Support Vector Machine:

ESC 2024 Spring Session Final Project 1조

김근영 김준엽 오동윤 송승은 장덕재

Contents

1

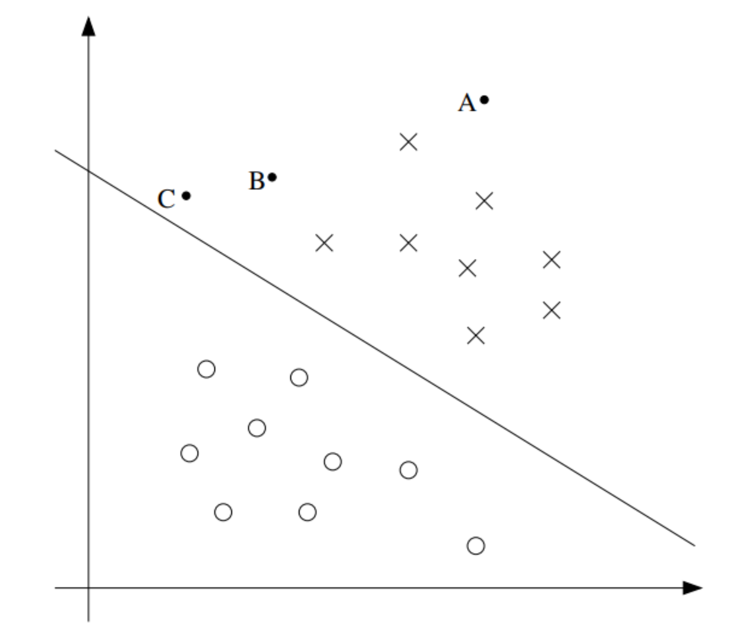
Support Vector Machine

How to find the best hyperplane

What is Hyperplane?

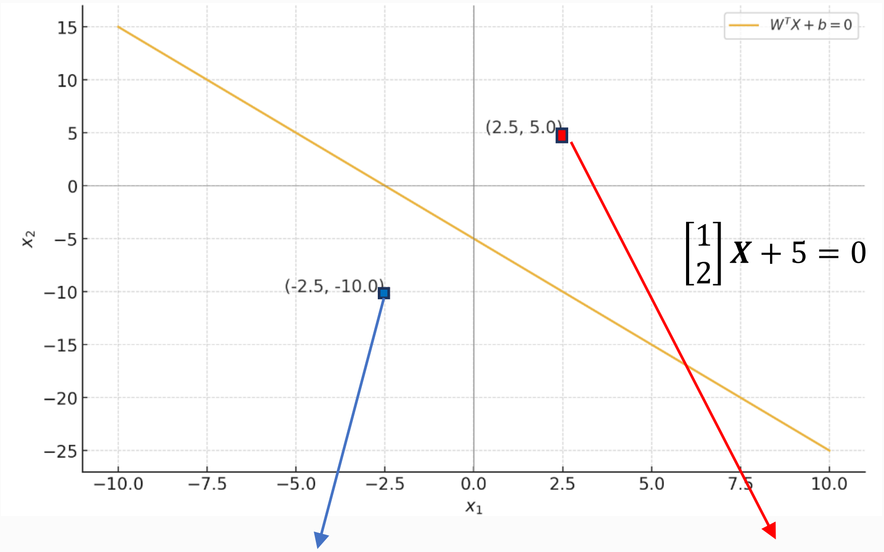
- Support vector machine은 그림의 O와 X를 나누는 최적의 boundary를 찾는 머신러닝 기법이다.
- 이 boundary를 초평면(hyperplane)이라고 한다.
- Hyperplane을 수식적으로 나타내면 다음과 같다:

$$\boldsymbol{W}^T\boldsymbol{X} + \boldsymbol{b} = 0$$



What is Hyperplane?

- Hyperplane을 기준으로 서로 다른 영역에 있는 데이터를 $\boldsymbol{W}^T\boldsymbol{X} + b$에 넣었을 때, 다른 부호가 나오게 된다
- X에 대응되는 데이터를 $\boldsymbol{W}^T\boldsymbol{X} + b$에 넣으면 양수, O에 대응되는 데이터를 넣으면 음수가 나오는 Hyperplane을 찾는다
- 따라서 유효한 Hyperplane은 다음의 특징을 만족해야 한다
  - $\boldsymbol{W}^T x + b > 0\ if\ x\ is\ data\ corresponding\ to\ X$
  - $\boldsymbol{W}^T x + b < 0\ if\ x\ is\ data\ corresponding\ to\ O$
- 그런데 위의 식은 다소 복잡하기 때문에, 문제를 단순화시키기 위해 X에 대응되는 데이터의 label로는 1, O에 대응되는 데이터의 label로는 -1을 부여한 뒤, $y(\boldsymbol{W}^T\boldsymbol{X} + b)$를 계산하면, 이 값은 항상 양수의 값을 가진다
- 따라서 유효한 Hyperplane이 가져야 하는 특징을 다음과 같이 단순화시켜 나타낼 수 있다:
  - $\boldsymbol{y}_i(\boldsymbol{W}^T\boldsymbol{X}_i + \boldsymbol{b}) > 0 \text{ for } \mathrm{i} = 1, \ldots, \mathrm{n}$

