



ESC

Classification & SVM

목차 / CONTENTS

1. Introduction

- An Overview of Classification
- Why Not Linear Regression?

2. Classification

- Logistic Regression
- Generative Models for Classification

3. Support Vector Machines

- Maximal Margin Classifier
- Support Vector Classifier(Soft Margin Classifier)
- Support Vector Machine with Kernel

4. Summary



01

Introduction

Simple Examples

다음과 같은 간단한 세 가지의 상황을 생각해 보자.

1. 저번주에 회장님의 아파서 세션에 빠지고 병원에 갔는데, 기침과 콧물이 있으나 증세가 비교적 가벼워 보인다.

의사는 회장님이 감기와 가벼운 독감 중 어떤 것을 앓고 있는지 판단해야 한다.

→ 여러 가지 증상 정보를 바탕으로 “이 사람의 상태가 무엇인지” 분류하는 문제.

2. 온라인 쇼핑몰이나 은행 시스템에서, 사용자 접속 기록(아이디, IP 주소 등)과 결제 패턴을 분석하여

해당 거래가 정상적인 거래인지, 비정상적인 거래인지를 분류해야 한다.

→ 주어진 여러 데이터를 종합해 “정상/비정상” 범주로 나누는 문제.

3. 포켓몬 게임을 시작할 때 일반적으로 물, 불, 풀타입의 세 가지 스타팅 포켓몬 중 하나를 고르게 된다.

플레이어의 특징이나 성향, 성격에 따라 어떤 타입을 고를 것인지 예측하고 싶다.

→ 주어진 데이터를 종합해 특정 카테고리(불/물/풀) 중 하나를 예측하는 문제.



An overview of Classification

Regression(회귀)과 **Classification(분류)**은 모두 **Supervised learning(지도 학습)** 알고리즘으로,
두 알고리즘 모두 머신러닝의 예측에 사용되며 레이블이 지정된 데이터셋과 함께 사용된다.

What is a Statistical Learning?

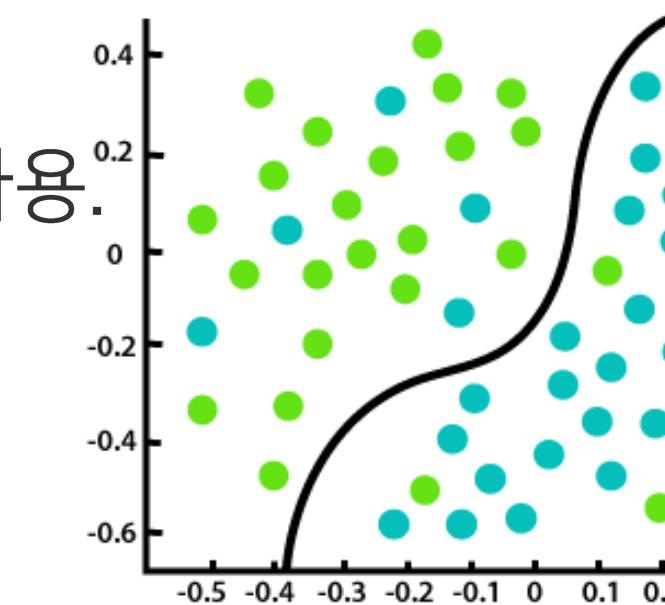
: vast set of tools for understanding data.

Supervised Statistical Learning

: involves building a statistical model for predicting(estimating) an output based on inputs.

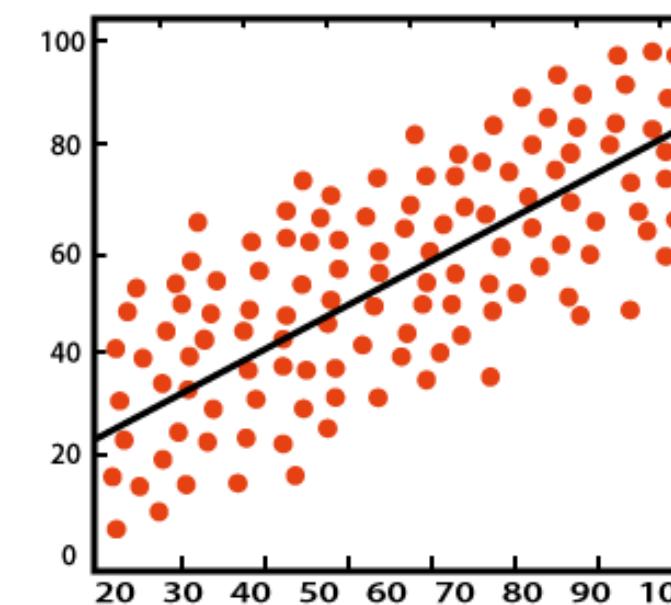
Classification의 경우 남성 vs 여성, 참 vs 거짓 등 **discrete한 value**를 예측/분류하는데 사용.

Regression은 가격, 급여, 나이 등과 같은 **continuous한 value**를 예측하는데 사용.



Classification

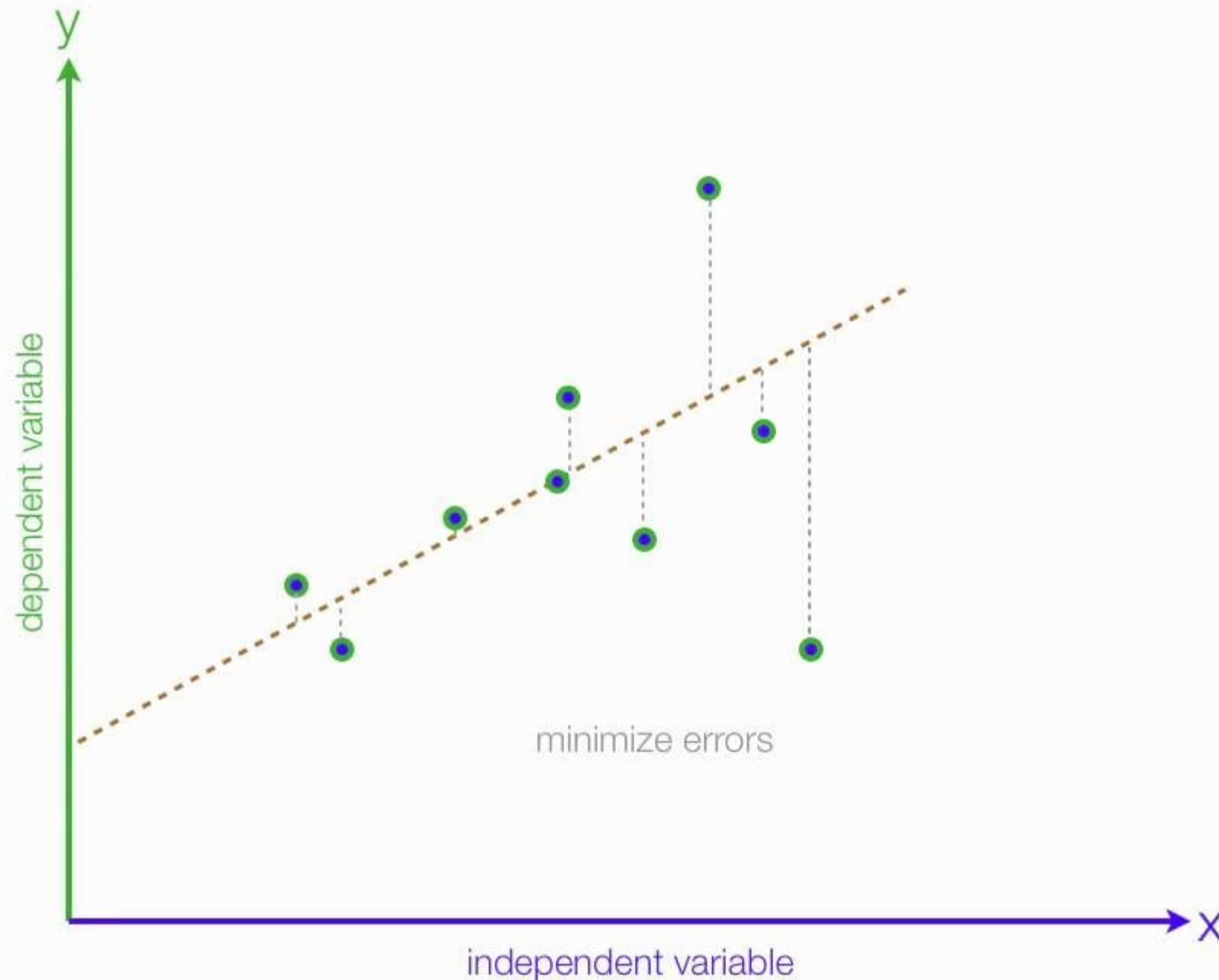
<https://www.analyticsvidhya.com/blog/2020/11/popular-classification-models-for-machine-learning/>



Regression



Linear Regression



<https://i.ytimg.com/vi/zPG4NjlkCjc/maxresdefault.jpg?sqp=-oaymwEmCIAKENAF8quKqQMa8AEB-AH-CYAC0AWKAgwIABABGFlgZShfMA8=&rs=AOn4CLBZnskaRI7tBftWcYmNlmlZv2i-fw>

일반적인 **Linear Regression**은 $Y = X\beta + \epsilon$ 형태에서
OLS(Ordinary Least Square)방법을 이용한다.

Linear Regression은 앞서 말했듯 **Qualitative**한 경우보다는
Quantitative한 경우에 더 많이 이용된다.

Linear Regression도 물론 **Qualitative**한 Y를 처리할 수 있다.
그럼에도 왜 **Regression**과 **Classification**을 따로 생각할까?



Why not Linear Regression?

Linear Regression을 사용할 때 생기는 문제를 피부로 느끼기 위해 다음과 같은 예시를 생각해보자.

응급실에는 많은 환자들이 방문한다. 환자의 증상을 기반으로 해서 환자가 어떤 병에 걸렸는지 예측해야 한다.

1. 뇌졸증, 2. 약물 중독, 3. 뇌전증이라는 총 3가지의 진단만 존재한다고 가정하자.

그렇다면 Y를 환자가 걸린 병이라는 확률변수로 두고, 다음과 같이 modeling이 가능하다.

$$Y = \begin{cases} 1 & \text{if stroke;} \\ 2 & \text{if drug overdose;} \\ 3 & \text{if epileptic seizure.} \end{cases}$$

언뜻 보면 이 상태로 least squares를 진행시켜도 될 것 같지만, 정말 큰 문제가 여럿 존재한다...

무엇일까?



Why not Linear Regression?

우선, 환자의 질병을 숫자로 변환하면서 인위적인 순서가 부여되었다(**ordering on the outcomes**).

$Y = \{1, 2, 3\}$ 으로 정하게 된 순간, 우리도 모르게 숫자 1, 2, 3 사이의 관계가 어떤 의미를 가질 것이라고 강제된다.

??? 난 그런 적 없는데요? 라고 생각할 수 있으나,

$$Y = \begin{cases} 1 & \text{if stroke;} \\ 2 & \text{if drug overdose;} \\ 3 & \text{if epileptic seizure.} \end{cases}$$

우리가 가정한 위와 같은 상황에서는

뇌졸증과 약물 중독 사이의 차이가 약물중독과 뇌전증 사이의 차이와 같다고 가정하는 효과가 생겨버린다.

즉, 수학적으로는 1과 2의 차이(1)와 2와 3의 차이(1)가 같아야 하지만 이런 순서는 그저 임의적일 뿐이지 실제 상황과 무관하다.

$$Y = \begin{cases} 1 & \text{if epileptic seizure;} \\ 2 & \text{if stroke;} \\ 3 & \text{if drug overdose,} \end{cases}$$

이렇게 조금만 다르게 modeling해버리게 되면, 같은 data를 가지고도 전혀 다른 결과를 도출할 수도 있게 된다.



Why not Linear Regression?

그렇다면 조금 더 쉬운 상황을 가정해보자.

Y에서 뇌전증을 빼고 뇌졸증과 약물 중독만을 고려해보자(binary qualitative response).

Linear Regression에서 predictor 부분에서 categorical을 다룰 때 사용하는 dummy variable의 관점을 생각해보자.

$$Y = \begin{cases} 0 & \text{if } \text{stroke}; \\ 1 & \text{if } \text{drug overdose}. \end{cases}$$

이런 modeling 하에서 Linear Regression을 적합(fitting)시켜 \hat{Y} 이 0.5보다 크면 약물 중독으로, 작으면 뇌졸증로 보자.
(이때는 참고로 0과 1을 바꿔서 modeling 해도 동일한 결과를 얻는다.)

일반적으로 Linear Regression은 $E = [Y | X]$ 을 모델링하고, binary case에서는 $E = [Y | X] = P(Y = 1 | X)$ 이므로
 $\hat{Y} = X \hat{b}$ 는 binary response의 estimate로도 볼 수 있다.

어 그럼 2개짜리에서는 이렇게 해도 되는거 아닌가? 라는 생각을 할 수도 있는데, 미안하지만 아니다…
정말 간단하면서도 우리의 직관을 벗어나는 문제점이 존재한다.



Why not Linear Regression?

오른쪽과 같이 Linear Regression을 fitting 시켰다고 했을 때

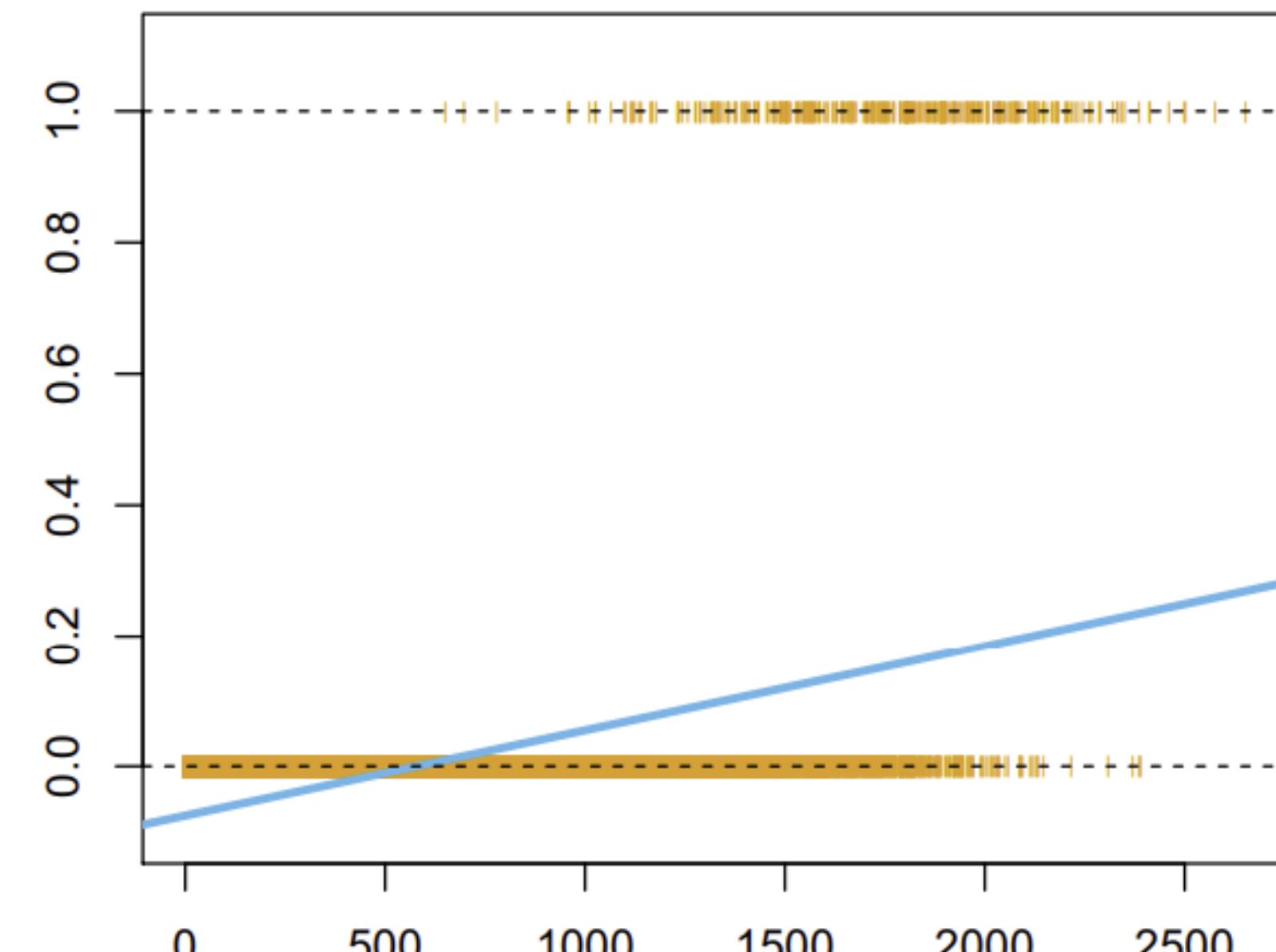
우리는 \hat{Y} 를 확률의 추정치라고 생각할 수 있다고 했지만,

특정 구간에서는 0보다 작거나 1보다 커지는 일이 생겨버린다.

