1 \le k \le K} P(Y = k | X = x) = arg \max_{1 \le k \le K} \pi_k \exp(-\frac{1}{2\sigma^2}(x^2 - 2\mu_k x + \mu_k^2))$

$$= arg \max_{1 \le k \le K} \exp(-\frac{1}{2\sigma^2}(x^2 - 2\mu_k x + \mu_k^2) + \log(\pi_k))$$

$$= arg \max_{1 \le k \le K} \boldsymbol{x} \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

- 두 번째 가정에 따라 quadratic 항을 삭제할 수 있어 선형 판별 함수를 갖게 됨

Linear Discriminant Analysis (3)

판별 함수 (discriminant function)

- 판별 함수 $\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$
- 데이터를 통한 추정값 사용

  1. $\hat{\pi}_k = \frac{n_k}{n}$

  2. $\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$

  3. $\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$

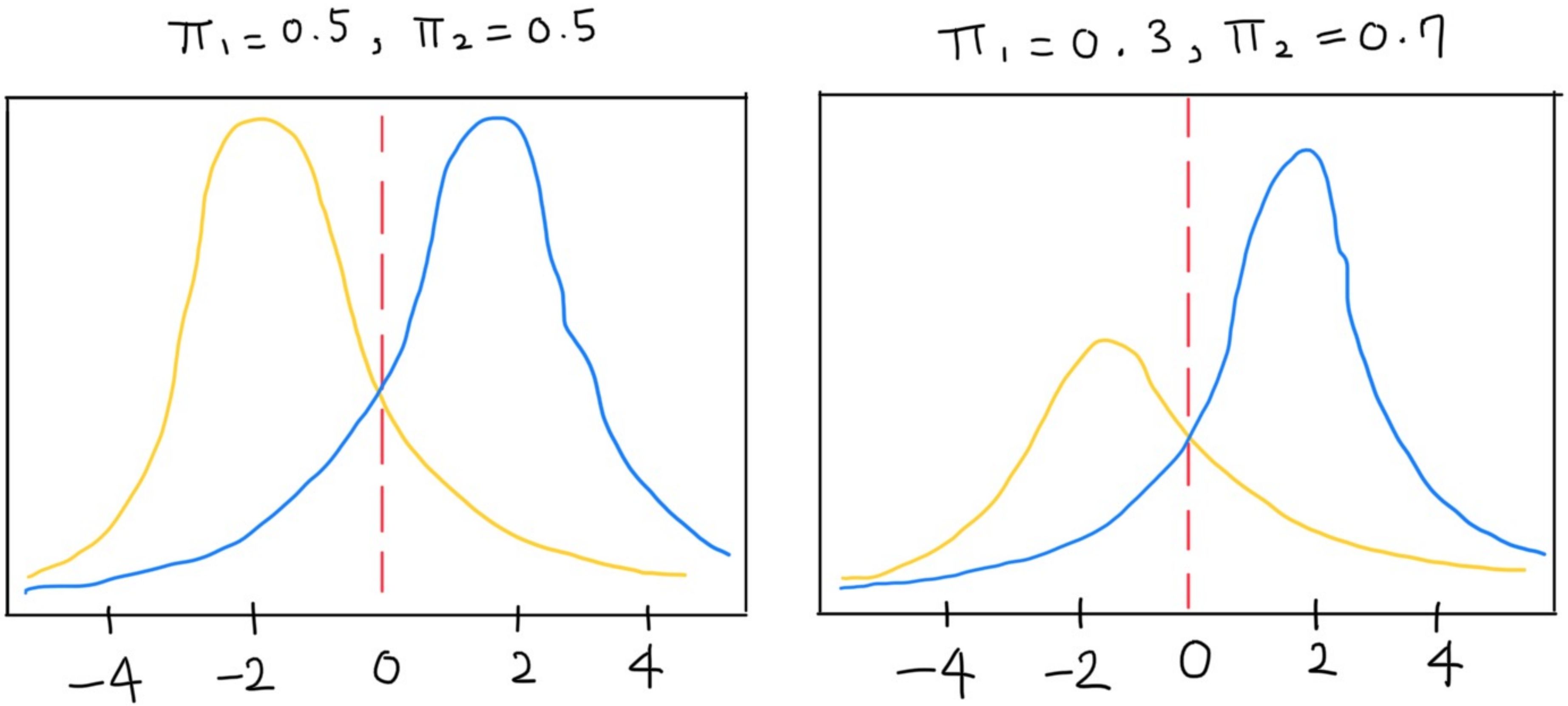
     (cf. 통계적 추정을 할 땐 표본자료 중 모집단에 대한 정보를 주는 독립적인 자료를 사용)