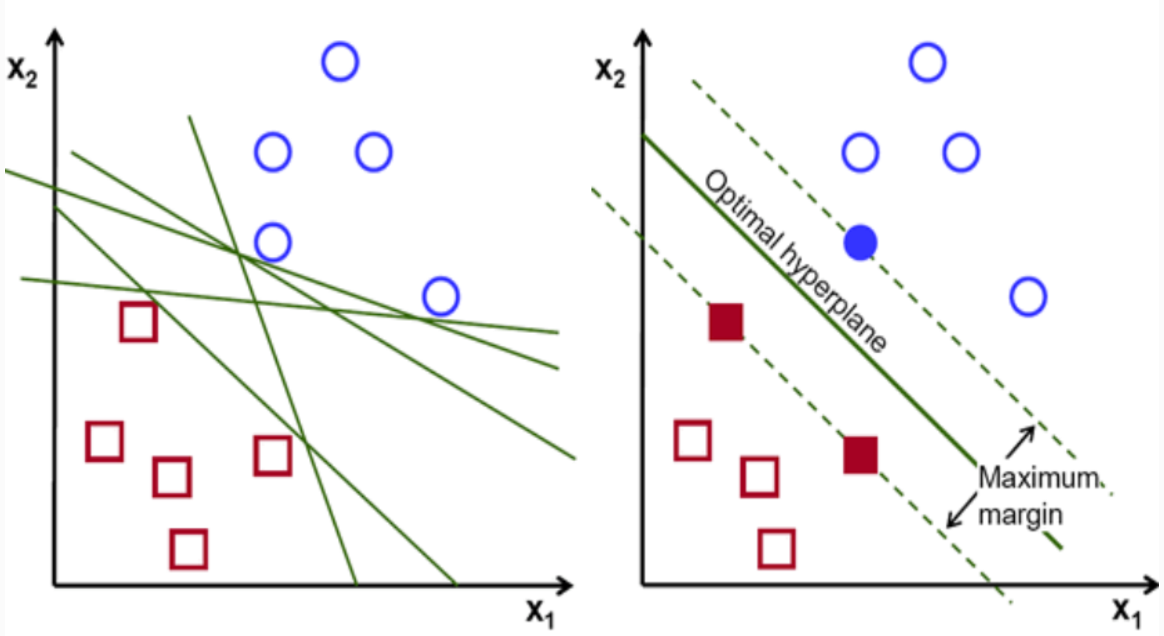


$\begin{bmatrix}1\\2\end{bmatrix}[-2.5 \quad -10] + 5 = -17.5$

$\begin{bmatrix}1\\2\end{bmatrix}[2.5 \quad 5] + 5 = 17.5$

Optimal Hyperplane

- 하지만 문제는 데이터를 완벽하게 분리하는 ‘유효한’ Hyperplane이 그림과 같이 여러 개 존재할 수 있다.
- 유효한 Hyperplane 중에서 이제는 ‘optimal’ hyperplane 을 찾아야 한다.
- Optimal hyperplane: 가장 큰 Margin을 가지는 hyperplane
- Margin: Support Vector들 사이의 거리
- Support Vector: Hyperplane에서 가장 가까운 각 진영의 데이터포인트
- 이때 Optimal hyperplane은 Support Vector들 사이의 중앙에 위치해야 한다



Computing Margin

- 어떠한 Hyperplane이 유효하고, Support Vector의 정중앙을 지난다고 하자
  - Hyperplane: $\boldsymbol{W}^T\boldsymbol{X} + b = 0$
- Support vector를 x_1, x_2라고 하자.
- 그러면 $\boldsymbol{W}^T x_1 + \boldsymbol{b} = \boldsymbol{c}$, $\boldsymbol{W}^T x_2 + \boldsymbol{b} = -\boldsymbol{c}$이다
- 그러면 Margin은 $\frac{2c}{||\boldsymbol{W}||}$로 나타난다.
- 그런데 W와 b에 대한 scaling을 적절하게 하면 c=1로 만들 수 있기 때문에, 주로 Margin= $\frac{2}{||\boldsymbol{W}||}$으로 생각한다
- 그리고 $\boldsymbol{W}^T\boldsymbol{x}_1 + b = 1$이라는 것은 $y_i(\boldsymbol{W}^T\boldsymbol{X}_{\boldsymbol{i}} + b)$의 최소값이 1이라는 것이기 때문에, 기존에 유효한 Hyperplane을 위한 조건이었던 $y_i(\boldsymbol{W}^T\boldsymbol{X}_{\boldsymbol{i}} + b) > 0 \text{ for } \mathrm{i} = 1, \ldots, \mathrm{n}$을 다음과 같이 조금 더 구체적으로 나타낼 수 있게 된다:
  - $y_i(\boldsymbol{W}^T\boldsymbol{X}_{\boldsymbol{i}} + b) \geq 1 \text{ for } \mathrm{i} = 1, \ldots, \mathrm{n}$