예측값에 따라 Y의 가능성에 대해서 생각해볼 수는 있지만,

이 예측값을 절대적인 확률로 해석하기에는 무리가 있고,

해석에서도 어려움을 겪을 수밖에 없게 된다.



정리해보자면,

- (1) Regression method는 3개 이상의 class(category)를 가지는 qualitative response에 대해서는 사용하기 어렵고
- (2) 2개만을 다루는 binary case에서도 확률의 추정치로 사용하기에도 다소 문제점이 존재한다.



02

Classification

Logistic Regression

Logistic Regression: Probability를 Modeling.

Rather than modeling the **response** variable Y , logistic regression models the **probability that Y belongs to a particular category**.

데이터가 특정 **category**에 속할 확률을 0~1사이의 값으로 예측하고, 그 확률에 따라 가능성이 더 높은 category에 속하는 것으로 분류!

즉, 특정 threshold 값을 정해서 확률의 추정치가 threshold 이상인 것과 미만인 것으로 분류한다는 말이다.

그렇다면 어떻게 $P(X) = P(Y = 1 | X)$ 와 X 사이의 관계를 modeling 할 수 있을까? (편의를 위해 binary case에서의 0/1을 사용하겠다.)

$P(X) = \beta_0 + \beta_1 X$ 라는 linear한 model을 사용했을 때, 확률값이 0보다 작거나 1보다 커지게 되는 문제가 발생했다.

그럼 이를 해결하기 위해서는? → 0과 1사이의 값만 나오도록 새롭게 modeling 해주면 된다!

$$\text{Logistic Function: } p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$



Fitting the Logistic Regression(참고)

Logistic Function을 fitting 할 때는 MLE(Maximum Likelihood Estimation)을 이용한다.

$$p(X) = P(Y = 1|X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$
$$P(Y = 0|X) = 1 - p(X) = \frac{1}{1 + e^{\beta_0 + \beta_1 X}}$$

$$P(Y_i|X_i) = p(X_i)^{Y_i} (1 - p(X_i))^{1-Y_i}$$

이를 바탕으로 Likelihood function을 구하면 다음과 같다.

$$L(\beta_0, \beta_1) = \prod_{i=1}^n p(X_i)^{Y_i} (1 - p(X_i))^{1-Y_i}$$

이제 여기에 로그만 씌우면 아래와 같아진다.

$$\ell(\beta_0, \beta_1) = \sum_{i=1}^n [Y_i \log p(X_i) + (1 - Y_i) \log(1 - p(X_i))]$$
$$= \sum_{i=1}^n \left[Y_i \log \left(\frac{e^{\beta_0 + \beta_1 X_i}}{1 + e^{\beta_0 + \beta_1 X_i}} \right) + (1 - Y_i) \log \left(\frac{1}{1 + e^{\beta_0 + \beta_1 X_i}} \right) \right]$$
$$= \sum_{i=1}^n [Y_i(\beta_0 + \beta_1 X_i) - \log(1 + e^{\beta_0 + \beta_1 X_i})]$$

미분 이후
Newton-Raphson
or Gradient Descent 이용



$$\frac{\partial \ell}{\partial \beta_0} = \sum_{i=1}^n \left[Y_i - \frac{e^{\beta_0 + \beta_1 X_i}}{1 + e^{\beta_0 + \beta_1 X_i}} \right]$$

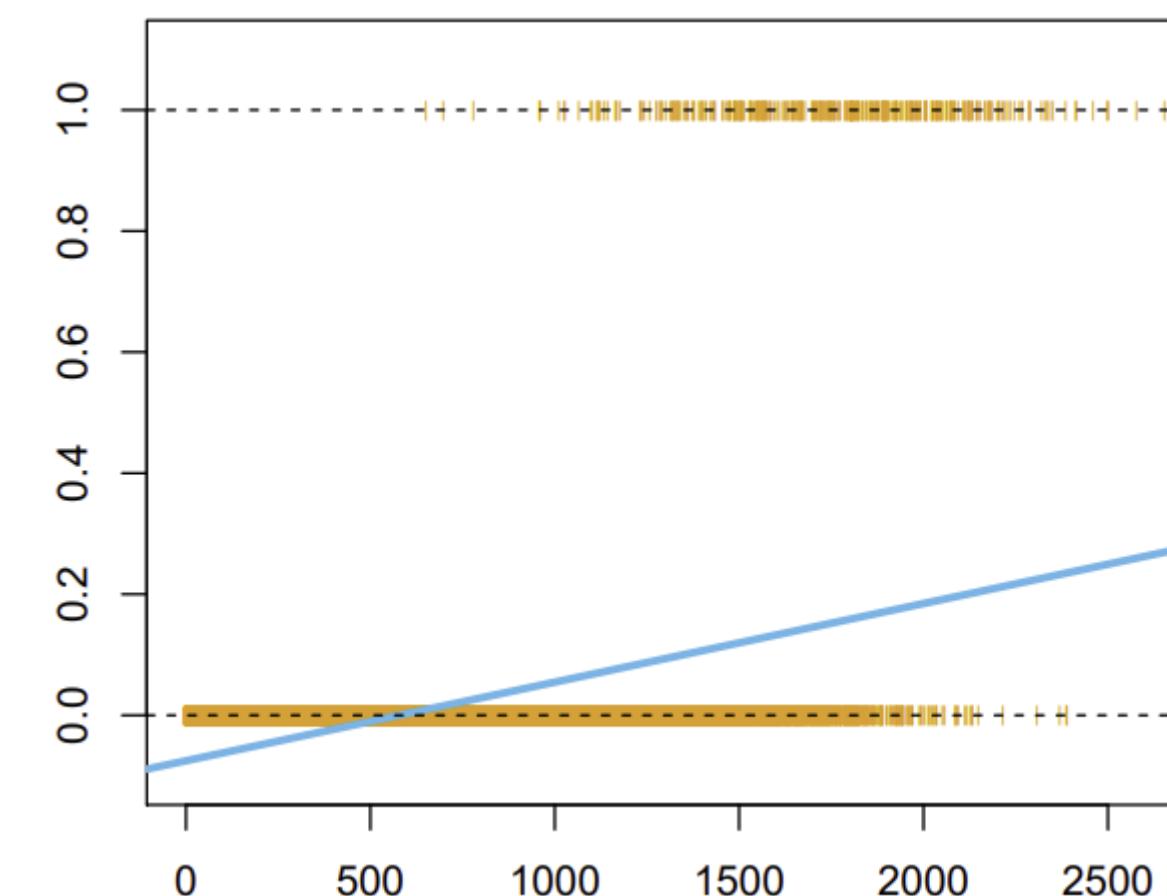
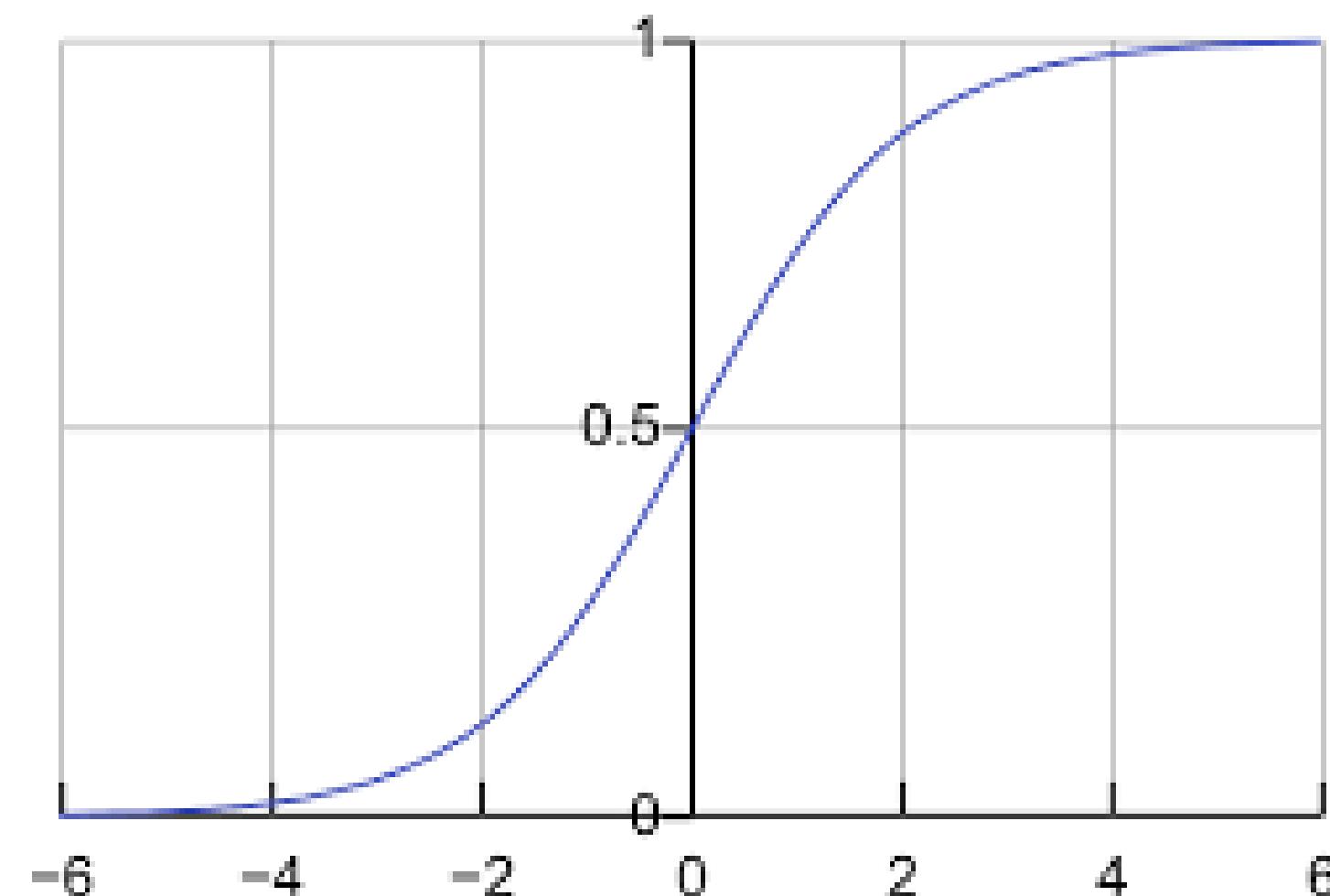
$$\frac{\partial \ell}{\partial \beta_1} = \sum_{i=1}^n X_i \left[Y_i - \frac{e^{\beta_0 + \beta_1 X_i}}{1 + e^{\beta_0 + \beta_1 X_i}} \right]$$



Sigmoid Function

Logistic function을 그려보게 되면 다음과 같은 S 형태의 그래프가 나오게 되는데, 이걸 흔히 sigmoid curve(function)이라고 부른다.

A **sigmoid function** is any **mathematical function** whose graph has a characteristic **S-shaped** or **sigmoid curve**.



이런식으로 modeling을 하게 되면, 앞서 언급했던 확률이 0보다 작아지거나 1보다 커지는 문제를 방지할 수 있다.



Odds and Log Odds(Logit)

$$\text{Odds} = \frac{p(X)}{1-p(X)}$$

: 특정 사건이 발생하지 않을 확률 대비 일어날 확률의 비율(확률과는 조금 다른 개념).

Ex) 5번의 게임 중 1번 이겼을 때, 승리에 대한 odds=0.2/(1-0.2)=1/4=0.25

5번의 게임 중 4번 이겼을 때, 승리에 대한 odds=0.8/(1-0.8)=4

$$\text{Probability} = \frac{p(X)}{p(\text{전체})} \rightarrow 5\text{번의 게임 중 1번 이겼을 때} : 0.2, 4\text{번 이겼을 때} = 0.8$$

Odds는 Probability와 다르게 **asymmetric**하다는 특징이 있다.

또한 확률이 0.5보다 커버리는 경우, 값이 1부터 무한대까지 큰 범위를 가지게 된다.

해석과 계산의 관점에서 어려움이 발생하여 대부분 로그를 취하여 사용한다.



Odds and Log Odds(Logit)

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0 + \beta_1 X}$$

↓
Log Odds or Logit

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X$$

Log Odds는 X에 대해 **linear**한 관계를 갖는다

→ **Linear Regression**처럼 해석할 수 있다.

β_1 의 의미를 살펴보면,

Linear Regression: X가 1 증가할 때, Y의 평균적인 증가량.

Logistic Regression: X가 1 증가할 때, log Odds의 증가량.

X가 1 증가할 때, log Odds의 증가량이 β_1 이므로, X가 1 증가할 때, Odds는 e^{β_1} 배 증가한다.

즉, β_1 의 부호에 따라 $p(X)$ 의 증감이 정해진다.

여기서 주의해야 할 것이 있는데, **Log Odds**가 X에 대해 선형적인 것이지

$p(X)$ 가 선형적인 것이 아니므로, 어떤 X값이 오느냐에 따라 $p(X)$ 의 변화속도가 달라진다.

결국 Logit의 궁극적인 의미는 다음과 같다.

1. 확률을 선형적으로 **modeling** + 범위를 0~1로 조정 가능.

2. 직관적인 해석 및 **Linear Regression**처럼 **Logistic Model**을 해석 가능.



Multiple Logistic Regression

Multiple Logistic Regression

: 여러 개의 독립변수를 활용해서 binary problem을 해결하겠다.

$$\log \left(\frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

$$p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}}$$

앞서 언급한 기본적인 Logistic Regression에서 predictor(X)를 추가한 일반화된 ver.이라고 생각하면 된다.

앞서 언급한 logit 등의 성질도 동일하게 이용할 수 있다.

다만, 다변수인 만큼 **confounding**이나 **multicollinearity**는 주의해야 할 필요가 있다.



Multinomial Logistic Regression

Multinomial Logistic Regression

: 이제 2개 이상의 다중 클래스 문제를 해결하겠다. ($K > 2$)

K 개의 클래스 중 하나를 기준점(**baseline**)으로 삼고,
나머지 $K-1$ 개의 logit을 modeling! → K 번째 클래스를 기준점으로 잡으면

Baseline을 설정하는 이유: 상대적인 비교를 위한 기준을 정하는 것.
즉, 다른 클래스들의 **log-odds**를 baseline 클래스와 비교해서 해석하는 방식으로 modeling 하겠다는 의미.

$$\log \left(\frac{\Pr(Y = k|X = x)}{\Pr(Y = K|X = x)} \right) = \beta_{k0} + \beta_{k1}x_1 + \cdots + \beta_{kp}x_p \quad \text{for } k = 1, \dots, K-1$$

Baseline 클래스의 logit은 항상 0 → 다른 클래스들은 baseline 클래스와의 logit 관계로 표현 가능!

Baseline을 어떤 것으로 설정하느냐에 따라 회귀 계수들은 달라질 수도 있겠지만(해석의 차이), 최종적인 예측값(확률)은 동일하게 도출됨.

(참고) baseline을 설정하지 않고 모든 클래스에 대해 대칭적인 방식으로 확률을 계산하는 방법: **Softmax Regression**
ML 분야에서 더 자주 사용된다고 하니, 관심있는 분들은 복습하시면서 찾아보세요 ㅎㅎ



Generative Models for Classification

Discriminative Model vs Generative Model

Discriminative Model(판별 모델): $P(Y | X)$ 를 직접 모델링 → X가 주어졌을 때 Y가 어떻게 결정되는가?

ex) Logistic Regression, Support Vector Machine 等

Generative Model(생성 모델): $P(X | Y)$ 와 $P(Y)$ 를 모델링하고 베이즈 정리를 통해 $P(Y | X)$ 를 간접적으로 추정

→ 각 클래스 $Y=k$ 가 데이터를 어떻게 생성하였는가의 관점에서 X 의 생성분포를 먼저 추정하고, 이를 바탕으로 역추론을 통해 분류.

Ex) LDA, QDA, Naive Bayes 등

생성모델은 다음과 같은 이유 때문에 사용된다.

1. 클래스간 분류가 뚜렷할 때: 클래스별로 데이터가 많이 떨어져 있는 경우 logistic의 회귀계수가 불안정해지는 경우가 존재하는데 이를 방지.
 2. 정규분포를 가정할 수 있고, sample수가 적을 때.
 3. 클래스가 여러 개일 때: 클래스별 분포를 잘 추정할 수만 있다면, 베이즈 정리를 통해 여러 클래스로 분류하는데 자연스럽게 확장 가능.



Generative Models for Classification

갑자기 뜬근 없이 생성이라니... 조금 이해가 안 갈 수도 있겠지만 아래와 같이 생각해보자.

분류(Classification)의 목적은 어떤 새로운 입력(X)가 주어졌을 때, 그에 어떤 클래스($Y=k$)에 속할지 예측하는 것이다.
즉, 우리가 궁극적으로 알고 싶은 것은 $P(Y = k | X = x)$ 이고, 이 확률의 의미는 $X = x$ 가 주어질 때 $Y = k$ 일 확률이 얼마인가?이다.

