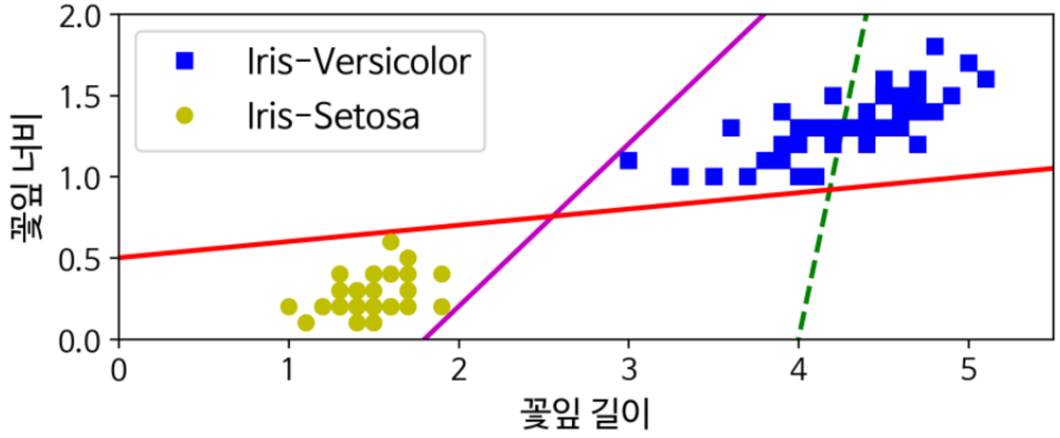
[2021.02.24] ML STUDY 5weeks
발표자: 서민경, 성지현

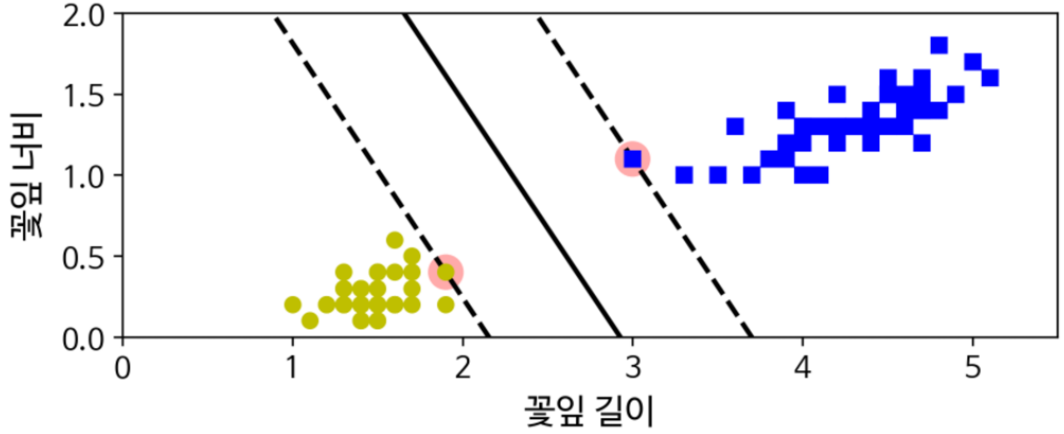
Support Vector Machine (SVM)