- $\hat{\delta}_k(x) = \boldsymbol{x} \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k)$

Linear Discriminant Analysis (4)

판별 함수 (discriminant function)

- 결정 경계는 $\delta_1(x) = \delta_2(x)$를 통해 얻을 수 있음



LDA for Multiple Feature

LDA for $\mathbf{p} > \mathbf{1}$

- 피처의 종류가 2개 이상인 경우, 두 가지 가정을 다음과 같이 수정함
  1. Density function이 Multivariate Gaussian density를 따른다

$$\mathrm{f_k(x)} = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}$$

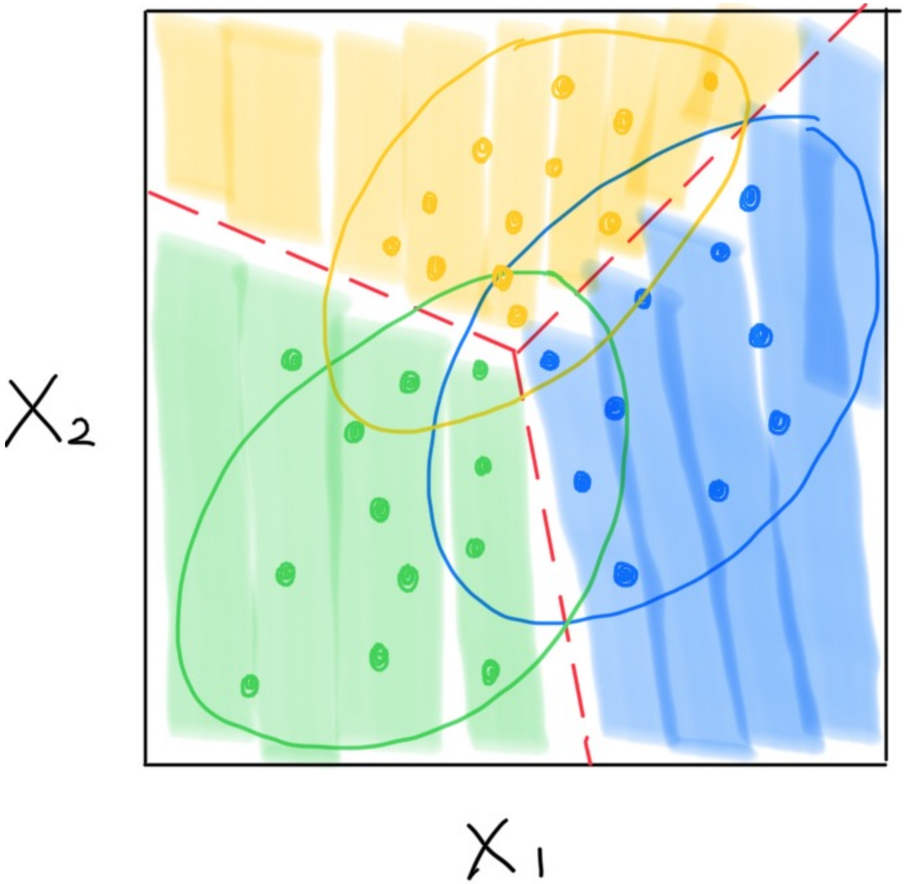
  2. $\Sigma_k = \Sigma \quad for\ all\ k$