자 이제, 생성이라는 말에 너무 꽂히지 말고 아래 예시를 참고해보자.

오른쪽 메일들을 보면서 스팸 필터링을 어떻게 할지 생각해보면

해당 이메일이 스팸일 확률을 바로 예측하기보다는

스팸 메일에서는 어떤 단어들이 주로 들어갈까? 혹은
정상 메일에서는 어떤 단어들이 주로 들어갈까?
라는 생각 과정을 거치는 것이 더 좋아 보인다.

즉, 단어들의 분포가 어떨까?를 먼저 모델링 하는 것이 더 직관적이다.
이를 단어들로 인해 메일이 생성된다~ 정도로 이해하면 납득이 되 것이다.

정리하자면, 생성모델의 관점에서 우리는 이메일이 스팸일 확률을 직접 예측하는 것이 아니라,
스팸/정상 메일에서 단어들이 어떻게 등장하는지 모델링하고
→ 이를 바탕으로 새로운 메일이 도착했을 때, 스팸일 확률을 계산하는 것이다.

<input type="checkbox"/>			빗썸	[빗썸] 예치금 이용료 정기 지급 안내
<input type="checkbox"/>			지모	지모: 현재 인기 많은 제품을 알려드려요. 🔥
<input type="checkbox"/>			교보문고	(광고)[교보문고]책과 강연을 한번에 보라 VORA 1월 추천도서!
<input type="checkbox"/>			배달의민족	배민클럽 정기결제 예정일을 안내드려요.
<input type="checkbox"/>			POLARIS Office	플라리스 오피스 개인정보이용내역 안내
<input type="checkbox"/>			토스뱅크	예금거래기본약관 개정 안내
<input type="checkbox"/>			아시아나항공	(광고) 지민구 회원님의 1월 마일리지 현황 및 새로운 소식을 안내 드립니다
<input type="checkbox"/>			Riot Games	라이엇 게임 계정 제재 안내
<input type="checkbox"/>			베ッセルホテルズ	【重要】[ベッセルクラブ]ポイントキャッシュバック利用方法変更
<input type="checkbox"/>			SSG.COM	[SSG.COM] 휴면 회원 정책 변경 안내
<input type="checkbox"/>			교보문고	(광고)[교보문고] 경제전문가 박정호 교수 <2025 트럼프의 귀환, 정책 변화와 트렌드> 강연 안내
<input type="checkbox"/>			Instagram	Samsung GT-I9300 Threads에서 새로운 Instagram 로그인 발생



ESC

Generative Models for Classification

그럼 앞서 이야기했던 것들을 식으로 정리하면 다음과 같다.

Y 가 K 개의 고유한 클래스를 가질 수 있다고 할 때

$\pi_k = P(Y = k)$: 사전확률(**prior probability**), 무작위로 선택된 관측지가 k 번째 클래스에 속할 확률.

$f_{k(x)} \equiv P(X | Y = k)$: 클래스 k 에서 X 의 확률밀도함수(**pdf**), 클래스 k 에서 특정값 X 가 나타날 확률.

위 두가지 확률을 통해 베이즈 정리를 이용하면 다음과 같은 사후확률(**posterior probability**)을 도출 가능하다.

$$p_k(x) = P(Y = k | X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)}$$

Bayes Classifier: 이 사후확률 중 가장 높은 k 를 선택하는 방식. 이론상 가장 낮은 **error rate**를 가진다.

$$\hat{y} = \arg \max_k \Pr(Y = k | X = x) = \arg \max_k \pi_k f_k(x)$$

그러나 현실적으로 $f_k(x)$ 를 정확히 알기 쉽지 않으므로...(실제 데이터의 분포는 찾기 힘듦, 데이터의 부재, 고차원 이슈 등)

이를 근사시키기 위해 생성모델을 사용한다!



Linear Discriminant Analysis(LDA) for p=1

설명변수가 딱 하나만 존재하고, $f_k(x)$ 를 정규분포로 가정하자(클래스별로 모두 같은 분산을 가진다고 가정).

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2\right) \longrightarrow p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^K \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)}$$

우리의 목표는 가장 큰 $p_k(x)$ 값을 가지는 k를 선택하는 것이다. 이를 위해 $p_k(x)$ 에 로그를 씌우고 식 정리를 하면 다음과 같은 판별 함수(**discriminant function**)을 얻을 수 있다.

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

해당 판별 함수가 x에 대해 **linear**한 형태를 가지므로 LDA라고 부르는 것!

이제 이걸 바탕으로 결정 경계(**decision boundary**)를 구하게 되는데, 두 클래스의 판별 함수가 동일한 점에서 결정된다!

$$\begin{aligned} \log p_k(x) &\propto \log \pi_k + \log f_k(x) \\ \log f_k(x) &= \log \left(\frac{1}{\sqrt{2\pi}\sigma^2} \exp\left(-\frac{(x - \mu_k)^2}{2\sigma^2}\right) \right) \\ &= -\frac{(x - \mu_k)^2}{2\sigma^2} - \frac{1}{2} \log(2\pi\sigma^2) \\ \delta_k(x) &= x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k) \end{aligned}$$



Linear Discriminant Analysis(LDA) for p=1

예를 들어 K=2(클래스가 두 개, Y=1, 2), $\pi_1 = \pi_2$ 인 상황에서의 decision boundary를 생각해보면,

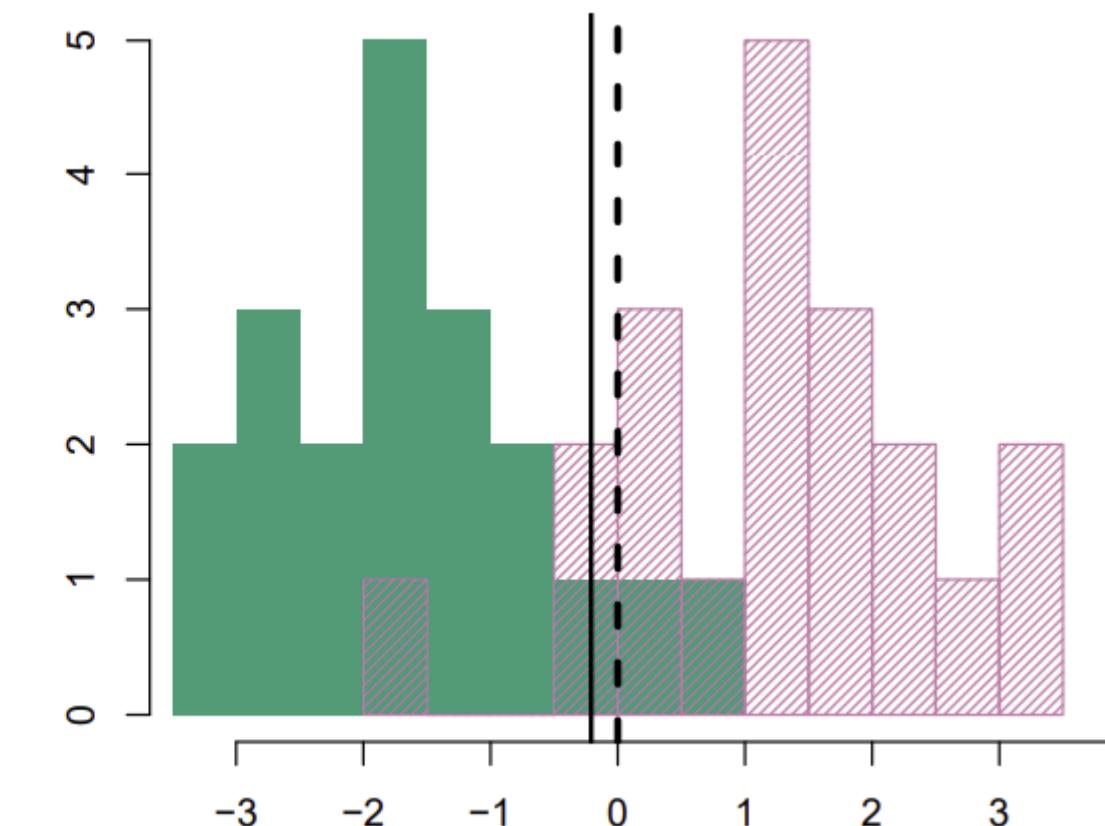
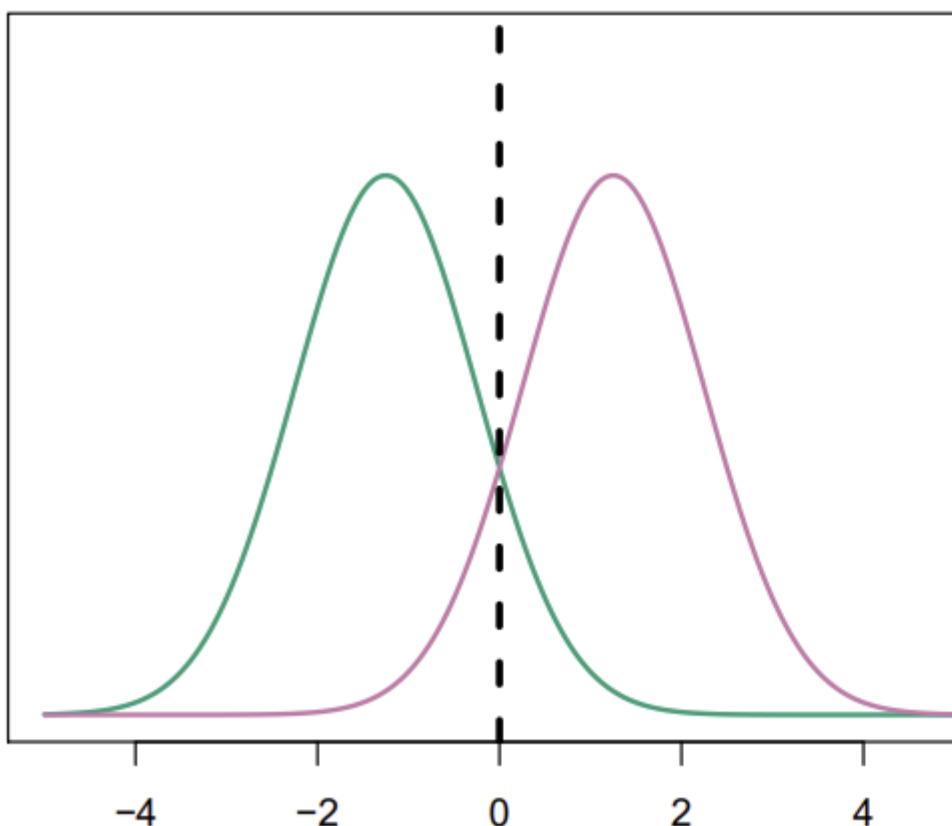
$$\delta_1(x) = \delta_2(x) \rightarrow x \cdot \frac{\mu_1}{\sigma^2} - x \cdot \frac{\mu_2}{\sigma^2} = \frac{\mu_1^2}{2\sigma^2} - \frac{\mu_2^2}{2\sigma^2} + \log\left(\frac{\pi_2}{\pi_1}\right)$$

$$x \cdot \frac{\mu_1}{\sigma^2} - x \cdot \frac{\mu_2}{\sigma^2} = x \cdot \frac{\mu_2}{\sigma^2} - x \cdot \frac{\mu_1}{\sigma^2} + \log\left(\frac{\pi_2}{\pi_1}\right) \rightarrow x(\mu_1 - \mu_2) = \frac{\mu_1^2 - \mu_2^2}{2} + \sigma^2 \log\left(\frac{\pi_2}{\pi_1}\right) \rightarrow x = \frac{\mu_1^2 - \mu_2^2}{2(\mu_1 - \mu_2)} + \frac{\sigma^2}{\mu_1 - \mu_2} \log\left(\frac{\pi_2}{\pi_1}\right)$$

$$x = \frac{\mu_1 + \mu_2}{2}$$

(두 클래스의 사전확률이 같다면)

$\mu_1 = -1.25$, $\mu_2 = 1.25$ 이고 $\sigma_1^2 = \sigma_2^2 = 1$ 인 경우 decision boundary는 $x=0$ 인 경우가 된다.



Linear Discriminant Analysis(LDA) for p=1

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

현실에서는 클래스별 평균 μ_k 와 분산 σ^2 에 대해 정확하게 알 길이 없으므로... π_k 도 마찬가지...
→ 추정하고 그 값을 대입해서 사용한다.

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

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

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

평균과 분산은 MLE로 추정하고, 사전확률의 경우 샘플 비율로 단순 추정한다(사실 이것도 MLE긴 하다).

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



Linear Discriminant Analysis(LDA) for p>1

이제 LDA를 다변량 데이터로(p>1) 확장해보자.

각 클래스별로 다변량 정규분포를 따른다고 가정하고 앞의 내용과 같이 공분산 행렬은 모두 같다.

$X \sim N_p(\mu, \Sigma)$ where μ is $p \times 1$ vector and Σ is $p \times p$ variance-covariance matrix.

공분산 행렬이 모두 같다는 전제하에 각 클래스 k의 pdf는 다음과 같이 정의된다. $f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_k)^T \Sigma^{-1} (x - \mu_k)\right)$

아까와 같은 방법으로 판별 함수를 구해보면 다음과 같다. $\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$
벡터와 행렬을 이용한 p차원의 일반화 ver.이라고 생각하면 되고, 여전히 x에 대해 linear함을 확인 가능하다.

이를 통해 decision boundary를 구해보면 서로 다른 k와 l에 대해 다음을 비교하면 된다!

$$\delta_k(x) = \delta_l(x)$$
$$x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k) = x \cdot \frac{\mu_l}{\sigma^2} - \frac{\mu_l^2}{2\sigma^2} + \log(\pi_l)$$

$$x = \frac{\mu_k^2 - \mu_l^2}{2(\mu_k - \mu_l)} + \frac{\sigma^2}{\mu_k - \mu_l} \log\left(\frac{\pi_l}{\pi_k}\right)$$

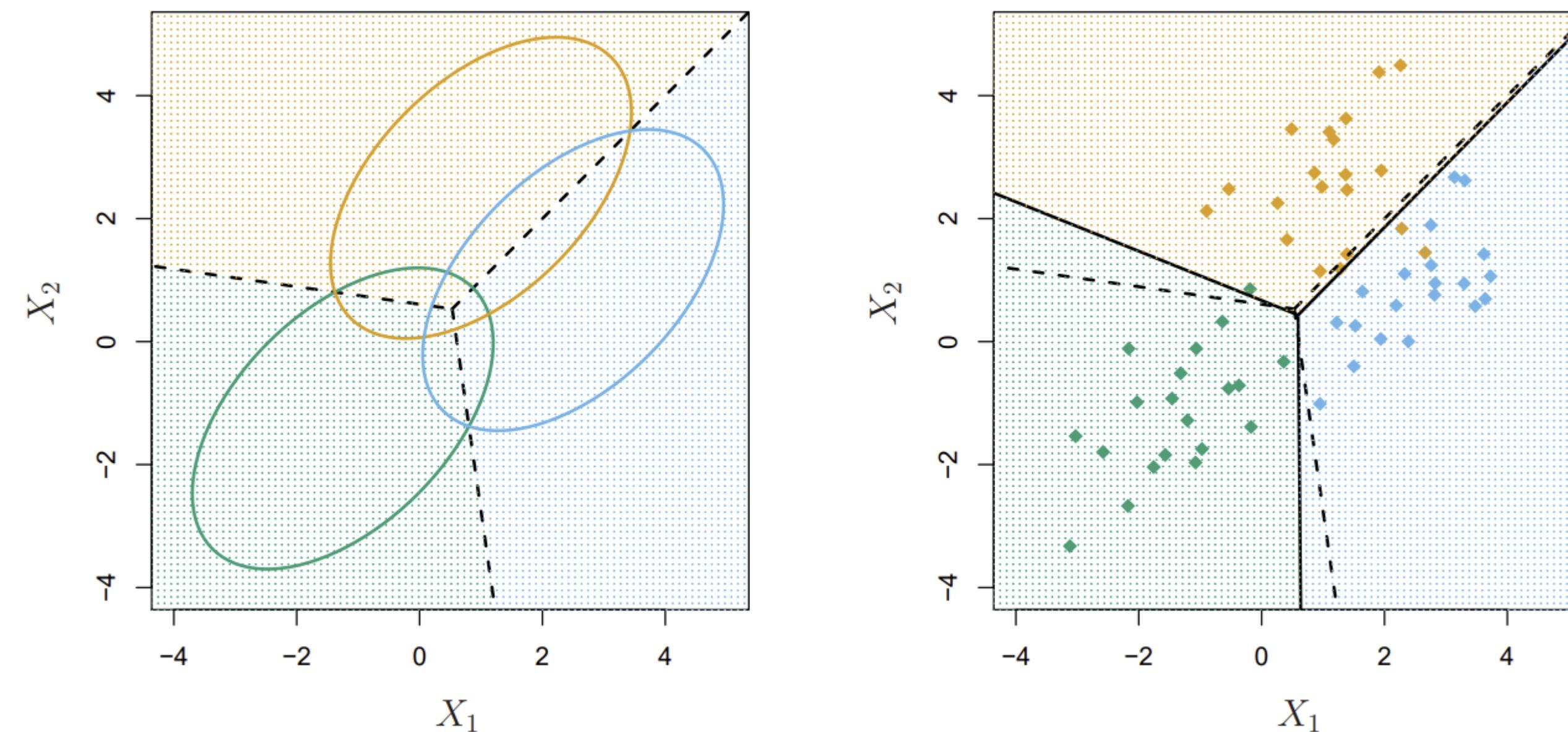
참고로 K개의 class가 있다면 각 클래스 쌍마다 경계가 생기므로 $K(K-1)/2$ 개의 경계가 생긴다.