Optimal Hyperplane

- 최적의 Hyperplane을 찾는다는 것은, 유효한 Hyperplane의 조건 하에서 Margin을 최대화시킨다는 것이다
- 따라서 최적화 문제는 다음과 같다
  - $\max_{W,b} \frac{2}{||W||}\ s.t.\, y_i(\boldsymbol{W}^T \boldsymbol{X_i} + b) \geq 1 \text{ for i} = 1, \ldots, \text{n}$
  - $\min_{W,b} \frac{||W||^2}{2} s.t.\, y_i(\boldsymbol{W}^T \boldsymbol{X_i} + b) \geq 1 \text{ for i} = 1, \ldots, \text{n}$

2

Lagrange Multipler Theorem

Theoretical background for optimization

Optimization Problem (via Lagrange Multiplier Theorem)

: Def 용어 : Theorem 이름

Def. optimization problem

$\min_{w} f(w)$ s.t. $h_i(w)=0,\ i=1,\cdots,\ell$, for some constant ℓ, some $f: U\subset\mathbb{R}^n\to\mathbb{R}$, $g: U\subset\mathbb{R}^n\to\mathbb{R}$.

Theorem. Lagrange Multiplier Theorem

$f: U\subset\mathbb{R}^n\to\mathbb{R}$, $g: U\subset\mathbb{R}^n\to\mathbb{R}$, $f, g\in C^1$.

$x_0\in U$, $g(x_0)=c_0$, $S=g^{-1}(c_0)$, $\nabla g(x_0)\neq 0$.

$f|_S$ has maximum or minimum at x_0.

* $f|_S: S\to\mathbb{R}$이 maximum/minimum을 가질 때만 사용가능 ⇒ S가 closed, bounded 임을 보이면 Heine-Borel Thm을 통해 S가 compact이고, Extreme Value theorem에 의해 $f|_S$ maximum & minimum 가짐!

$\Rightarrow \exists \lambda\in\mathbb{R}$ s.t. $\nabla f(x_0)=\lambda\nabla g(x_0)$

pf) Since $\nabla g(x_0)\neq 0$, S is regular surface.

manifold 에서 중요 technique

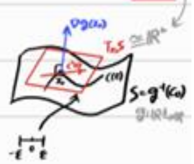
Fix curve $c: (-\epsilon,\epsilon)\to S$, $c(t)$ on S s.t. $c(0)=x_0$, $c'(0)=v$, for some arbitrary $v\in T_{x_0}S$

Then, $\frac{d}{dt}g(c(t))=\frac{d}{dt}c_0=0$. ($\because$ $c(t)$ is on S)

Also, by Chain Rule, $\frac{d}{dt}g(c(t))\Big|_{t=0}=\nabla g(x_0)\cdot c'(0)=\nabla g(x_0)\cdot v$

(∇g(x₀) 이해)

↳ $g:\mathbb{R}^n\to\mathbb{R}$ 이므로 gradient "vector"로 표현되며, 내적으로 표현.



Thus, $\nabla g(x_0)\perp v$

Since $x_0\in U$ is extrema, $\frac{d}{dt}f(c(t))\Big|_{t=0}=0$.

Also, by Chain Rule, $\frac{d}{dt}f(c(t))\Big|_{t=0}=\nabla f(x_0)\cdot c'(0)=\nabla f(x_0)\cdot v$.

Thus, $\nabla f(x_0)\perp v$.

Since $v\in T_pS$ was arbitrary, $\nabla f(x_0)\perp T_{x_0}S$, $\nabla g(x_0)\perp T_{x_0}S$.

Thus, $\nabla f(x_0)\parallel\nabla g(x_0)$.

Thus, $\exists\lambda\in\mathbb{R}$ s.t. $\nabla f(x_0)=\lambda\nabla g(x_0)$. ∎

Corollary $S=g_1^{-1}(c_0)\cap\cdots\cap g_k^{-1}(c_0)$, $\nabla g_1(x_0),\cdots,\nabla g_k(x_0)$ linearly independent.