Contents

1 마진과 서포트 벡터

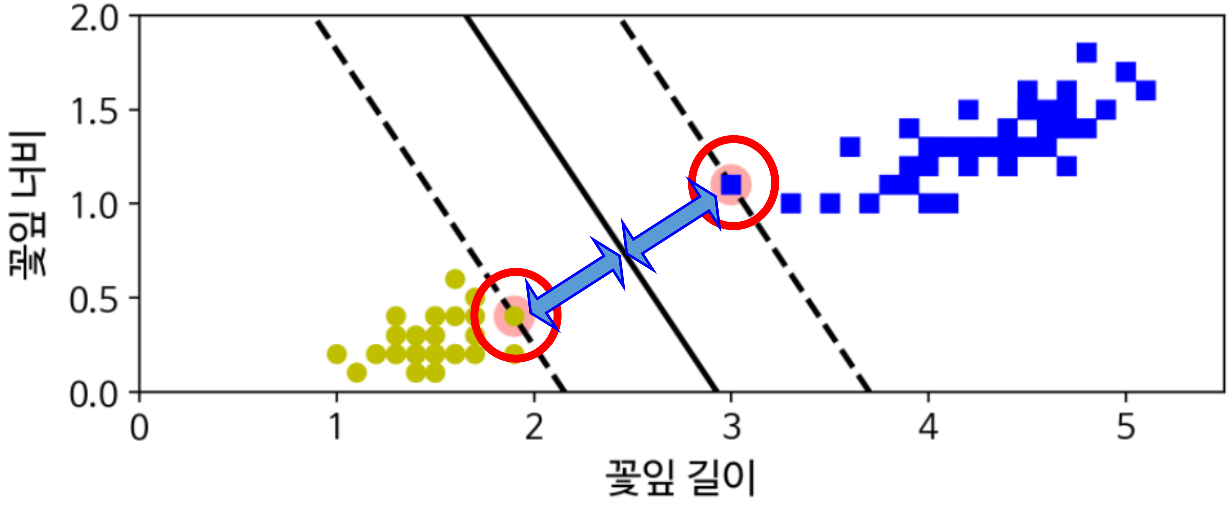


- **초록 점선** : 두 샘플을 분리하지 못함
- **빨간 선**, **보라 선** : 두 샘플을 분리함
  하지만, 샘플과 너무 가까워, 새로운 test data가 들어왔을 때 제대로 분리하지 못할 가능성 존재함.



- **검정 실선** : 두 샘플을 분리함
  실선으로부터 가장 가까운 샘플과도 일정한 거리를 유지해, 새로운 test data가 들어왔을 때에도 제대로 분리할 것으로 기대됨

1 마진과 서포트 벡터



[검정 실선]
분할 초평면
: 두 샘플을 분리하는 실선 (평면)

[빨간 동그라미]
서포트 벡터 (Support Vector)
: 초평면에 가장 가까운 몇 개의 훈련 샘플 포인트

[파란색 화살표]
마진 (Margin)
: 서로 다른 클래스의 서포트 벡터에서 초평면에 달하는 거리의 합

→ 두 샘플을 잘 분리하고 싶다면, 마진의 크기가 가장 큰 분할 초평면을 선택해야 함!

1 마진과 서포트 벡터

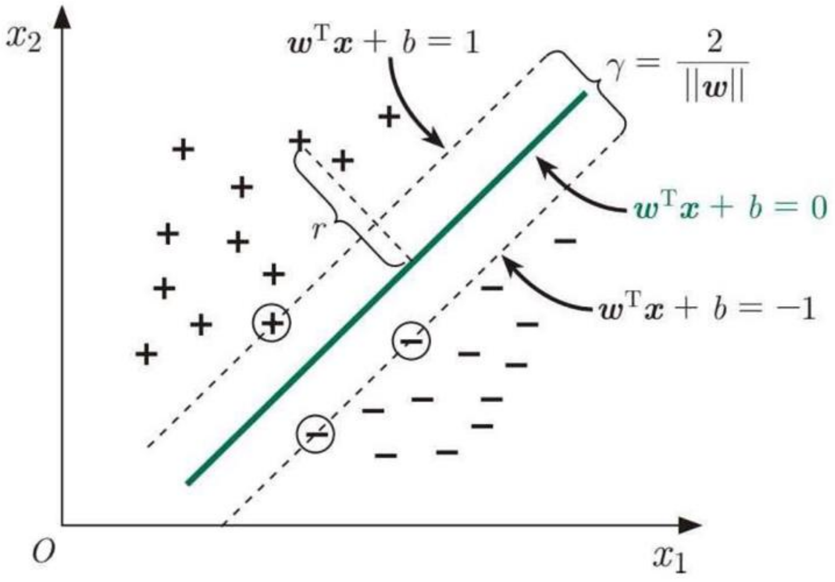


그림 6.2 서포트 벡터와 마진

- **분할 초평면** : $\omega^T x + b = 0$

ω : 법선 벡터; 초평면 방향 결정
b : 초평면과 원점 간의 거리를 결정

- **임의의 점 x에서 초평면까지의 거리** : $r = \frac{|\omega^T x + b|}{||\omega||}$

- **마진** : $\gamma = \frac{2}{||\omega||}$

$\boldsymbol{f(x) = \omega^T x + b}$: **서포트 벡터 머신의 기본 모델!**

최대 마진을 가지는 분할 초평면을 가지고 싶다면? $||\omega||^{-1}$ 을 최대화 시키면 됨!