이후에는 동일하게 estimate를 찾아서 대입하면 된다.



Linear Discriminant Analysis(LDA) for $p>1$

$K=3$ 이고 $p=2$ 인 경우를 생각해보면 다음과 같다.



Quadratic Discriminant Analysis(QDA)

자 그럼 이제 LDA와 비슷한 가정에서 출발하되, 조금의 변주를 줘보자.

LDA는 각 클래스가 정규분포를 따르고 같은 분산(공분산 행렬)을 가진다고 가정하는데, 여기서 분산과 관련한 가정을 없애자. 즉, 각 클래스마다 고유한 **covariance matrix**가 있다는 것이다.

그렇다면 우리의 가정은 다음과 같이 바뀌고, 이에 따른 판별 함수는 아래와 같아진다.

$$\begin{aligned} X \sim N_p(\mu_k, \Sigma_k) \quad \delta_k(x) &= -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) - \frac{1}{2} \log |\Sigma_k| + \log \pi_k \\ &= -\frac{1}{2}x^T \Sigma_k^{-1} x + x^T \Sigma_k^{-1} \mu_k - \frac{1}{2}\mu_k^T \Sigma_k^{-1} \mu_k - \frac{1}{2} \log |\Sigma_k| + \log \pi_k \end{aligned}$$

X가 LDA와 달리 **quadratic form**으로 나타나기 때문에 **QDA**라고 부른다.



LDA vs QDA: Bias-Variance Tradeoff

LDA와 QDA의 가장 큰 차이점은 공분산 행렬이 같은지 아닌지 여부이다.

우선 공분산 행렬을 추정하는데 필요한 parameter의 수에 대해 생각해보자.

LDA: 대각 원소(분산) p 개 + 비대각원소(공분산) $\binom{p}{2} = \frac{p(p - 1)}{2}$ 개 = $p(p+1)/2$ 개

QDA: LDA $\times K$ 개(클래스의 개수)

설명 변수와 클래스의 개수가 많으면 많을수록 **QDA**에서는 엄청나게 많은 **parameter**를 추정해야 한다.

$$\text{MSE} = (\text{Bias}[\hat{f}(X)])^2 + \text{Variance}[\hat{f}(X)] + \sigma^2$$

Variance: 모델을 학습할 때 다른 학습 데이터(train set)를 사용하면 \hat{f} 가 얼마나 변동하는가 → 모델이 데이터에 얼마나 민감한가?

Bias: 실제 문제를 단순한 모델로 근사할 때 발생하는 오차 → 모델이 얼마나 단순화되어 있는가?

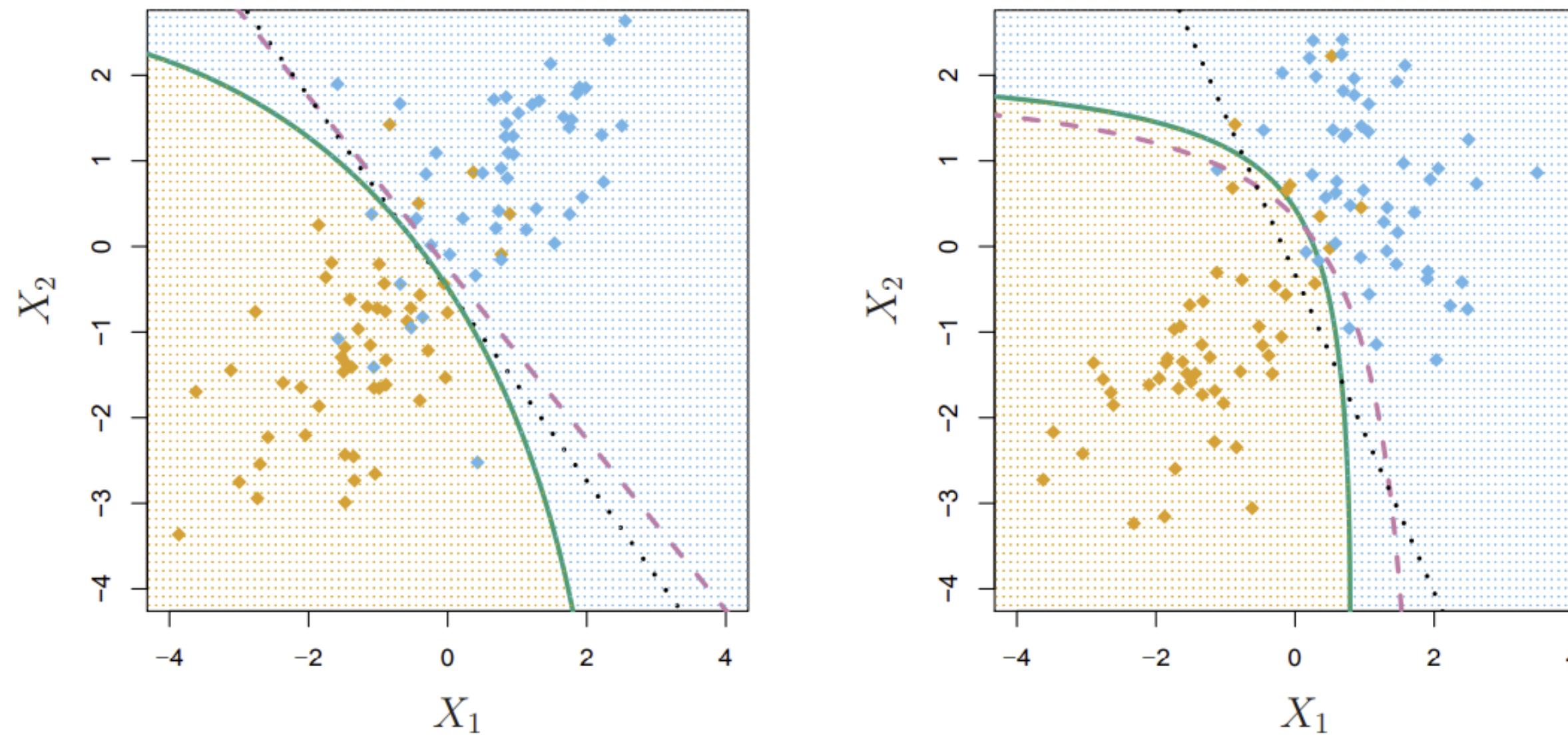
Bias-Variance Tradeoff:

유연한 모델(복잡한 모델) → **Bias↓, Variance↑(QDA)** vs 제약이 많은 단순한 모델 → **Bias↑, Variance↓(LDA)**



LDA vs QDA: Bias-Variance Tradeoff

LDA	QDA
추정해야 할 parameter 개수 적음 → 낮은 분산	추정해야 할 parameter 개수 많음 → 높은 분산
공분산 행렬 가정이 틀릴 경우 → 높은 편향	유연하고 비선형적인 boundary → 낮은 편향
데이터가 적을 때 좋은 성능	데이터가 많을 때 충분히 학습하면 좋은 성능



Naïve Bayes

결국 생성모델에서 최종적인 목표는 사후확률을 구하는 것이었고, 식을 다시 보면 아래와 같다.

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)}$$

이 과정에서 $f_k(x)$ 를 구하는 과정이 매우 어렵다고 했고 이를 위해 여러 가지 방법을 이용했다.

Naïve Bayes는 LDA&QDA와는 조금 다른 가정을 토대로 추정을 진행한다.

Within the k th class, the p predictors are independent → 모든 설명 변수가 클래스 k 에서 독립이다.

즉, $f_k(x) = f_{k1}(x_1) \times f_{k2}(x_2) \times \cdots \times f_{kp}(x_p)$, where f_{kj} is the pdf of the j th predictor among observations in the k th class.

이 가정 덕분에 **joint distribution**을 고려할 필요가 없어진다!

→ 현실적으로 변수 간의 관계를 다변량 정규분포로 잘 표현할 수 없는 경우도 많고, **공분산 행렬을 추정하는 데 많은 데이터가 필요**.

→ 독립성 가정을 통해 계산을 현저히 줄이고, 적은 데이터만으로도 modeling 가능. $\Pr(Y = k|X = x) = \frac{\pi_k \times f_{k1}(x_1) \times f_{k2}(x_2) \times \cdots \times f_{kp}(x_p)}{\sum_{l=1}^K \pi_l \times f_{l1}(x_1) \times f_{l2}(x_2) \times \cdots \times f_{lp}(x_p)}$

현실적으로 설명 변수끼리 독립이라는 가정을 그대로 적용하기에는 무리가 있지만, 실제로 성능이 꽤 괜찮은 경우가 많다고 한다.

1. 적은 데이터를 이용해야 할 때 (특히 $n < p$ 고차원의 경우): **joint distribution**을 직접 추정하기 어려우므로 Naïve Bayes가 좋은 대안!
2. **Bias-Variance Tradeoff**의 관점: 독립을 가정하게 되면 **Bias**는 당연히 증가, 다만 **variance** 감소. 1과 관련 **overfitting** 방지.



03

Support Vector Machines

What Is a Hyperplane?

Hyperplane(초평면): p차원 공간에서 초평면은 차원이 p-1인 평평한 **affine** 부분공간이다(**affine flat subspace**).
(affine subspace는 linear subspace와 달리 원점을 지날 필요가 없다 정도로만 이해하자.)

초평면은 다음과 같은 식으로 정의된다: $\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p = 0$
p차원의 벡터 X가 해당 정의를 만족하면, 그 X는 초평면 상의 점이다.

식만 봐서는 무슨 말인지 감이 잡히질 않으니... 간단하게 p=2인, 즉 2차원의 공간에서의 hyperplane을 생각해보자.

$$1 + 2X_1 + 3X_2 = 0$$

: 2차원 공간에서의 초평면은 차원이 1인 affine 부분공간이다 → 원점을 지나지 않는 직선!

2차원 벡터 X가 $1 + 2X_1 + 3X_2 = 0$ 을 만족한다면, 오른쪽 그림의 선 위에 위치할 것이다.

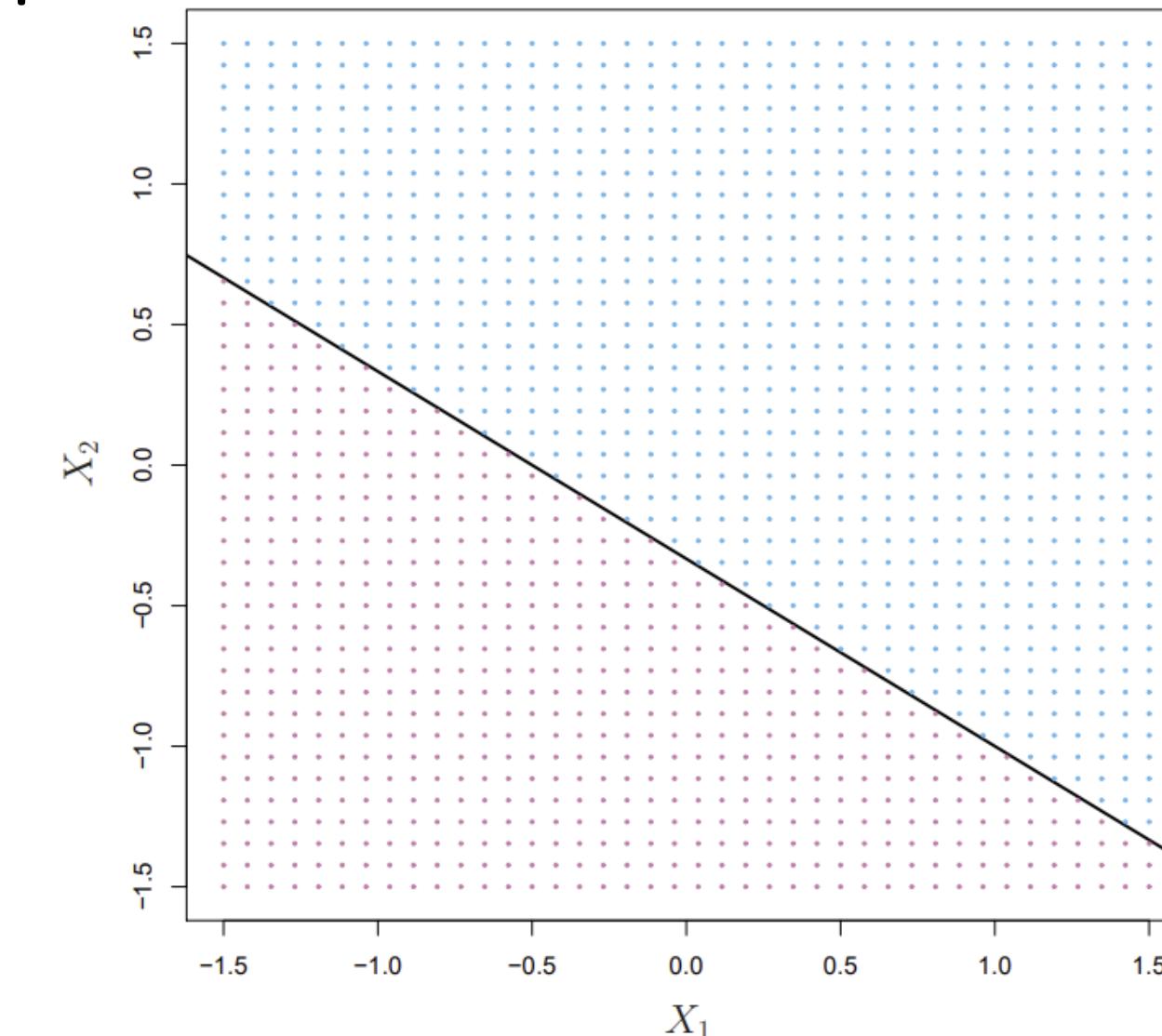
그럼 정의를 조금 비틀어서 다음을 생각해보자.

$$1 + 2X_1 + 3X_2 > 0 \text{ or } 1 + 2X_1 + 3X_2 < 0$$

위의 식들은 X가 해당 초평면의 한 쪽에 놓인다는 것을 의미한다.

파란색 영역의 경우는 0보다 큰 경우를, 보라색 영역의 경우는 0보다 작은 경우를 의미한다.

따라서, 우리는 이제 초평면은 p차원 공간을 두 개로 나눈다고 생각할 수 있다.



What Is a Hyperplane?