$f|_S$ has maximum or minimum at x_0.

$\Rightarrow \exists\lambda_1,\cdots,\lambda_k\in\mathbb{R}$ s.t. $\nabla f(x_0)=\lambda_1\nabla g_1(x_0)+\cdots+\lambda_k\nabla g_k(x_0)$.

Def. Lagrangian

$$L(w,B)=f(w)+\sum_{i=1}^{\ell}B_i h_i(w).$$

$\frac{\partial L}{\partial w_i}=0$, $\frac{\partial L}{\partial B_i}=0$ 을 통해 w, B 찾기

$\Leftrightarrow \min_w f(w)$ s.t. $h_i(w)=0,\ i=1,\cdots,\ell$ with Lagrangian Multiplier Theorem.

($\because$ ① $\frac{\partial L}{\partial B_i}=0 \Leftrightarrow \underbrace{h_i(w)=0,\ \forall i\in\{1,\cdots,\ell\}}_{\text{constraint (a)}}$ / ② $\frac{\partial L}{\partial w_i}=0 \Leftrightarrow \frac{\partial f}{\partial w_i}+\sum_{j=1}^{\ell}B_j\frac{\partial h_j(w)}{\partial w_i}$, $\forall i\in\{1,\cdots,n\}$

$\nabla f(w)=\left(\frac{\partial f}{\partial w_1},\cdots,\frac{\partial f}{\partial w_n}\right)$, $\nabla h_j(w)=\left(\frac{\partial h_j}{\partial w_1},\cdots,\frac{\partial h_j}{\partial w_n}\right)$.

By ②, $\underbrace{\nabla f(w)+B_1\nabla h_1(w)+\cdots+B_\ell\nabla h_\ell(w)=0.\ \ \text{Set } B_j=-\lambda_j,\ \forall j\in\{1,\cdots,\ell\}}_{\text{Lagrange Multiplier Theorem (b)}}$ (a),(b) 이용)

Relation of Primal/Dual Optimization Problem

Def. Primal optimization problem

$\min_w f(w)$ s.t. $g_i(w) \le 0,\ i=1,\cdots,k,\quad h_i(w)=0,\ i=1,\cdots,\ell.$

Def. generalized Lagrangian

$L(w,\alpha,\beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{\ell} \beta_i h_i(w)$

$\theta_P(w) := \max_{\alpha,\beta,\alpha_i \ge 0} L(w,\alpha,\beta) = \max_{\alpha,\beta,\alpha_i\ge 0} \left(f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{\ell} \beta_i h_i(w) \right)$

Then, $\theta_P(w) = \begin{cases} f(w), & \text{if } g_i(w)\le 0,\ i=1,\cdots,k,\ \ h_i(w)=0,\ i=1,\cdots,\ell. \\ \infty, & \text{o.w.} \end{cases}$

Straightforward!

Thus, $\min_w f(w)$ s.t. $g_i(w)\le 0,\ i=1,\cdots,k,\quad h_i(w)=0,\ i=1,\cdots,\ell.$

$\Leftrightarrow \min_w \theta_P(w) = \min_w \max_{\alpha,\beta,\alpha_i\ge 0} L(w,\alpha,\beta)$

Def. Dual optimization problem

$\max_{\alpha,\beta,\alpha_i\ge 0} \min_w L(w,\alpha,\beta)$.

cf) $f(z,w) = \sin(z+w)$ equality not hold

LHS $= -1$ $\left(w := -z - \frac{\pi}{2}\right)$

RHS $= 1$ $\left(z := -w + \frac{\pi}{2}\right)$

Lemma. Minimax Inequality.

domain Z 자정의 성질

For any function $f: Z \times W \to \mathbb{R}$, $\sup_{z\in Z} \inf_{w\in W} f(z,w) \le \inf_{w\in W} \sup_{z\in Z} f(z,w)$.

pf) Fix $z\in Z$. Then, $\inf_{w\in W} f(z,w) \le f(z,w),\ \forall w\in W.$ ($\because$ def of infimum) $\cdots$ ①

Fix $w\in W$. Then, $f(z,w) \le \sup_{z\in Z} f(z,w),\ \forall z\in Z.$ ($\because$ def of supremum) $\cdots$ ②

Then, $\inf_{w\in W} f(z,w) \le \sup_{z\in Z} f(z,w),\ \forall w\in W, \forall z\in Z$

least upper bound

Then, $\forall w\in W$, $\sup_{z\in Z} f(z,w)$ is upper bound of $\inf_{w\in W} f(z,w)$. $\Rightarrow \sup_{z\in Z}\inf_{w\in W} f(z,w) \le \sup_{z\in Z} f(z,w)$.