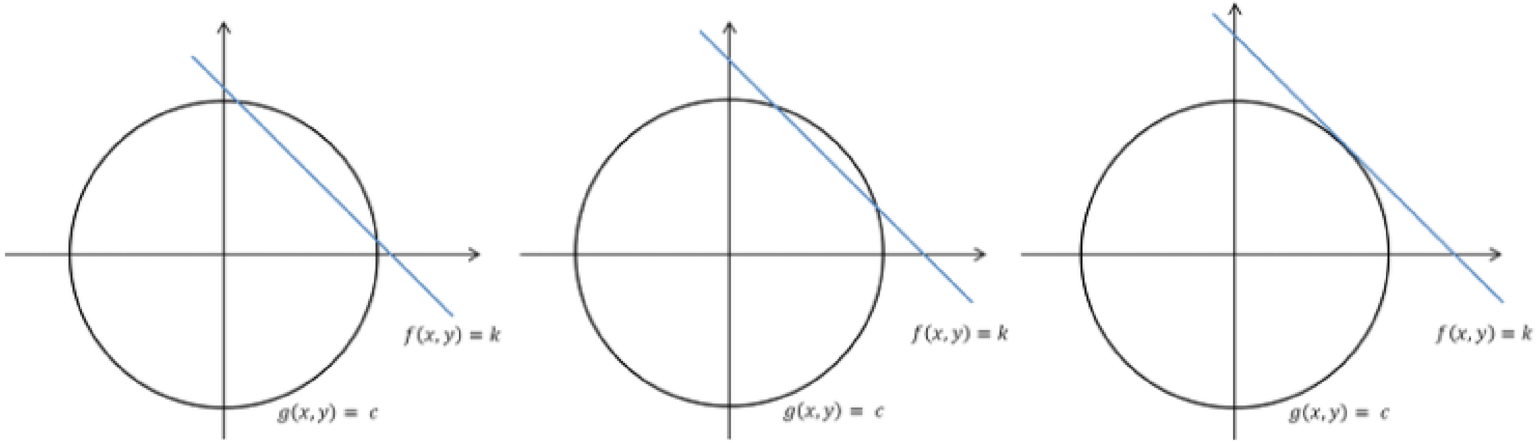
$$\max_{\boldsymbol{w},b} \frac{2}{||\boldsymbol{w}||} \quad \text{s.t. } y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geqslant 1, \quad i = 1, 2, \ldots, m.$$

2 쌍대 문제

라그랑주 승수법 (Lagrange multiplier method)

: 제약 조건이 있는 최적화 문제를 풀기 위해 고안된 방법. (최적 해의 필요 조건을 찾는 방법)

***기본 가정**: 제약 조건 g를 만족하는 f의 최소/최댓값은 f와 g가 접하는 지점에 존재할 수도 있다.*



[그림 1] 제약 조건 $g(x,y)=c$를 만족하는 $f(x,y)$의 최댓값 문제에 대한 기하학적 표현

- f와 g가 접하는 지점을 찾기 위해 gradient vector를 사용
- 어떠한 지점에서의 접선 벡터와 gradient vector 내적은 0 → gradient vector와 접선 벡터는 수직