Figure 6.1 with the plane

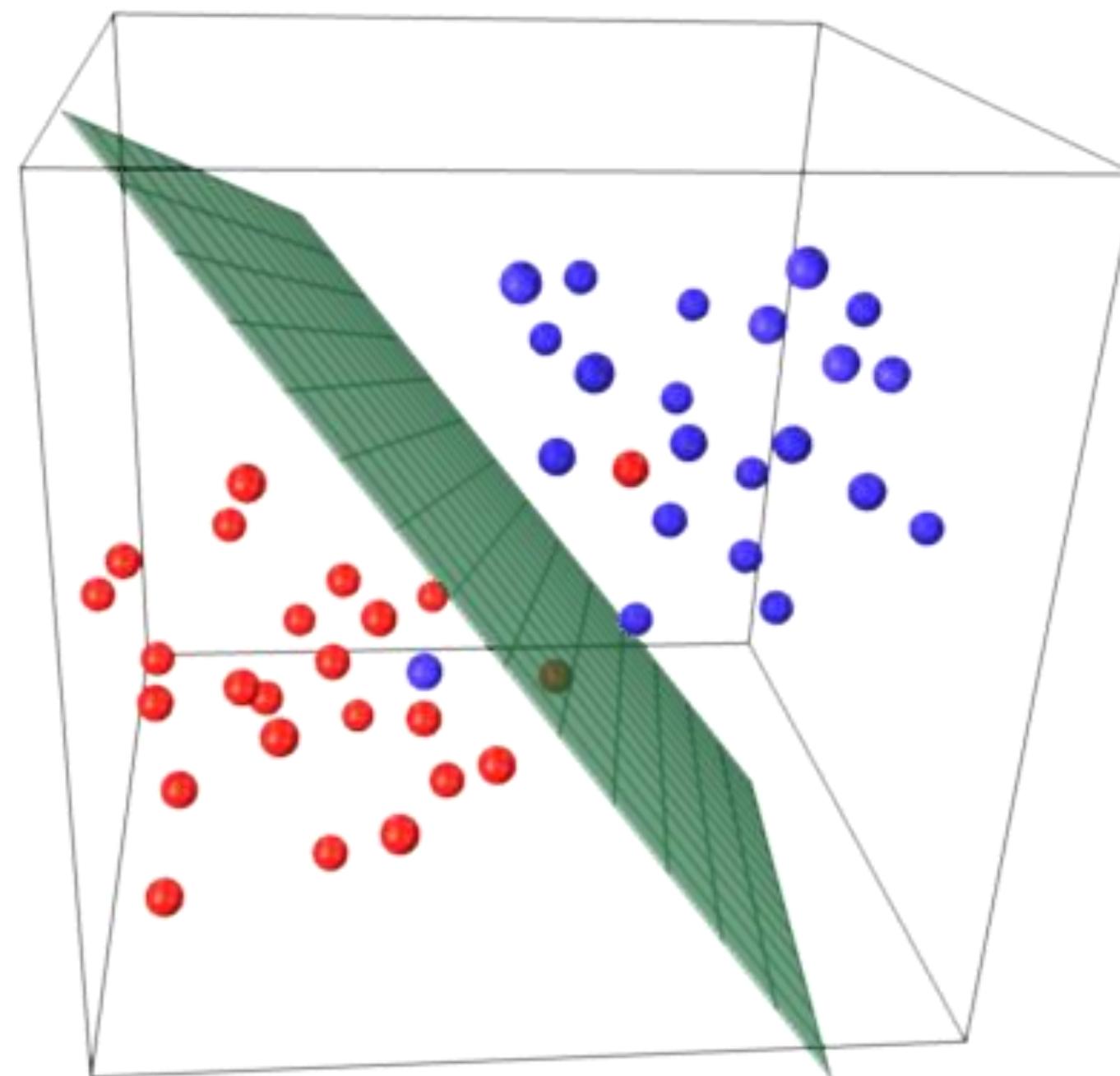
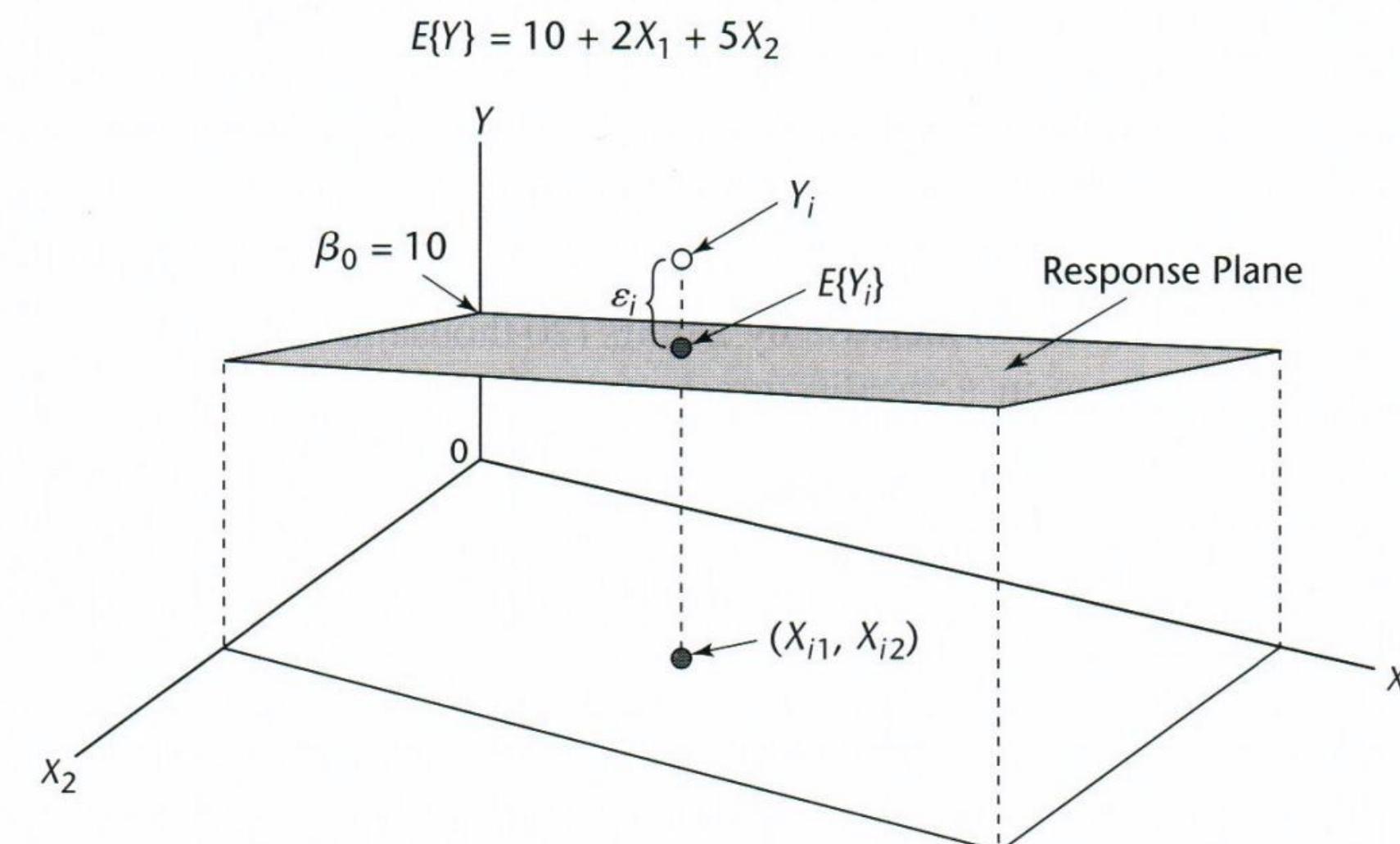


FIGURE 6.1
Response
Function is a
Plane—Sales
Promotion
Example.

$$E(Y) = 10 + 2X_1 + 5X_2$$



Classification Using a Separating Hyperplane

그럼 이제 **hyperplane**이 특정 공간을 두 개로 분리한다는 점을 이용하여 어떤 **분류기(classifier)**을 만들 수 있을지 생각해 보자.

P차원 공간에 n 개의 train data set이 존재하고 이를 $X(n \times p)$ 라는 matrix로 표현하면 다음과 같다.

$$x_1 = \begin{pmatrix} x_{11} \\ \vdots \\ x_{1p} \end{pmatrix}, \dots, x_n = \begin{pmatrix} x_{n1} \\ \vdots \\ x_{np} \end{pmatrix}$$

각 데이터는 -1 혹은 1이라는 두 개의 클래스(Y)에 포함된다.

$\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p = 0$ 인 초평면에 대해 0보다 크면 $Y=1$ 로, 작다면 $Y=-1$ 로 라벨을 붙이게 되면...

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} > 0 \text{ if } y_i = 1,$$

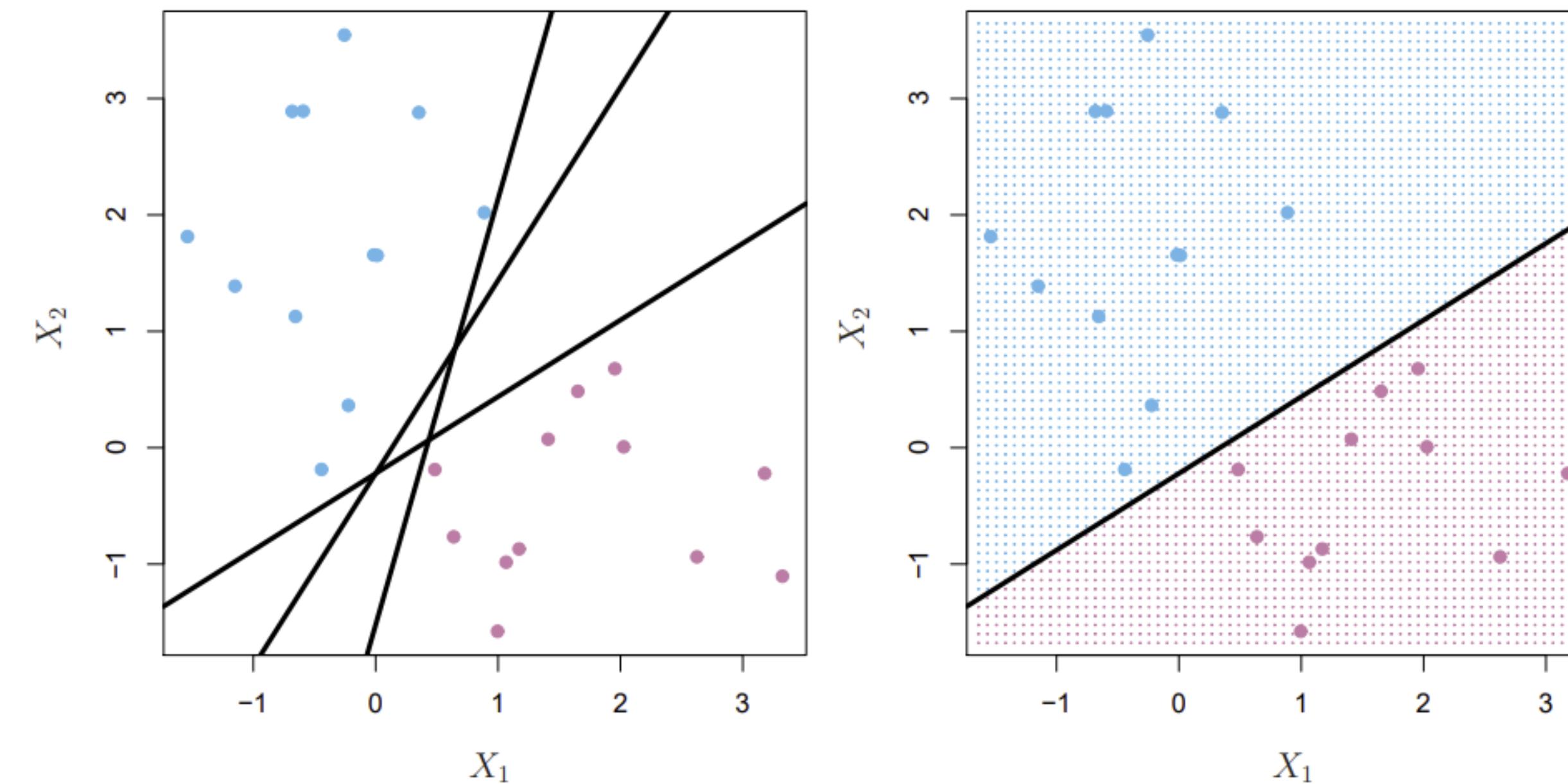


$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) > 0 \quad \text{for all } i=1 \sim n$$

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} < 0 \text{ if } y_i = -1.$$



Classification Using a Separating Hyperplane



초평면이 공간을 두 개로 나누는 이 방식은, 선형적인 **decision boundary**로 이어짐을 알 수 있다.

그런데 하나 큰 문제가 있다. 데이터가 완벽하게 나눠지는 경우, 이를 구분하는 **separating hyperplane**은 무수히 많이 존재한다.

적당한 hyperplane 하나를 찾게 되면 약간만 이동하거나 회전시켜도 여전히 데이터를 완벽하게 분리할 수 있기 때문이다.
그렇다면 어떤 초평면을 선택하는 것이 가장 합리적일까?



Maximal Margin Classifier

정답은 바로 **maximal margin hyperplane**(최대 마진 초평면)을 고르는 것이다!

margin(마진): hyperplane과 가장 가까운 data와의 거리.

즉 우리는 모든 data에서 가장 가까운 점(Perpendicular Distance)이 가장 먼 초평면을 고르면 된다.

Maximal Margin Classifier: maximal margin hyperplane을 이용한 classifier.

분류 방법: $f(x^*) = \beta_0 + \beta_1 x_1^* + \cdots + \beta_p x_p^*$ 의 부호를 이용해 양수면 클래스 1, 음수면 -1로 분류.

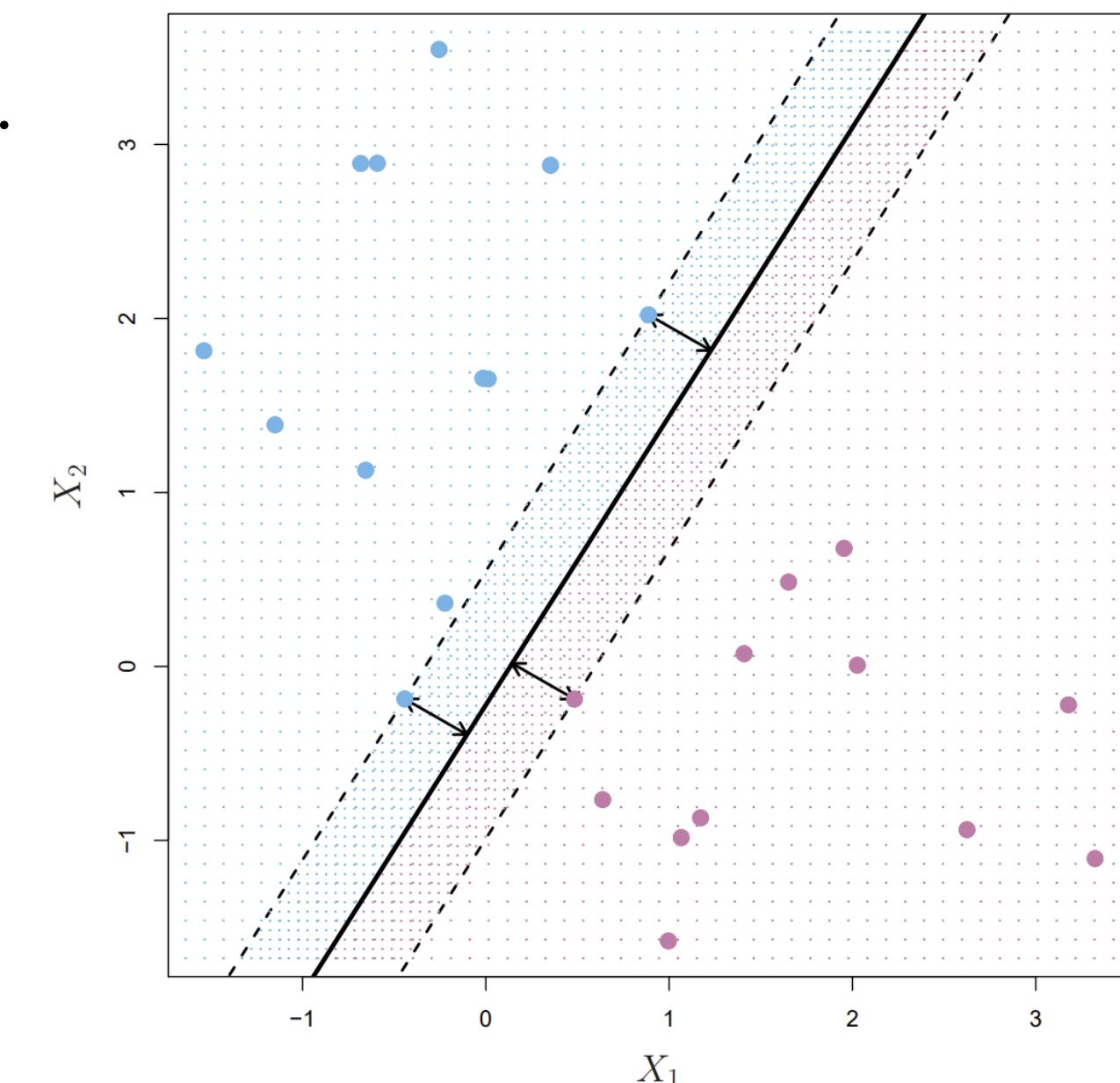
Support Vector: margin을 결정하는 train data 중 **hyperplane**에 가장 가까운 점들.

이 점들이 약간 이동하면 maximal margin hyperplane도 이동할 것!

→ 이러한 의미에서 “**support**”하기 때문에 이렇게 부름.

Maximal margin hyperplane은 **support vector**에만 의존적이다.

즉, support vector가 아닌 다른 데이터는 hyperplane 결정에 영향을 주지 않는다.



Maximal Margin Classifier

그렇다면 왜 margin이 최대가 되는 classifier을 골라야 하는 것일까?

여러 가지 이유가 있지만 간단하게 몇 가지만 소개해보면 아래와 같다.

1. **Generalization(일반화 문제)**: margin이 클수록 새로운 데이터에 대한 예측 성능(일반화 성능)이 좋아질 가능성이 높다.
→ Train set 뿐만 아니라 새로운 데이터에 대해서도 더 안정적인 성능을 낼 수 있다.

직관적으로 생각해볼 때

margin이 크면 → decision boundary가 두 클래스 사이에서 여유 있게 설정된다.