Also, $\sup_{z\in Z}\inf_{w\in W} f(z,w)$ is lower bound of $\sup_{z\in Z} f(z,w)$. $\Rightarrow \sup_{z\in Z}\inf_{w\in W} f(z,w) \le \inf_{w\in W}\sup_{z\in Z} f(z,w)$.

Remark. By minimax inequality, $\boxed{d^*} := \max_{\alpha,\beta,\alpha_i\ge 0} \min_w L(w,\alpha,\beta) \le \boxed{p^*} = \min_w \max_{\alpha,\beta,\alpha_i\ge 0} L(w,\alpha,\beta) \Rightarrow$ weak duality

Def. Slater's Condition

The problem satisfies Slater's condition if it is strictly feasible,

보통의 general한 상황을 이야기하는 condition

i.e. $\exists w_0 \in D$ s.t. $g_i(w_0) < 0,\ i=1,\cdots,k,\quad h_i(w_0)=0,\ i=1,\cdots,\ell.$

domain of problem

Remark. If the Slater's condition is satisfied, then $d^* = p^* \Rightarrow$ Strong duality

($\because$ Strong duality via Slater Condition Theorem)

Theorem. Strong duality via Slater Condition

If the primal problem is convex, and satisfies the weak Slater's Condition, then strong duality holds.

optimal margin classifier의 domain은 convex!

pf) 못 찾았음.

KKT Conditions

Def. KKT conditions (Karush–Kuhn–Tucker conditions)

$$\frac{\partial}{\partial w_i} L(w,\alpha,\beta)\Big|_{w=w^*,\,\alpha=\alpha^*,\,\beta=\beta^*} = 0, \quad i=1,\cdots,d. \qquad (1)$$

$$\frac{\partial}{\partial \beta_i} L(w,\alpha,\beta)\Big|_{w=w^*,\,\alpha=\alpha^*,\,\beta=\beta^*} = 0, \quad i=1,\cdots,\ell. \qquad (2)$$

$$\alpha_i^* g_i(w^*) = 0, \quad i=1,\cdots,K. \qquad (3)$$

$$g_i(w^*) \le 0, \quad i=1,\cdots,K. \qquad (4)$$

$$\alpha_i^* \ge 0, \quad i=1,\cdots,K. \qquad (5)$$

Theorem. For problem with strong duality (e.g. assume Slater conditions),

w^*, α^*, β^* satisfies KKT conditions $\Longleftrightarrow$ w^*, α^*, β^* are primal and dual solutions.

pf) $\Leftarrow$) $f(w^*) = \min_w L(w, \alpha^*, \beta^*)$ $\quad$ ($\because$ dual solutions)

$$= \min_w \left(f(w) + \sum_{i=1}^{K} \alpha_i^* g_i(w) + \sum_{i=1}^{\ell} \beta_i^* h_i(w) \right)$$

$$\le f(w^*) + \sum_{i=1}^{K} \alpha_i^* g_i(w^*) + \sum_{i=1}^{\ell} \beta_i^* h_i(w^*)$$

$$\le f(w^*) \quad (\because \alpha_i^* \ge 0,\ g_i(w^*) \le 0,\ \forall i \in \{1,\cdots,K\},\quad h_i(w^*) = 0,\ \forall i \in \{1,\cdots,\ell\}.)$$

Thus, all inequalities are actually equality.

Thus, straightforwardly, w^*, α^*, β^* satisfies KKT conditions

$$\Rightarrow \min_w L(w, \alpha^*, \beta^*) = f(w^*) + \sum_{i=1}^{m} \alpha_i^* g_i(w^*) + \sum_{i=1}^{\ell} \beta_i^* h_i(w^*) = f(w^*) \quad (\because \text{KKT conditions})$$

Thus, w^*, α^*, β^* are primal and dual solutions.

언제나 $d^* \le p^*$이다, $d^* = p^*$ 조건이 있고, $p^* = L(w^*, \alpha^*, \beta^*)$이므로 자동적으로 d^* 만족시키는 경우 w^*, α^*, β^*에서 $L(w,\alpha,\beta)$

Remark. If WTS w^*, α^*, β^* are primal and dual solutions,

ETS w^*, α^*, β^* satisfies KKT conditions.

Apply to Optimal Margin Classifier

Ex. (Optimal Margin Classifier)

$\forall i \in [n]$, $x^{(i)} \in \mathbb{R}^d$, $y^{(i)} \in \{-1, 1\}$, $w = (w_1, \cdots, w_d)' \in \mathbb{R}^d$, $b \in \mathbb{R}$.

Solve $\min_{w,b} \frac{1}{2}\|w\|^2$ s.t. $y^{(i)}(w^T x^{(i)} + b) \geq 1$, $i = 1, \cdots, n$, which is primal optimization problem.