$$\nabla f = \lambda \nabla g \quad ①$$

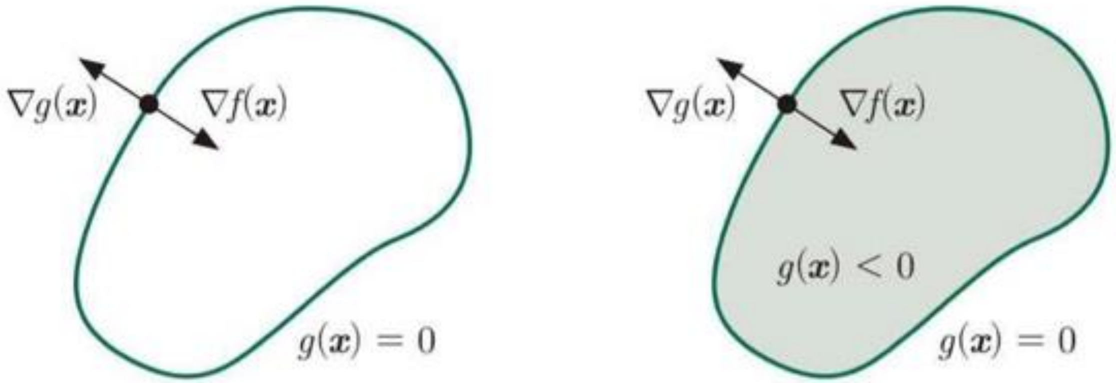
* λ : 라그랑주 승수

- 라그랑주 승수법에서는 다음의 보조 함수를 정의함. $L(x,y,\lambda) = f(x,y) - \lambda(g(x,y)-c)$ ②

- $\nabla L = 0$의 식은 ① 식과 동일함

➔ L을 x, y로 편미분 한 식과 제약 조건인 g(x,y)=c를 이용하면 미지수 3개인 문제의 해를 구할 수 있음!

라그랑주 승수법 (Lagrange multiplier method)



(a) 등식 제약 조건 (b) 부등식 제약 조건

그림 B.1 라그랑주 승수법의 기하학적 함의: (a) 등식 제약 조건 $g(x) = 0$ 혹은 (b) 부등식 제약 조건 $g(x) \leqslant 0$에서 목표 함수 $f(x)$를 최소화한다 (초록색 곡선으로 $g(x) = 0$인 곡면을 표현했고, 음영으로 $g(x) < 0$ 부분을 나타냈다)

- 만약 **제약 조건**이 **부등식 (g(x)≤0)**이라면, ∇f와 ∇g의 방향은 무조건 반대가 되어야 함.
- 따라서, g(x)≤0의 제약이 존재할 때 f(x)를 최소화하는 것은 아래의 제약에서 라그랑주 함수를 최소화 하는 것으로 전환할 수 있음.

$$\begin{cases} g(x) \leq 0 \\ \lambda \geq 0 \\ \lambda g(x) = 0 \end{cases}$$

Karush-Kuhn-Tucker (KKT) 조건

2 쌍대 문제

❖ 라그랑주 승수법 (Lagrange multiplier method)

: 제약식에 형식적인 라그랑주 승수를 곱한 항을 최적화하려는 목적 식에 더함으로 인해,
제약된 문제를 제약이 없는 문제로 바꾸는 기법

$$\boldsymbol{f(x) = \omega^T x + b} \quad + \quad \boldsymbol{\alpha_i \geq 0} \quad \rightarrow \quad \boldsymbol{L(\omega, b, \alpha) = \frac{1}{2}||\omega||^2 + \sum_{i=1}^{m} \alpha_i (1 - y_i(\omega^T x_i + b))}$$

라그랑주 승수

라그랑주 함수 형태로 바꾼 식

- $L(\omega, b, \alpha)$ 는 $\frac{\partial L}{\partial \omega} = 0,\ \frac{\partial L}{\partial b} = 0$인 지점에서 최소값 가짐

$$\frac{\partial L}{\partial \omega} = \omega - \sum_{i=1}^{m} \alpha_i y_i x_i = 0 \quad \rightarrow \quad \omega = \sum_{i=1}^{m} \alpha_i y_i x_i$$

→ 이 식을 위의 $L(\omega, b, \alpha)$ 식에 대입해서 정리!

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{m} \alpha_i y_i = 0 \quad \rightarrow \quad \sum_{i=1}^{m} \alpha_i y_i = 0$$

2 쌍대 문제

$$L(\omega, b, \alpha) = \frac{1}{2}||\omega||^2 + \sum_{i=1}^{m} \alpha_i (1 - y_i(\omega^T x_i + b))$$

← $\boldsymbol{\omega = \sum_{i=1}^{m} \alpha_i y_i x_i}$ 식 대입!