Margin이 작으면 → decision boundary가 너무 가깝게 설정되어 작은 변화에도 영향을 많이 받는다 → **overfitting** 가능성 높아짐.
즉, 더 큰 margin을 갖는 decision boundary가 데이터에 대한 작은 변화에도 더욱 견고하다.

2. **Robustness(강건성)**: 클래스 분리를 위해 hyperplane을 정할 때, margin이 크면 데이터가 변동해도 decision boundary가 쉽게 변하지 않는다. → **Noise**에 강하다.

margin이 작으면

→ 데이터에 작은 변화가 생기더라도 decision boundary가 크게 이동할 가능성이 크다 .

→ 새로운 data가 조금만 움직여도 잘못 분류될 확률이 높아진다.

→ margin이 최대화된 hyperplane이 가장 안정적이고 **robust한 classifier**의 역할을 할 수 있다.



Construction of the Maximal Margin Classifier

사실, maximal margin classifier은 다음 최적화 문제의 **solution**이다.

$$\underset{\beta_0, \beta_1, \dots, \beta_p, M}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \sum_{j=1}^p \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \dots, n.$$

$$y_i = 1 \text{인 경우} \rightarrow \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} \geq M \text{ (양수)}$$

우선 두 번째 제약조건은 모든 데이터가 올바른 클래스에 분류되도록 보장하는 조건이다.
각 데이터가 올바른 클래스에 속하면서, 최소한 **margin M**만큼 떨어져 있도록 보장해준다.

첫 번째 제약 조건은 사실 hyperplane 자체와 관련한 제약은 아니지만
hyperplane이 여러 개 존재하는 상황을 방지하기 위한 **scaling term** 정도로 생각하면 된다.

if $\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} = 0$ defines a hyperplane, then so does
 $k(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) = 0$ for any $k \neq 0$.

최종적으로 각 **data**에서 **hyperplane**까지의 거리를 생각해보면

$$\frac{|\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}|}{\sqrt{\sum_{j=1}^p \beta_j^2}}$$



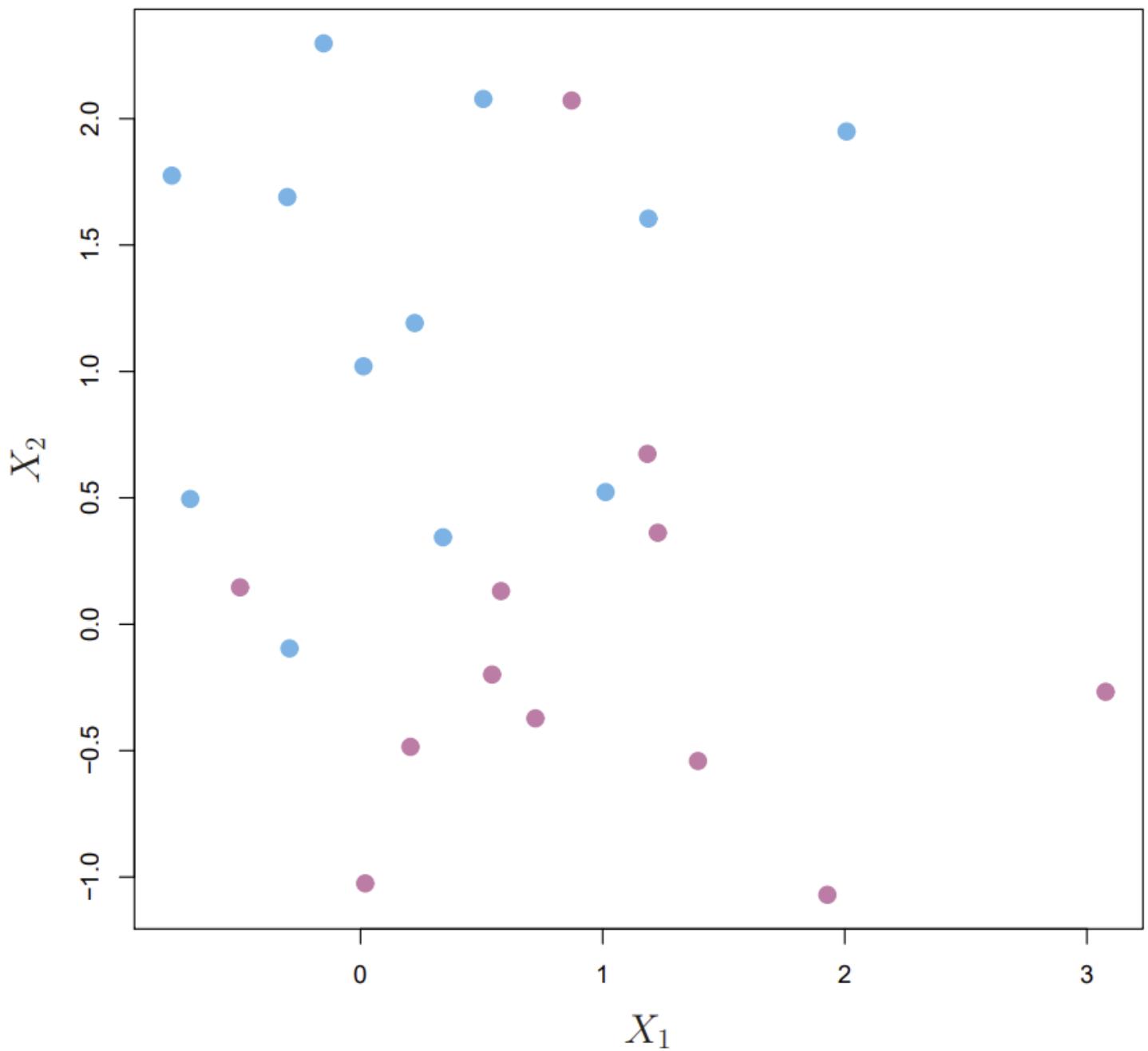
$$|\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}|$$

결국 **margin**의 최대화
= **hyperplane**과 가장 가까운 **data**와 거리 최대화



The Non-separable Case

그러나 maximal margin classifier가 완벽한 것만은 아니다.



다음과 같이 데이터가 분포하는 경우, 이를 완벽하게 구분하는 hyperplane을 만들 수 없다.

이를 해결하기 위해 margin의 개념을 확장하여 **soft margin**을 사용해 문제를 해결해보자.

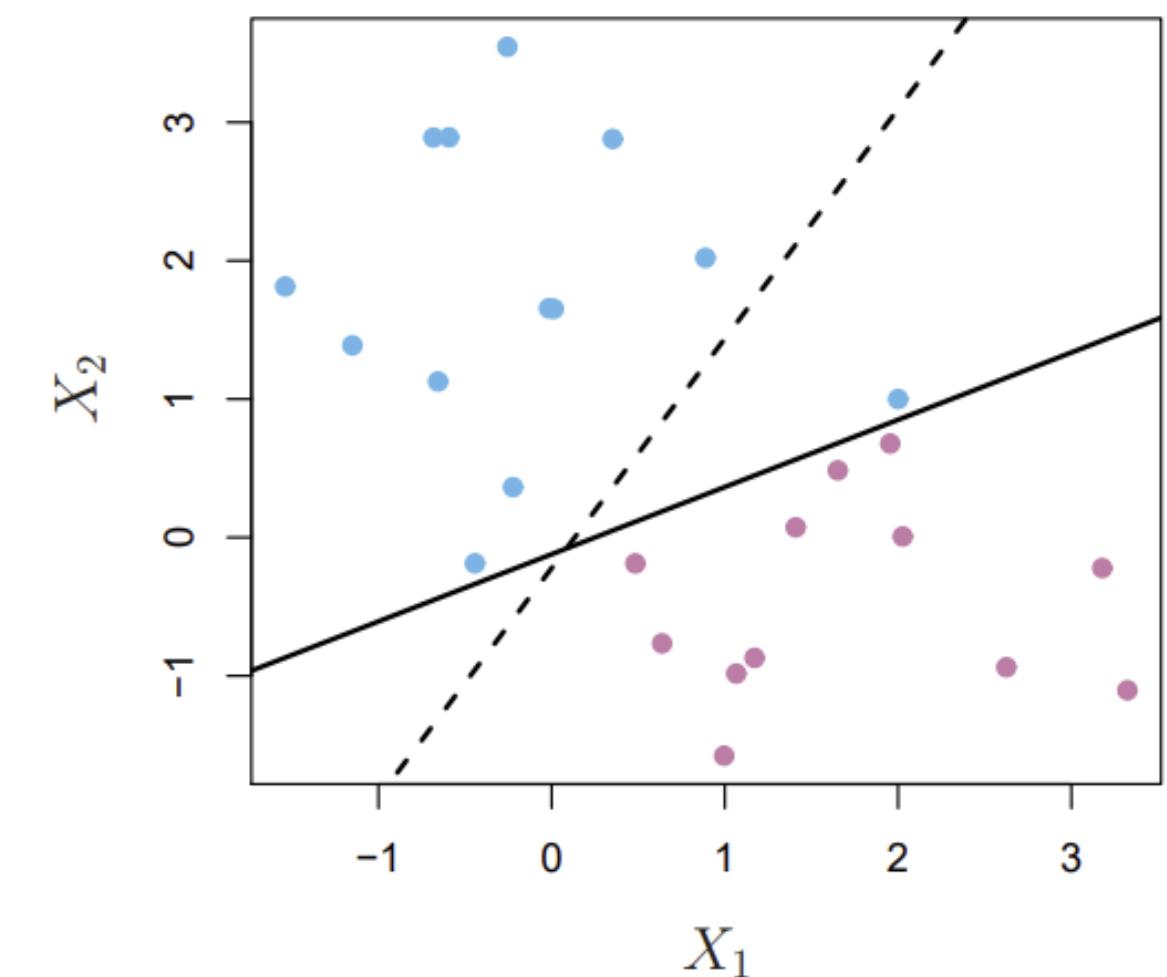
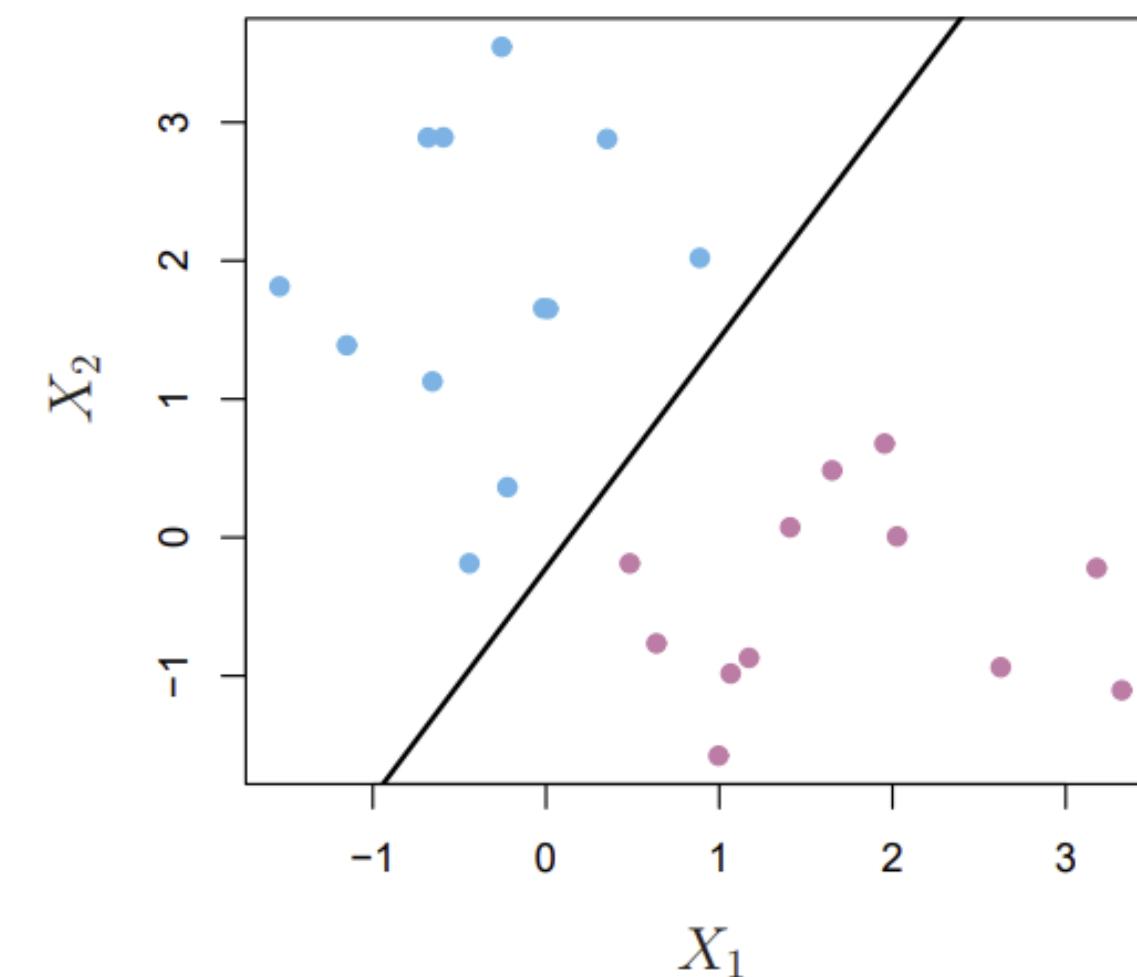


Support Vector Classifier(Soft Margin Classifier)

앞선 상황과 더불어 오른쪽 상황과 같은 문제도 생각해볼 수 있다.

모든 data를 잘 분리시키는 hyperplane을 잘 설정해도 개별 관측지에 민감하여 하나의 관측지가 추가되면서 급격한 변화가 일어날 수도 있다.

새로 얻은 hyperplane은 **margin**이 굉장히 작게 나타나고, maximal margin classifier가 개별 관측지에 민감하다는 사실은 **train data**를 **overfitting**할 수 있다는 사실을 시사한다.



그렇다면 개별 관측지에 대해 **robust**하고, 대부분(**모든x**) **train data**들을 더 잘 분류할 수 있는 새로운 classifier를 생각해보자.

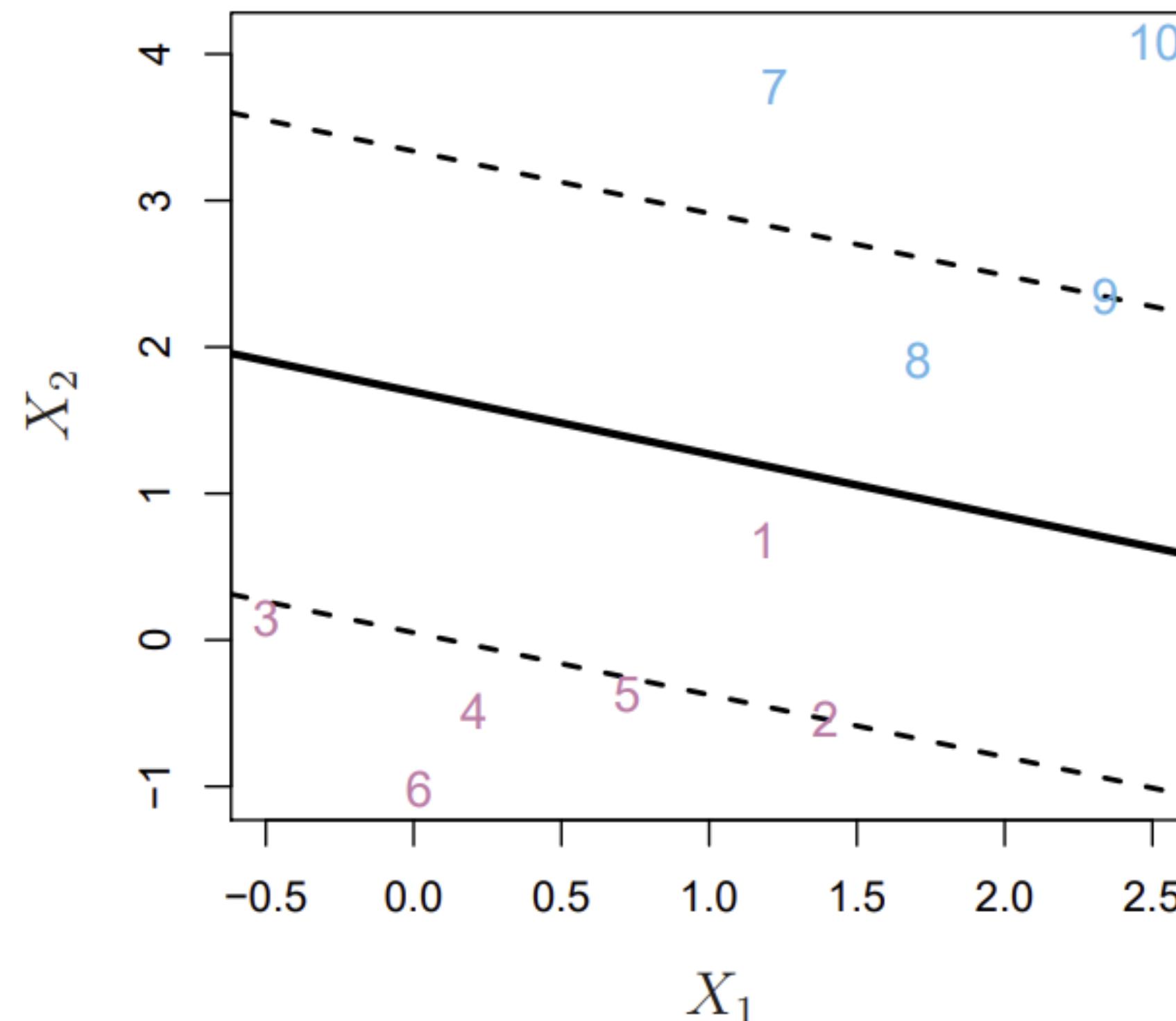
즉, 몇 개 정도는 잘못 분류하더라도, 나머지 관측지들을 더 잘 분류할 수만 있다면 의미가 있을 것이다.