$\Leftrightarrow \min_{w,b} \underbrace{\boxed{\frac{1}{2}\|w\|^2}}_{f(w,b)}$ s.t. $\underbrace{\boxed{-y^{(i)}(w^T x^{(i)} + b) + 1}}_{g_i(w,b)} \leq 0$, $i = 1, \cdots, n$. $f, g_i : \mathbb{R}^d \times \mathbb{R} (\cong \mathbb{R}^{d+1}) \to \mathbb{R}$

Define generalized Lagrangian $L(w, b, \alpha) := \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y^{(i)}(w^T x^{(i)} + b) - 1)$

Since convex and satisfies Slater's condition, then strong duality holds.

Thus, ETS w^*, b^*, α^* satisfies KKT conditions.

(+) KKT Conditions 만으로 이 문제에서는 minima들의 (w^*, α^*, b^*) closed form을 전부 찾기 어려움

(+) (의의)

KKT Conditions를 통해 위 문제와 동치가 되는 dual optimization problem을 다음처럼 쓸 수 있음.

$\max_{\alpha} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle \right)$ s.t. $\alpha_i \geq 0$, $i = 1, \cdots, n$, $\sum_{i=1}^{n} \alpha_i y^{(i)} = 0$.

⇒ Input data들의 inner product 형태로 표현된 깔끔한 이 식의 structure 확인 가능. (알고리즘 할 때 유용)

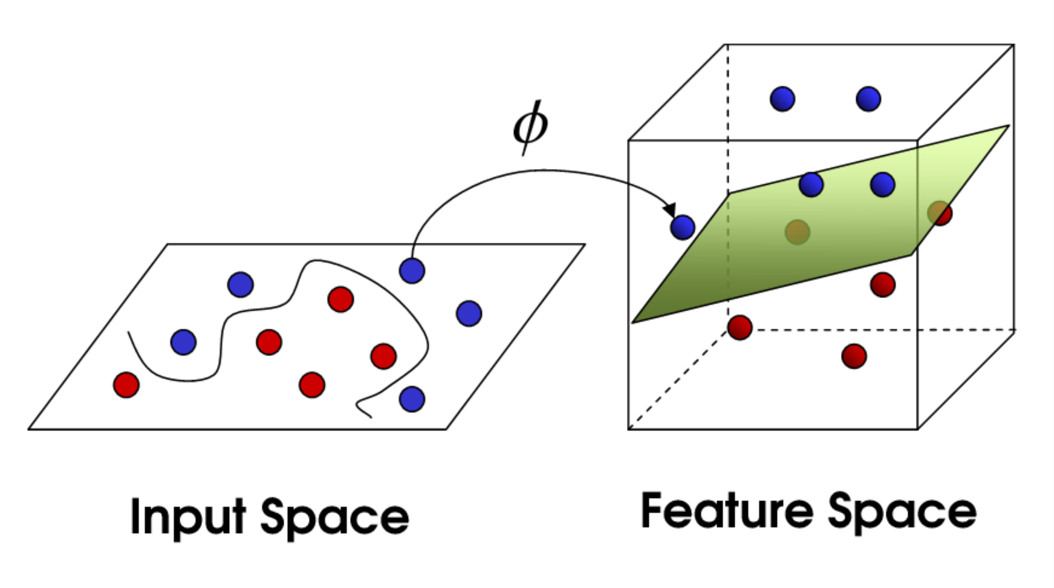
3

Kernels

Dual solution is in form of inner product!

Kernels

- 만약 데이터가 선형적으로 분리할 수 없는 형태라면, 데이터를 고차원으로 mapping을 시켜서, 고차원에서 SVM을 사용하면 된다.
- 하지만 이러한 방법에는 두 가지 문제점이 존재한다
  - 고차원에서 계산을 수행하므로 계산 비용이 증가한다
  - 어떠한 Basis function을 사용해야 데이터들이 선형적으로 분리 가능한지에 대한 사전 지식이 없을 수 있다.
- 이러한 문제점들을 해결한 것이 바로 Kernel Trick이다.



Kernels

- Kernel은 다음과 같이 정의된다:
  - $k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$
  - 즉 고차원으로 매핑했을 때, 데이터 간의 내적이 바로 커널인 것이다.
- Kernel의 장점
  1. 하나의 커널이 여러 개의 고차원 공간을 대표한다. 따라서 하나의 커널을 살펴보는 것이 여러 개의 공간을 한번에 살펴보는 효과를 가져온다
  2. 일반적으로 데이터를 무한 차원으로 매핑하는 건 불가능한데, 이러한 무한 차원에서 사용되는 내적값을 나타낼 수도 있다 (Gaussian Kernel). 어떠한 유한 차원의 데이터를 무한 차원으로 매핑하게 되면, 높은 확률로 선형적으로 분리 가능하게 된다.
- SVM에서는 데이터의 내적의 형태만을 사용하게 되는데, 그러면 굳이 '데이터 => 변환 => 내적'의 단계를 거치기보다, 바로 '데이터 => 내적'으로 가고자 하는 것이 바로 Kernel trick이다.