$$\begin{aligned} \frac{1}{2}\|w\|_2^2 &= \frac{1}{2} w^T w \\ &= \frac{1}{2} w^T \sum_{j=1}^{n} \alpha_j y_j x_j \\ &= \frac{1}{2} \sum_{j=1}^{n} \alpha_j y_j (w^T x_j) \\ &= \frac{1}{2} \sum_{j=1}^{n} \alpha_j y_j (\sum_{i=1}^{n} \alpha_i y_i x_i^T x_j) \\ &= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \end{aligned}$$

$$\begin{aligned} -\sum_{i=1}^{n} \alpha_i (y_i (w^T x_i + b) - 1) &= -\sum_{i=1}^{n} \alpha_i y_i (w^T x_i + b) + \sum_{i=1}^{n} \alpha_i \\ &= -\sum_{i=1}^{n} \alpha_i y_i w^T x_i - b \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i \\ &= -\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{n} \alpha_i \end{aligned}$$

➔ $\max L_D(\alpha_i) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{i=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j$ 조건 : $\sum_{i=1}^{m} \alpha_i y_i = 0$, $\alpha_i \geq 0$

2 쌍대 문제

- α의 해를 구한 후, ω와 b를 구하면, SVM 모델 식은 다음의 형태로 얻을 수 있음.

$$\boldsymbol{f(x) = \omega^T x + b = \sum_{i=1}^{m} \alpha_i y_i x_i^T x + b}$$

- KKT 조건에 의해, 지금까지 전개한 식들은 아래의 조건을 만족해야 함.

$$\boldsymbol{\alpha_i \geq 0} \qquad \boldsymbol{y_i f(x_i) - 1 \geq 0} \qquad \boldsymbol{\alpha_i (y_i f(x_i) - 1) = 0}$$

- 즉, 어떠한 훈련 샘플이라도 $\boldsymbol{\alpha_i = 0}$ 또는 $\boldsymbol{y_i f(x_i) = 1}$을 만족할 것임

1) $\alpha_i = 0$인 경우, 위 f(x) 식의 시그마 부분이 사라짐 → f(x)에 아무런 영향 가하지 못함

2) $y_i f(x_i) = 1$인 경우, 이에 대응하는 샘플 포인트는 최대 마진 경계상에 위치할 것임
→ 해당 샘플 포인트는 서포트 벡터에 해당함!
: ***서포트 벡터의 중요한 성질 중 하나!***

2 쌍대 문제

➢ 그렇다면, 아래의 식의 해는 어떻게 구할 수 있을까?

$$\max L_D(\alpha_i) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{i=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j \quad \text{조건 :} \quad \sum_{i=1}^{m} \alpha_i y_i = 0\ , \quad \alpha_i \geq 0$$

➡ ***SMO** (Sequential Minimal Optimization) 방법을 이용!*

SMO :

[기존] α_i 이외의 모든 파라미터를 고정한 뒤, α_i에서의 극한값을 구하는 방법

[현재] $\sum_{i=1}^{m} \alpha_i y_i = 0$ 조건 때문에, 매번 두 개의 변수 α_i, α_j를 제외한 나머지 파라미터를 고정함.

1) α_i, α_j 한 쌍 선택

2) α_i, α_j 이외의 파라미터를 고정시키고 $\max L_D(\alpha_i)$ 식의 해를 구한 후, 새로운 α_i, α_j 얻음

➔ *수렴할 때 까지 1)과 2)를 계속 반복함!*

2 쌍대 문제

기존 $\boldsymbol{max\, L_D(\alpha_i)}$ 식의 제약 조건

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

$\boldsymbol{\alpha_i, \alpha_j}$만 고려할 때, $\boldsymbol{max\, L_D(\alpha_i)}$ 식의 제약 조건

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \quad \rightarrow \alpha_i y_i + \alpha_j y_j = c = -\sum_{k \neq i,j} \alpha_k y_k$$