이것이 **Support Vector Classifier(Soft Margin Classifier)**의 작동 원리이다.

Soft Margin의 의미: 일부 data에 대해 **margin**이 위반(내부에 존재하거나, 분류되지 못하거나).



Support Vector Classifier(Soft Margin Classifier)



Support Vector Classifier(Soft Margin Classifier)

Support Vector Classifier: margin을 최대화 하면서도 적절한 수준에서 오차를 조금 허용하여 최적의 hyperplane을 찾는 모델.

이는 다시 다음과 같은 최적화 문제로 이어질 수 있는데...

$$\begin{aligned} & \underset{\beta_0, \beta_1, \dots, \beta_p, \epsilon_1, \dots, \epsilon_n, M}{\text{maximize}} && M \\ & \text{subject to} && \sum_{j=1}^p \beta_j^2 = 1, \\ & && y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i) \\ & && \epsilon_i \geq 0, \quad \sum_{i=1}^n \epsilon_i \leq C, \end{aligned}$$

Slack Variable ϵ_i
: 완벽하게 분리되지 않는 데이터 처리를 위한 허용 오차 변수
 $\epsilon = 0 \rightarrow$ 잘 분류됨.
 $0 < \epsilon < 1 \rightarrow$ margin 안에 있지만 올바르게 분류됨.
 $\epsilon > 1 \rightarrow$ hyperplane 반대 방향으로 아예 잘못 분류됨

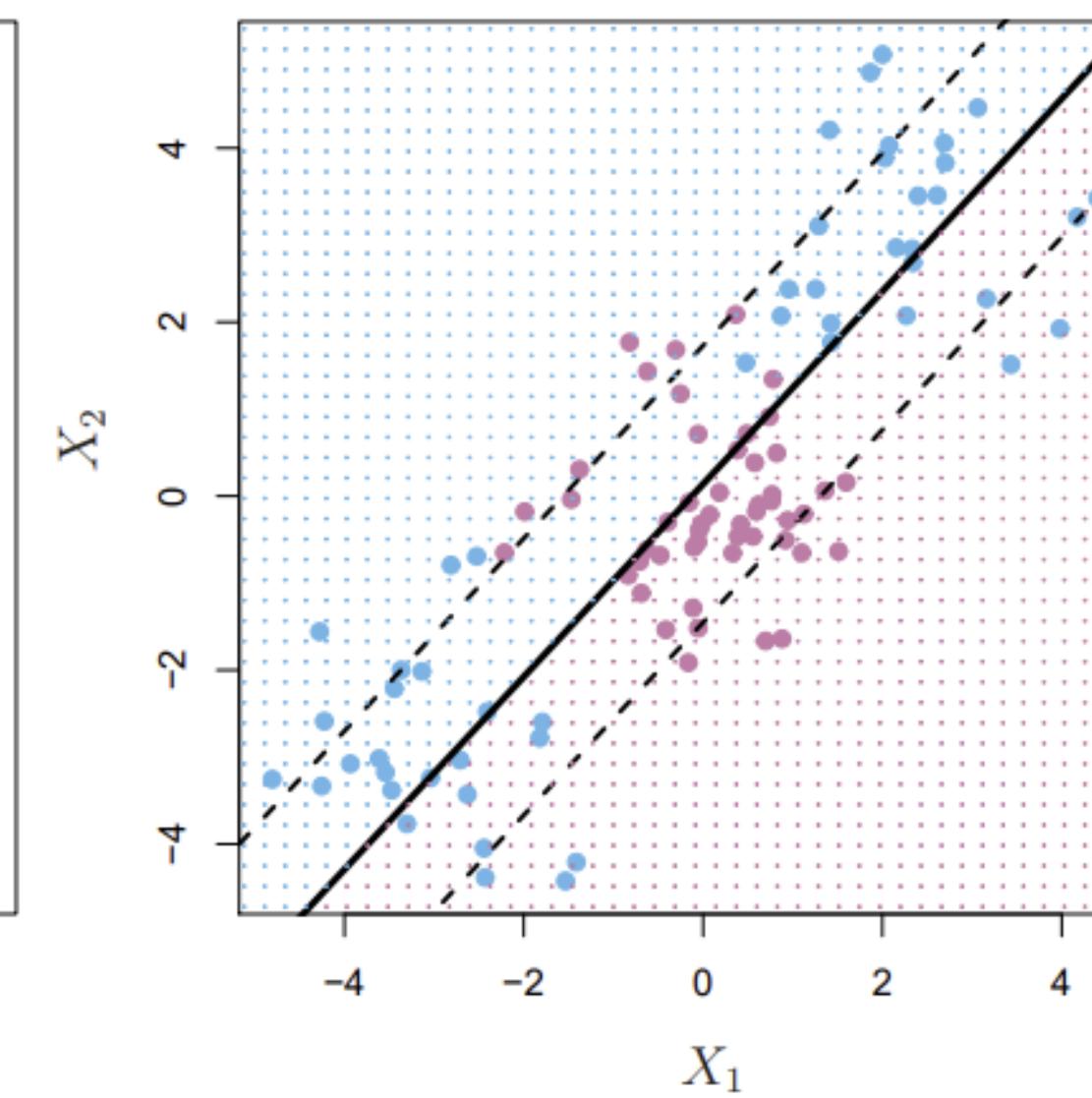
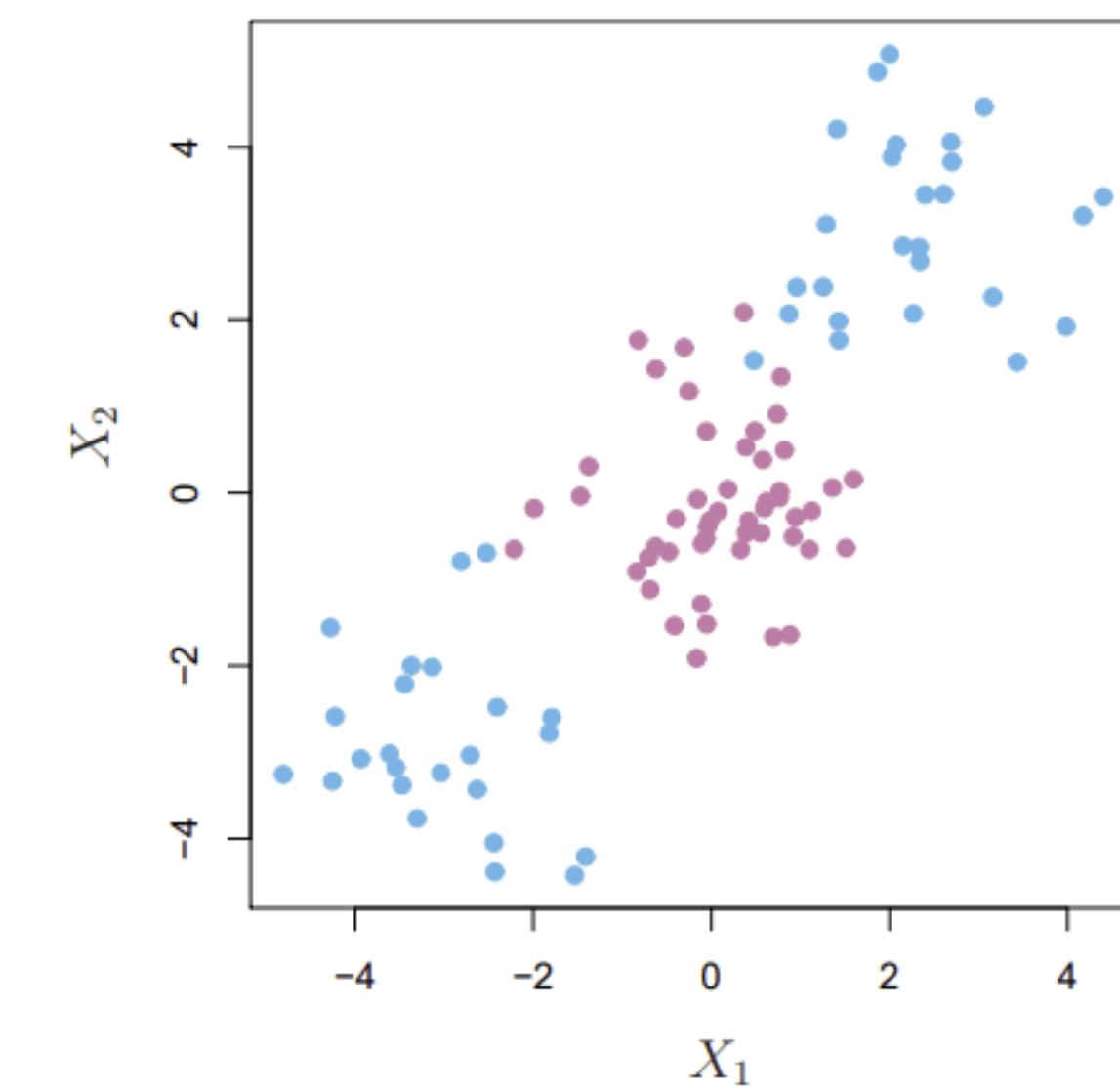
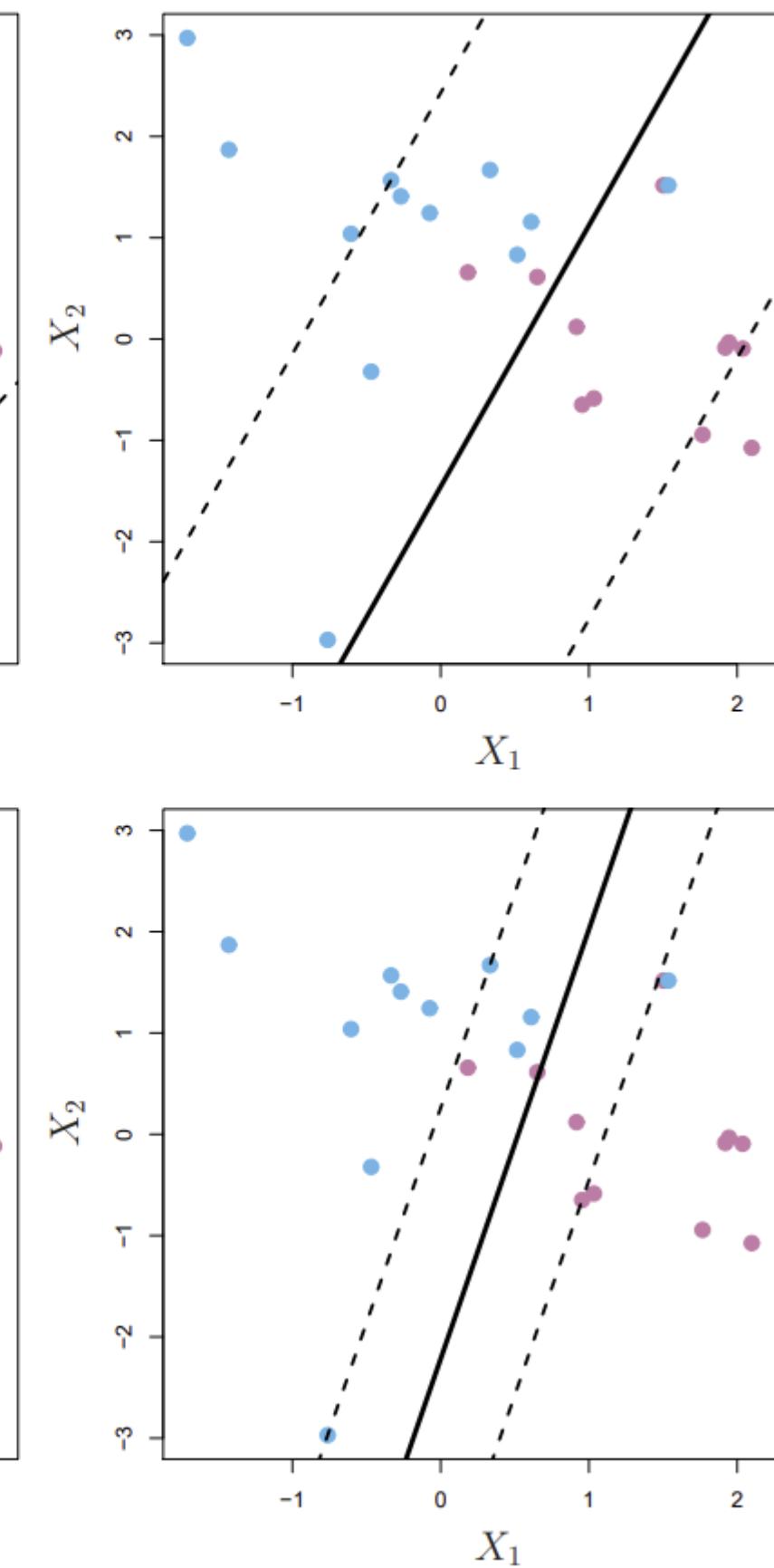
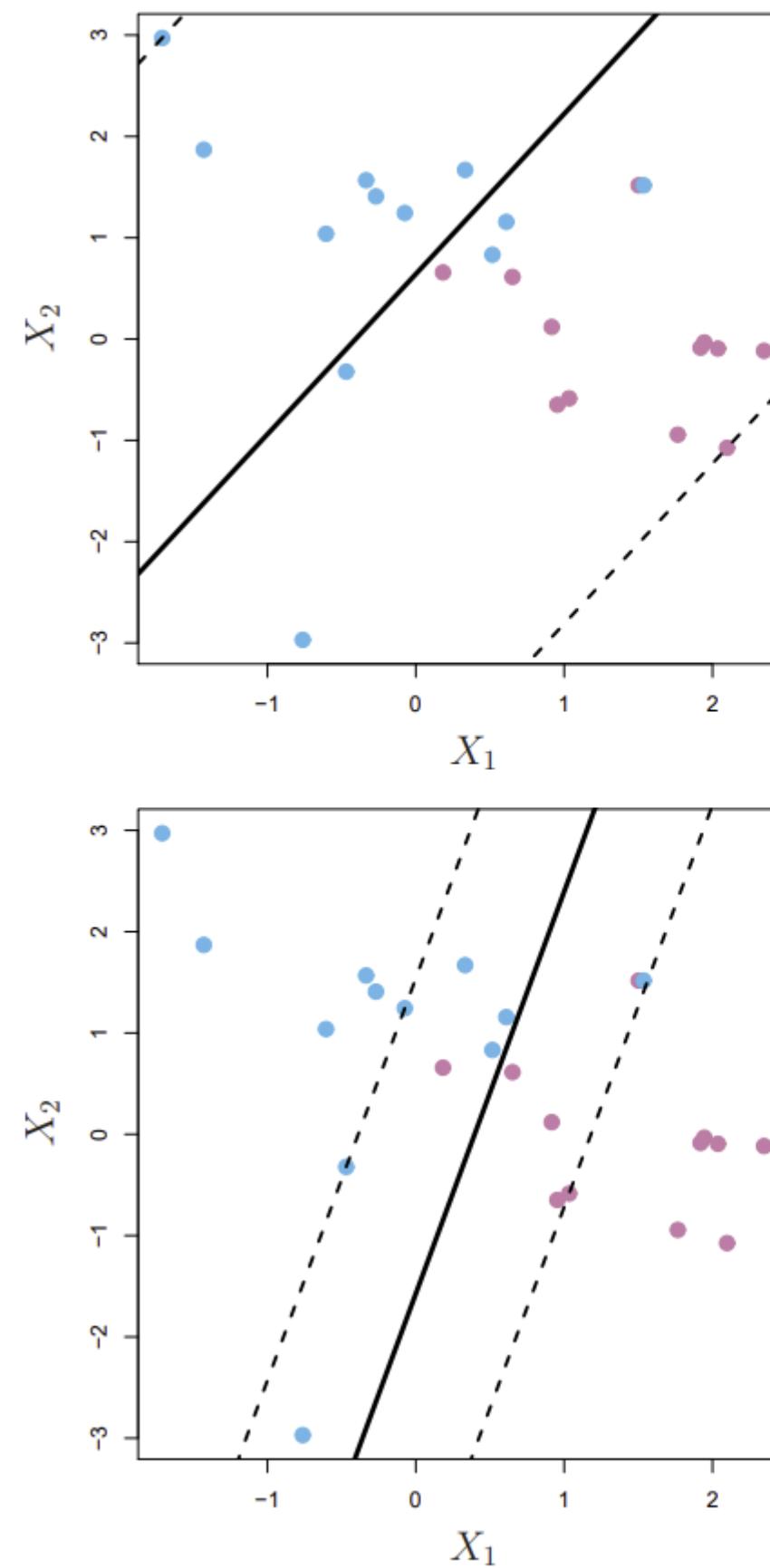
Tuning Parameter C
: Margin 위반을 허용하는 수와 정도
 $C=0 \rightarrow$ Margin 위반 절대 X, 그냥 maximal margin classifier
 $C>1 \rightarrow$ 최대 C개의 데이터의 Margin 위반 허용
 C 가 작음 \rightarrow margin을 좁게 유지
 $\rightarrow \text{Bias} \downarrow, \text{Variance} \uparrow \rightarrow$ overfitting 가능성 존재.
 C 가 큼 \rightarrow margin을 넓게 유지
 $\rightarrow \text{Bias} \uparrow, \text{Variance} \downarrow \rightarrow$ 일반화 성능이 좋아짐.