- $\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log(\pi_k)$

$$= c_{k0} + c_{k1}x_1 + c_{k2}x_2 + \cdots + c_{kp}x_p$$

LDA for Multiple Feature

LDA for $\mathbf{p} = \mathbf{2}, \mathbf{K} = \mathbf{3}$



Quadratic Discriminant Analysis

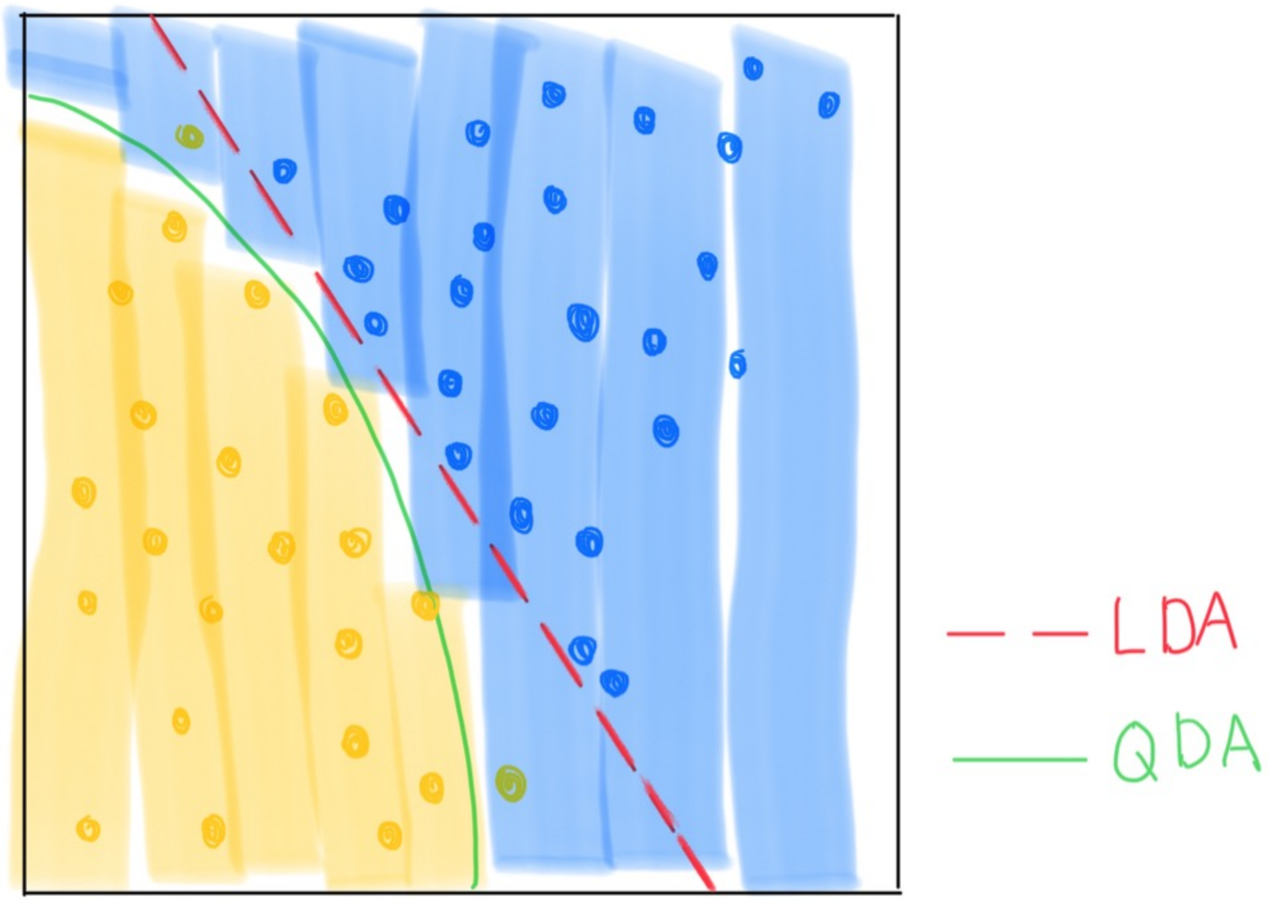
Quadratic Discriminant Analysis

- 비선형 결정 경계를 위해선 QDA를 사용해야 함
- 두 번째 가정을 없애고, 각 class는 각자의 covariance matrix Σ_k를 갖음
- $\mathrm{f_k}(\mathrm{x}) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)}$
- $\delta_k(x) = -\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\Sigma_k| + \log\pi_k$

$$= -\frac{1}{2}x^T\Sigma_k^{-1}x + x^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\log|\Sigma_k| + \log\pi_k$$

Quadratic Discriminant Analysis

QDA vs. LDA



Thank you

Classification Model