$$\alpha_i \geq 0, \quad \alpha_j \geq 0$$

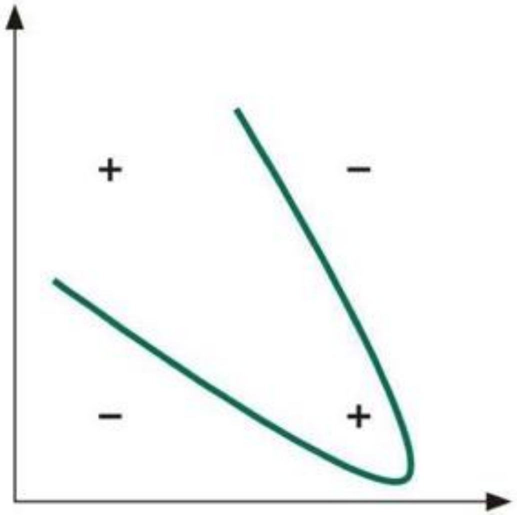
$\boldsymbol{\alpha_i y_i + \alpha_j y_j = c = -\sum_{k\neq i,j} \alpha_k y_k}$ 식을 이용해 $\boldsymbol{max\, L_D(\alpha_i)}$ 식의 $\boldsymbol{\alpha_j}$를 소거하면,
$\boldsymbol{\alpha_i}$의 단변량 이차 프로그래밍 문제로 변함!

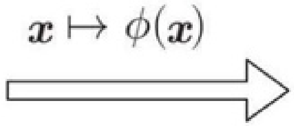
- $\boldsymbol{f(x) = \omega^T x + b}$ 식의 b 항 (바이어스 항)

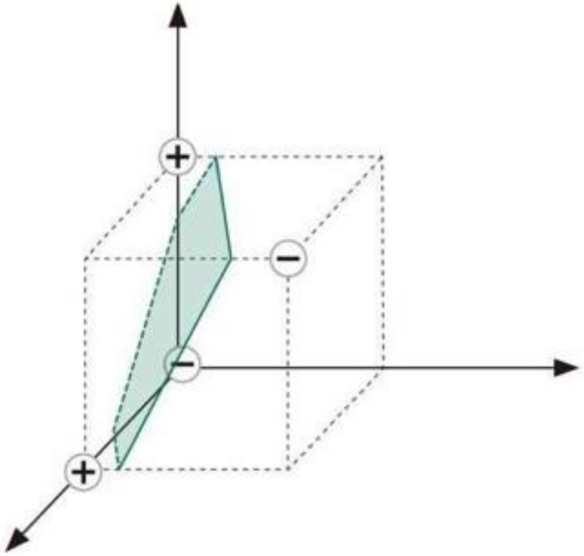
  모든 서포트 벡터 (x_s, y_s)에 대해 $y_s f(x_s) = 1$이 존재하기 때문에, 아래의 식으로 구할 수 있음.

$$y_s\left(\sum_{i \in S} \alpha_i y_i x_i^T x_s + b\right) = 1 \quad \Rightarrow \quad b = \frac{1}{|S|} \sum_{s \in S} \left(\frac{1}{y_s} - \sum_{i \in S} \alpha_i y_i x_i^T x_s\right)$$