Support Vector Classifier도 Maximal Margin Classifier와 마찬가지로 **support vector**에만 의존한다.

\rightarrow hyperplane에서 멀리 떨어진 data들에 대해 상당히 robust하다!



Support Vector Classifier(Soft Margin Classifier)



그럼 이런 비선형인 경우는 어떡하지...?

Tuning parameter C를 조정하면서 SVC를 modeling



Classification with Non-Linear Decision Boundaries

실제 데이터에서는 클래스 간 경계가 **non-linear**인 경우가 많아 SVC가 제대로 작동하지 않을 수 있다.
→ 변수 공간(feature space)을 확장하여 비선형 경계를 모델링하자!

Linear Regression에서도 X와 Y가 비선형관계이면 다항식을 추가하여 선형 관계로 변환시킨다.

Ex) polynomial regression, interaction term…

이와 같은 원리로 SVC도 feature space를 확장시킬 수 있다!

$$X_1, X_2, \dots, X_p$$



$$X_1, X_1^2, X_2, X_2^2, \dots, X_p, X_p^2$$

$$\begin{aligned} & \underset{\beta_0, \beta_{11}, \beta_{12}, \dots, \beta_{p1}, \beta_{p2}, \epsilon_1, \dots, \epsilon_n, M}{\text{maximize}} \quad M \\ & \text{subject to } y_i \left(\beta_0 + \sum_{j=1}^p \beta_{j1} x_{ij} + \sum_{j=1}^p \beta_{j2} x_{ij}^2 \right) \geq M(1 - \epsilon_i), \\ & \sum_{i=1}^n \epsilon_i \leq C, \quad \epsilon_i \geq 0, \quad \sum_{j=1}^p \sum_{k=1}^2 \beta_{jk}^2 = 1. \end{aligned}$$

p차원 공간에서 선형 경계를 찾을 수 없음.

- 변수 공간을 확장하여 비선형적인 변수를 추가.
- 새로운 2p차원 공간에서 linear한 boundary를 제공하는 hyperplane 찾기 가능.
- 고차원에서는 데이터가 선형적으로 분리 가능.
- 그러나 실제로 우리가 다뤄야 하는 p차원 입장에서는 비선형 boundary임.



Classification with Non-Linear Decision Boundaries

실제 데이터에서는 클래스 간 경계가 **non-linear**인 경우가 많아 SVC가 제대로 작동하지 않을 수 있다.
→ 변수 공간(feature space)을 확장하여 비선형 경계를 모델링하자!

Linear Regression에서도 X와 Y가 비선형관계이면 다항식을 추가하여 선형 관계로 변환시킨다.

Ex) polynomial regression, interaction term…

이와 같은 원리로 SVC도 feature space를 확장시킬 수 있다!

$$X_1, X_2, \dots, X_p$$



$$X_1, X_1^2, X_2, X_2^2, \dots, X_p, X_p^2$$

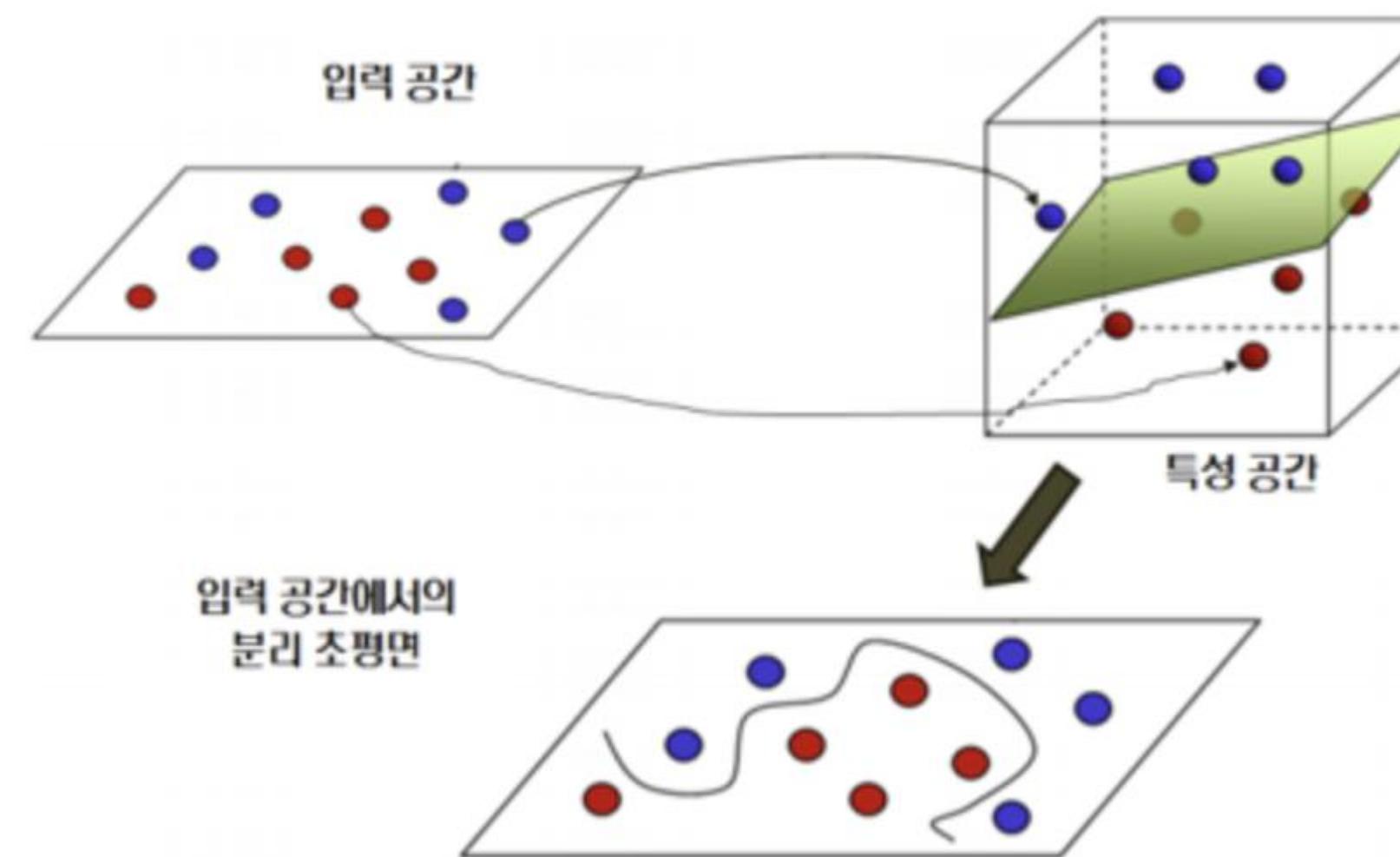
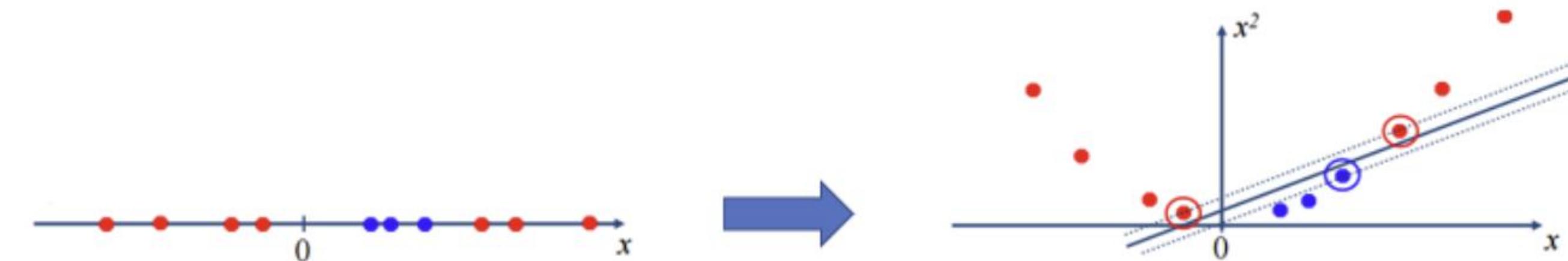
$$\begin{aligned} & \underset{\beta_0, \beta_{11}, \beta_{12}, \dots, \beta_{p1}, \beta_{p2}, \epsilon_1, \dots, \epsilon_n, M}{\text{maximize}} && M \\ & \text{subject to } y_i \left(\beta_0 + \sum_{j=1}^p \beta_{j1} x_{ij} + \sum_{j=1}^p \beta_{j2} x_{ij}^2 \right) \geq M(1 - \epsilon_i), \\ & \sum_{i=1}^n \epsilon_i \leq C, \quad \epsilon_i \geq 0, \quad \sum_{j=1}^p \sum_{k=1}^2 \beta_{jk}^2 = 1. \end{aligned}$$

p차원 공간에서 선형 경계를 찾을 수 없음.

- 변수 공간을 확장하여 비선형적인 변수를 추가.
- 새로운 2p차원 공간에서 linear한 boundary를 제공하는 hyperplane 찾기 가능.
- 고차원에서는 데이터가 선형적으로 분리 가능.
- 그러나 실제로 우리가 다뤄야 하는 p차원 입장에서는 비선형 boundary임.



Classification with Non-Linear Decision Boundaries



<https://m.blog.naver.com/winddori2002/221662413641>



ESC

Support Vector Machine with Kernel

Support Vector Machine: SVC의 확장개념, 커널(kernel)을 사용하여 특정 방식으로 변수 공간을 확장한 결과.
X와 Y 사이의 비선형 관계를 설명하기 위해 변수 공간 확장은 필수적인데, SVM은 커널을 사용하여 효율적인 연산량으로 변수 공간을 확장한다.

비선형 관계를 다루기 위해서는 고차원 공간으로의 변환(mapping)이 필요.
또 decision boundary를 찾으려면 고차원에서의 내적(inner product) 연산 또한 필요.
→ 직접 변환을 시켜서 내적을 계산하면 연산량이 너무 많아지고 효율성이 떨어짐.

이에 대한 해결책이 Kernel Trick!

고차원 변환을 직접 수행하지 않고도 내적 계산을 커널 함수(Kernel function)로 대체하여 효율적으로 처리 가능.

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \longrightarrow K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

$$K(x_i, x_j) = x_i^T x_j$$

Linear Kernel

$$K(x_i, x_j) = (1 + x_i^T x_j)^d$$

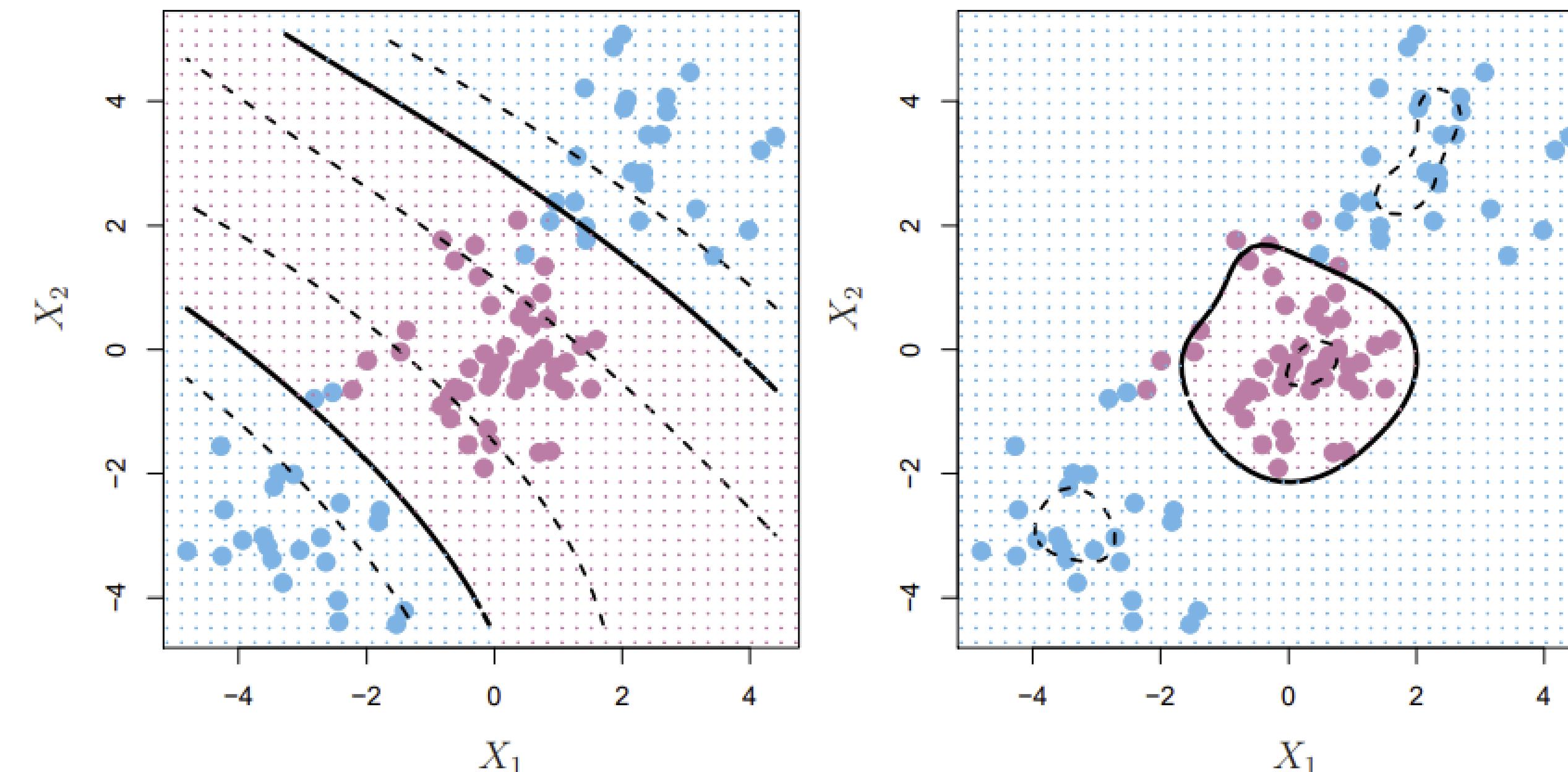
Polynomial Kernel

$$K(x_i, x_j) = \exp \left(-\gamma \sum_{j=1}^p (x_{ij} - x_{i'j})^2 \right)$$

Radial Kernel



Support Vector Machine with Kernel



04

Summary

Summary

Classification: 데이터를 여러 클래스 중 하나로 분류하는 작업, decision boundary를 찾는 방식에 따라 다양한 방법 존재.

Logistic Regression: 확률을 선형적으로 모델링, odds 사용.

Generative Model: $P(X | Y)$ 와 $P(Y)$ 를 추정한 후 베이즈 정리를 이용하여 $P(Y|X)$ 를 계산.

LDA(Linear Discriminant Analysis): 공통 공분산 행렬을 가정, decision boundary를 선형으로 유지.

QDA(Quadratic Discriminant Analysis): 클래스별 공분산 행렬을 따로 추정하여 더 유연한 분류가 가능, decision boundary가 비선형.

Naive Bayes: 조건부 독립 가정, 계산량을 크게 줄이는 방식으로 간단하지만 빠르고 성능이 준수

Maximal Margin Classifier: 가장 가까운 데이터(서포트 벡터)들과의 거리(margin)를 최대화하는 결정 초평면을 찾는 방식, 데이터를 선형으로 분리.

Support Vector Classifier(SVC): 일부 오분류를 허용하는 소프트 마진(soft margin)을 도입한 방식.

Support Vector Machine with Kernel(SVM): 커널 트릭(Kernel Trick)을 이용해 고차원 공간으로 변환 없이 효율적으로 문제 해결.



THANK YOU