Advantages of Kernels

제곱꼴도 Kernel이다

$$k(\mathbf{x},\mathbf{z}) = (\mathbf{x}^T\mathbf{z})^2 = (x_1z_1 + x_2z_2)^2$$
$$= x_1^2z_1^2 + 2x_1z_1x_2z_2 + x_2^2z_2^2$$

- 다음의 내적과 같으며 해가 유일하지 않다

$$(x_1^2, \sqrt{2}x_1x_2, x_2^2) \text{ and } (z_1^2, \sqrt{2}z_1z_2, z_2^2)$$

$$(x_1^2, x_1x_2, x_1x_2, x_2^2) \text{ and } (z_1^2, z_1z_2, z_1z_2, z_2^2)$$

- 따라서 하나의 커널을 사용함으로써 여러 개의 고차원 공간에 대한 탐색을 한번에 할 수 있다.
- 만약 제곱꼴을 커널로 사용했을 때, optimization이 안 된다면, 그와 관련된 모든 basis function으로는 데이터를 선형적으로 분리할 수 없다는 사실을 알 수 있다.

가우시안 커널(x와 z가 1차원)

- 아래의 수식은 가우시안 커널이 무한 차원의 공간에서의 내적임을 보이는 수식이다.

- $e^{-\frac{(x-z)^2}{2}} = e^{-\frac{x^2}{2}}e^{-\frac{z^2}{2}}e^{xz} = e^{-\frac{x^2}{2}}e^{-\frac{z^2}{2}}\left(1 + xz + \frac{x^2z^2}{2!} + \frac{x^3z^3}{3!} + \cdots\right)$

- $\phi(x) = e^{-\frac{x^2}{2}}\begin{bmatrix} 1 \\ x \\ \frac{x^2}{\sqrt{2!}} \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$; infinite dimensional

Optimization Using Kernel

- SVM에서의 $Optimal\ Hyperplane$을 찾기 위해선 원래는 다음의 $Primal\ Optimization$을 풀어야 한다
    - $min_{w,b}\frac{1}{2}||w||^2, s.t\ y^{(i)}\left(w^T\phi(x^{(i)})+b\right)\geq 1, for\ i=1,\ldots,n$
- 하지만 이러한 형태는 $basis\ function\ \phi$를 정의해야만 하고 고차원에서 데이터를 다루어야 하는 단점이 존재한다
- 그래서 $Dual\ Optimization$을 사용할 것이다.
    - SVM은 다행히 Slater's condition을 만족하기 때문에 Dual Solution과 Primal Solution이 같다 ($strong\ duality$)
    - $KKT\ condition$을 활용하여 $Dual\ Problem$을 나타내면 다음과 같으며, 여기서 주목할 점은 $Dual\ Problem$에서는 ϕ를 정의할 필요 없이 바로 커널만을 사용할 수 있다는 것이다.

$$max_\alpha\ W(\alpha)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}y^{(i)}y^{(j)}\alpha_i\alpha_j\boxed{<\phi(x^{(i)}),\phi(x^{(j)})>.}$$

$$s.t.\alpha_i\geq 0, i=\ 1,\ldots,n$$

$$\sum_{i=1}^{n}\alpha_i y^{(i)}=0$$

- 이 부분을 커널로 대체

4

Code

Data

In [ ]:
```
import numpy as np

q4_data = np.load('q4_data/q4_data.npy', allow_pickle=True).item()
x_train = q4_data['q4x_train']
y_train = q4_data['q4y_train'].flatten()
x_test = q4_data['q4x_test']
y_test = q4_data['q4y_test'].flatten()
```

In [2]:
```
# Hyperparameters
lr = 1e-3
C = 1
num_epochs = 2000

print('Number of training data:', x_train.shape[0])
print('Number of test data    :', x_test.shape[0])
print('Feature dimension      :', x_test.shape[1])
```

```
Number of training data: 76
Number of test data    : 24
Feature dimension      : 4
```

Train Data

- 76개의 4차원 데이터, +1, -1로 라벨링

Test Data

- 24개의 4차원 데이터

BGD Training

```
In [3]: def svm_train_bgd(x, y, C, lr, num_epochs):
            N, D = x.shape
            w = np.zeros(D)
            b = 0

            for i in range(num_epochs):
                indicator = 1 - y*(x@w + b)
                indicator[indicator < 0] = 0
                indicator[indicator > 0] = 1
                w_grad = w - C * x.T @ (indicator * y)
                b_grad = -C * np.sum(indicator * y)
                w = w -  lr * w_grad
                b = b -  lr * b_grad

            return w, b
```