3 커널 함수





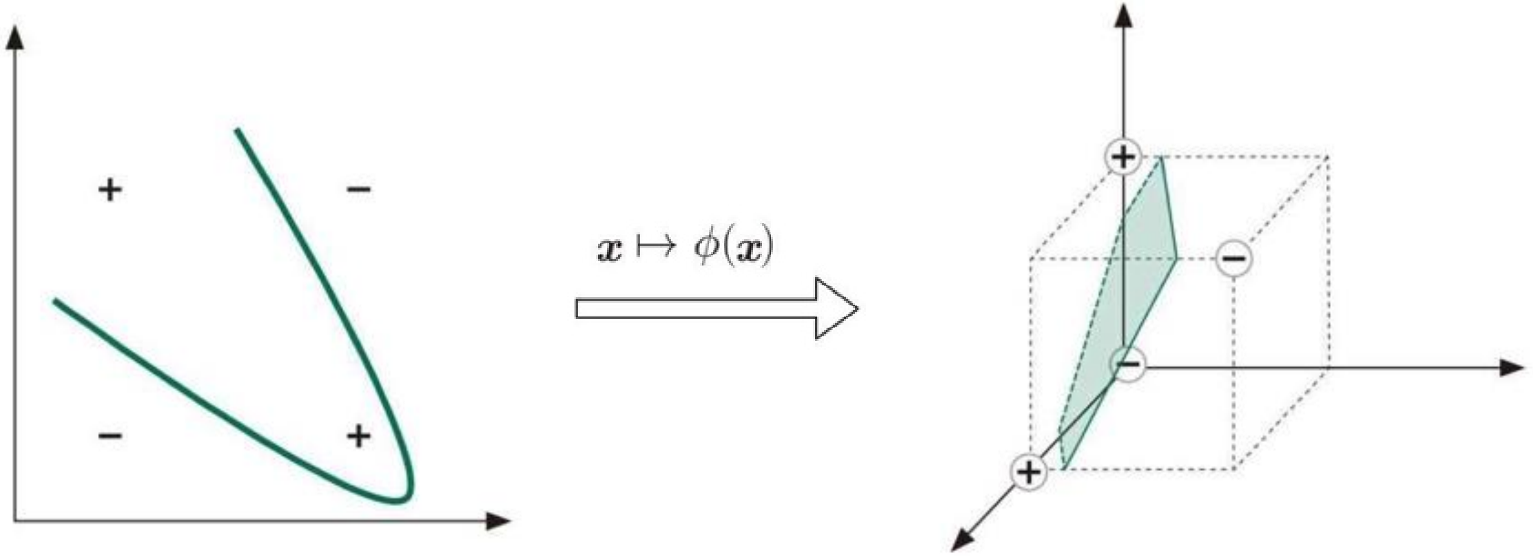


위의 훈련 샘플은 선형
분리가 불가능함

기존 원시 공간보다 **더 높은 차원의 특성 공간**으로
투영하여 선형 분리가 가능하게 만들 수 있음

"만약 원시 공간이 유한한 차원을 가졌다면, 즉 속성의 개수가 유한하다면
반드시 고차원특성 공간에서 샘플을 분할할 수 있다!"

3 커널 함수



- $\boldsymbol{\phi(x)}$: x를 투영시킨 후의 고유 벡터
- $\boldsymbol{f(x) = \omega^T \phi(x) + b}$: 특성 공간에서 분할 초평면에 대응하는 모델

➢ 위 경우의 쌍대 문제 식 : $\max \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j)$ 조건 : $\sum_{i=1}^{m} \alpha_i y_i = 0$, $\alpha_i \geq 0$

✓ $\phi(x_i)^T \phi(x_j)$: 차원이 매우 높아 계산하기 힘들 수 있음

3 커널 함수

커널 함수 $\boldsymbol{\kappa}(x_i, x_j)$ 가정! $\boldsymbol{\kappa}(x_i, x_j) = <\boldsymbol{\phi}(x_i), \boldsymbol{\phi}(x_j)> = \boldsymbol{\phi}(x_i)^T \boldsymbol{\phi}(x_j)$

[가정 전] $\max \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j)$ 조건 : $\sum_{i=1}^{m} \alpha_i y_i = 0$, $\alpha_i \geq 0$

[가정 후] $\max \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \kappa(x_i, x_j)$ 조건 : $\sum_{i=1}^{m} \alpha_i y_i = 0$, $\alpha_i \geq 0$

$$\boldsymbol{f(x) = \omega^T \phi(x) + b = \sum_{i=1}^{m} \alpha_i y_i \kappa(x_i, x_j) + b}$$

: 서포트 벡터 전개 (Support vector expansion)

" 모든 문제에 대해, 적절한 커널 함수는 반드시 존재할까? "

3 커널 함수

정리 6.1 **커널 함수:** $\mathcal{X}$가 입력 공간을 나타내고, $\kappa(\cdot,\cdot)$가 $\mathcal{X} \times \mathcal{X}$에서의 대칭 함수라면 κ는 모든 데이터 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$에 대한 커널 함수다. 커널 행렬 kernel matrix $\mathbf{K}$는 항상 양의 준정부호 행렬positive semi-definite이다.

$$\mathbf{K} = \begin{bmatrix} \kappa(\boldsymbol{x}_1,\boldsymbol{x}_1) & \cdots & \kappa(\boldsymbol{x}_1,\boldsymbol{x}_j) & \cdots & \kappa(\boldsymbol{x}_1,\boldsymbol{x}_m) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \kappa(\boldsymbol{x}_i,\boldsymbol{x}_1) & \cdots & \kappa(\boldsymbol{x}_i,\boldsymbol{x}_j) & \cdots & \kappa(\boldsymbol{x}_i,\boldsymbol{x}_m) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \kappa(\boldsymbol{x}_m,\boldsymbol{x}_1) & \cdots & \kappa(\boldsymbol{x}_m,\boldsymbol{x}_j) & \cdots & \kappa(\boldsymbol{x}_m,\boldsymbol{x}_m) \end{bmatrix}.$$

- 하나의 대칭함수에 대응하는 커널 행렬이 양의 준정부호 행렬이라면, 이를 커널 함수로 사용할 수 있음.
- 모든 커널 함수는 은연중에 재생 커널 힐베르트 공간 (Reproducing Kernel Hilbert Space, RKHS) 이라고 부르는 특성 공간을 정의하고 있음.

  ➔ **항상 대응하는 투영 ϕ를 찾을 수 있음!**

- 우리는 샘플이 공간 내에서 선형 분리되길 원함
- 따라서, 특성 공간의 좋고 나쁨은 서포트 벡터 머신 성능에 큰 영향을 미침
- 하지만, 특정 투영의 형식을 모를 때 어떤 커널 함수가 가장 적합한지 알 수 없음
- 따라서, **적절한 커널 함수를 선택**하는 것이 **매우 중요함**.

3 커널 함수

표 6.1 자주 사용하는 커널 함수

명칭	표현식	파라미터
선형 커널	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$	
다항식 커널	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j)^d$	다항식의 차수는 $d \geqslant 1$
가우스 커널	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\lVert \boldsymbol{x}_i - \boldsymbol{x}_j \rVert^2}{2\sigma^2}\right)$	가우스 커널의 넓이는 $\sigma > 0$
라플라스 커널	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\lVert \boldsymbol{x}_i - \boldsymbol{x}_j \rVert}{\sigma}\right)$	$\sigma > 0$
시그모이드 커널	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\beta \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j + \theta)$	$\beta > 0,\ \theta > 0$, tanh은 탄젠트 하이퍼볼릭 함수(hyperbolic tangent function)

- 만약 κ_1과 κ_2가 커널 함수라면, 임의의 정수 γ_1, γ_2에 대한 선형 조합

$$\gamma_1 \kappa_1 + \gamma_2 \kappa_2 \quad \text{식 6.25}$$

- 만약 κ_1과 κ_2가 커널 함수라면, 커널 함수의 직접곱direct product

$$\kappa_1 \otimes \kappa_2(\boldsymbol{x}, \boldsymbol{z}) = \kappa_1(\boldsymbol{x}, \boldsymbol{z}) \kappa_2(\boldsymbol{x}, \boldsymbol{z}) \quad \text{식 6.26}$$

- 만약 κ_1이 커널 함수라면 모든 함수$g(\boldsymbol{x})$에 대해

$$\kappa(\boldsymbol{x}, \boldsymbol{z}) = g(\boldsymbol{x}) \kappa_1(\boldsymbol{x}, \boldsymbol{z}) g(\boldsymbol{z}) \quad \text{식 6.27}$$

이때, 식 6.25부터 6.27까지는 모두 커널 함수입니다.

2. default hyperparameter를 사용한 SVM 모델

- SVM (using default hyperparameter)

```
from sklearn.svm import SVC
from sklearn import metrics
svc = SVC()
svc.fit(X_train, y_train)
y_pred = svc.predict(X_test)
print('Accuracy Score :')
print(metrics.accuracy_score(y_test,y_pred))
```

```
Accuracy Score :
0.9763406940063092
```

- SVM (using default linear kernel)

```
svc = SVC(kernel='linear')
svc.fit(X_train, y_train)
y_pred = svc.predict(X_test)
print('Accuracy Score :')
print(metrics.accuracy_score(y_test,y_pred))
```

```
Accuracy Score :
0.9779179810725552
```

- SVM (using default RBF kernel)

```
svc = SVC(kernel='rbf')
svc.fit(X_train, y_train)
y_pred = svc.predict(X_test)
print('Accuracy Score :')
print(metrics.accuracy_score(y_test,y_pred))
```

```
Accuracy Score :
0.9763406940063092
```

- SVM (using default polynomial kernel)

```
svc = SVC(kernel='poly')
svc.fit(X_train, y_train)
y_pred = svc.predict(X_test)
print('Accuracy Score :')
print(metrics.accuracy_score(y_test,y_pred))
# 정확도 하락 (polynomial kernel은 training dataset을 overfitting해서 일 가능성 존재)
```

```
Accuracy Score :
0.9589905362776026
```