Input/Return 설명

- x : training data / y : training labels (1 or -1)
- C : slack cost / lr : learning rate
- num_epochs: The number of training epochs
- w : weights / b : bias

Code 설명

- Loss를 최소화 하는 w, b를 Batch Gradient Descent 사용해서 찾는다

$$\min_{\mathbf{w},b} L(\mathbf{w},b) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\max\left(0, 1 - y^{(i)}\left(\mathbf{w}^\top\mathbf{x}^{(i)} + b\right)\right)$$

$$\nabla_{\mathbf{w}} L(\mathbf{w},b) = \mathbf{w} - C\sum_{i=1}^{N}\mathbb{I}\left[1 - y^{(i)}\left(\mathbf{w}^\top\mathbf{x}^{(i)} + b\right) \geq 0\right] y^{(i)}\mathbf{x}^{(i)},$$

$$\nabla_{b} L(\mathbf{w},b) = -C\sum_{i=1}^{N}\mathbb{I}\left[1 - y^{(i)}\left(\mathbf{w}^\top\mathbf{x}^{(i)} + b\right) \geq 0\right] y^{(i)},$$

SGD Training

In [4]:
```
def svm_train_sgd(x, y, C, lr, num_epochs):
    N, D = x.shape
    w = np.zeros(D)
    b = 0

    for i in range(num_epochs):
        for j in range(N):
            indicator = 1 - y[j] * (w@x[j] + b)
            indicator = 1 if indicator > 0 else 0
            w_grad = (1/N) * w - C * indicator * y[j] * x[j]
            b_grad = -C * indicator * y[j]
            w = w - lr * w_grad
            b = b - lr * b_grad

    return w, b
```

Input/Return 설명

- BGD와 동일하다

Code 설명

- Loss를 최소화 하는 w, b를 Stochastic Gradient Descent 사용해서 찾는다

$$L^{(i)}(\mathbf{w}, b) = \frac{1}{2N}\|\mathbf{w}\|^2 + C \cdot \max\left(0, 1 - y^{(i)}\left(\mathbf{w}^\top \mathbf{x}^{(i)} + b\right)\right)$$

$$\nabla_w L(w, b) = w - C\sum_{i=1}^{N} \mathbb{I}\left(1 - y^{(i)}(w^T x^{(i)} + b) \geq 0\right) \cdot y^{(i)} x^{(i)}$$

$$\nabla_b L(w, b) = -C\sum_{i=1}^{N} \mathbb{I}\left(1 - y^{(i)}(w^T x^{(i)} + b) \geq 0\right) \cdot y^{(i)}$$

SVM Prediction

In [5]:
```
def svm_predict(x, w, b):

    y_pred = np.ones(x.shape[0])
    result = x@w + b
    y_pred[result < 0] = -1

    return y_pred
```

In [6]:
```
def evaluate(preds, labels):

    return (preds == labels).mean()
```

Code 설명

- x: 우리가 예측하고 싶은 데이터
- w, b : 우리가 학습한 파라미터

Classification Rule

- x@w + b 가 0보다 크거나 같다면 1
- 0보다 작으면 -1로 예측
- y_pred: 예측값

$$h(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$$

Classification rule:

- $y = +1$ if $h(\mathbf{x}) \geq 0$
- $y = -1$ otherwise

Result

```
In [7]: # Train SVM with BGD
w, b = svm_train_bgd(x_train, y_train, C, lr, num_epochs)
print('Weights', w)
print('Bias    ', b)

# Test accuracy
preds_test = svm_predict(x_test, w, b)
test_acc = evaluate(preds_test, y_test)
print('BGD test accuracy: {:5.2f}%'.format(100.*test_acc))
```

```
Weights [-0.18558916 -0.33076923  0.85189816  0.70158907]
Bias     -0.7170000000000005
BGD test accuracy: 95.83%
```

```
In [8]: # Train SVM with SGD
w, b = svm_train_sgd(x_train, y_train, C, lr, num_epochs)
print('Weights', w)
print('Bias    ', b)

# Test accuracy
preds_test = svm_predict(x_test, w, b)
test_acc = evaluate(preds_test, y_test)
print('SGD test accuracy: {:5.2f}%'.format(100.*test_acc))
```

```
Weights [-0.18895699 -0.34671336  0.84538791  0.76344282]
Bias     -0.7350000000000005
SGD test accuracy: 95.83%
```

결과 해석

- BGD와 SGD로 구한 w, b 값은 다르지만,
- 둘 다 Test Accuracy가 95.83%이다
- Learning Rate와 epoch 수를 조절하면 Accuracy 상향이 가능하다

감사합니다