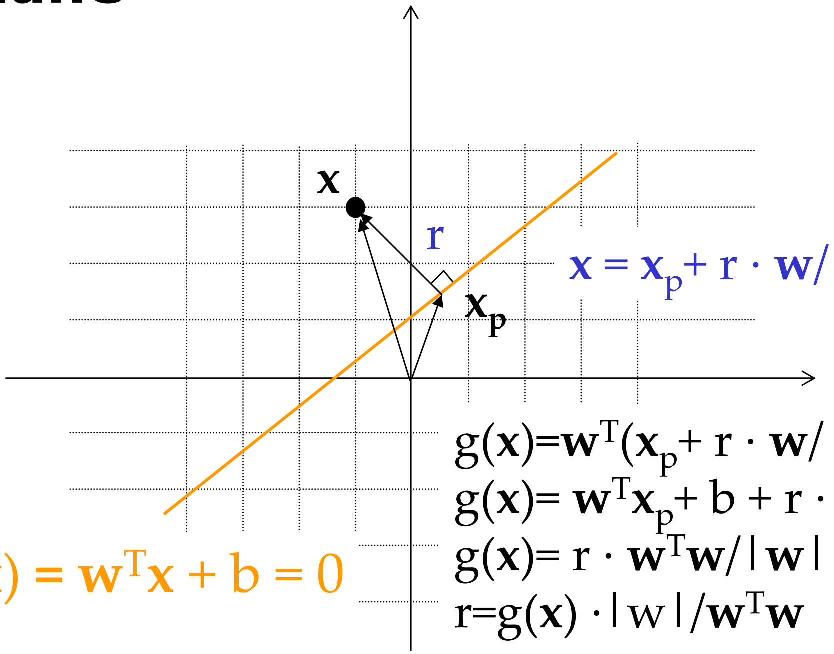
Machine Learning

Ch 09. Support Vector Machine (SVM)

CONTENTS

✓ Background

➢ Distance from a Sample to the Optimal Hyperplane



$\mathbf{x} = \mathbf{x}_p + r \cdot \mathbf{w}/|\mathbf{w}|$

$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0$

$g(\mathbf{x}) = \mathbf{w}^T(\mathbf{x}_p + r \cdot \mathbf{w}/|\mathbf{w}|) + b$

$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x}_p + b + r \cdot \mathbf{w}^T\mathbf{w}/|\mathbf{w}|$

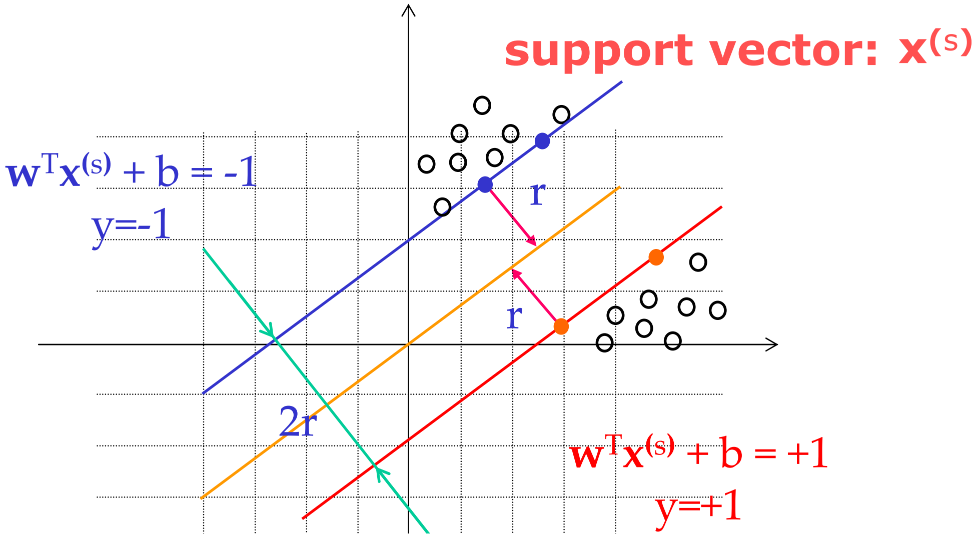
$g(\mathbf{x}) = r \cdot \mathbf{w}^T\mathbf{w}/|\mathbf{w}|$

$r = g(\mathbf{x}) \cdot |w|/\mathbf{w}^T\mathbf{w}$

$\boxed{r = g(\mathbf{x}) / |\mathbf{w}|}$

✓ Background

➢ **Support vector 의미와, SVM의 목표**



$r = g(\mathbf{x}^{(s)}) / |\mathbf{w}|$

where, $g(\mathbf{x}^{(s)}) = \mathbf{w}^T\mathbf{x}^{(s)} + b = \pm 1$ for $y^{(s)} = \pm 1$

margin: $2r = 2 / |\mathbf{w}|$ $\quad \left| \frac{1}{\|\mathbf{w}\|} - \frac{-1}{\|\mathbf{w}\|} \right| = \frac{2}{\|\mathbf{w}\|}$

✓ Background

➢ **Support vector 의미와, SVM의 목표**

margin: $\left| \frac{1}{\|\mathbf{w}\|} - \frac{-1}{\|\mathbf{w}\|} \right| = \frac{2}{\|\mathbf{w}\|}$

$\mathbf{w}^T\mathbf{x} + b \leq -1, \quad y=-1$

$\mathbf{w}^T\mathbf{x} + b \geq +1, \quad y=+1$

$$\underset{w,b}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{subject to} \quad y_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1, \quad i = 1, \ldots, n.$$

Primal 문제

✓ Hard margin Support Vector Machine

➢ **Lagrange multiplier method (라그랑주 승수법)**

- 제약조건이 있을 때 유용하게 사용 가능.
- Primal 문제 → Dual 문제 (Optimization)
  - 두 해가 같으려면 Karush–Kuhn–Tucker (KKT) 조건이 성립해야함.

- Dual 문제 : $L_P(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1]$

- KKT 조건
  - $\frac{dLp}{d\mathbf{w}} = 0$ $\quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$
  - $\frac{dLp}{db} = 0$ $\quad 0 = \sum_{i=1}^{n} \alpha_i y_i \qquad (\alpha_i \geq 0)$
  - $a_i[1 - y_i(w^T x_i + b)] = 0$

✓ Hard margin Support Vector Machine

➢ **Lagrange multiplier method (라그랑주 승수법)**

- $L_P(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1]$

$$= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{w}^T\boldsymbol{x}_i - b\sum_{i=1}^{n}\alpha_i y_i + \sum_{i=1}^{n}\alpha_i = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j$$

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

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j$$

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j \qquad \boldsymbol{w} = \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i$$

$$b = \sum_{i=1}^{n} y_i - \sum_{j=1}^{n}\alpha_j y_j x_i^T x_j$$

✓ How about?

➢ **에러가 있는 경우**

① 선형이라고 가정하고 에러 일부를 인정.

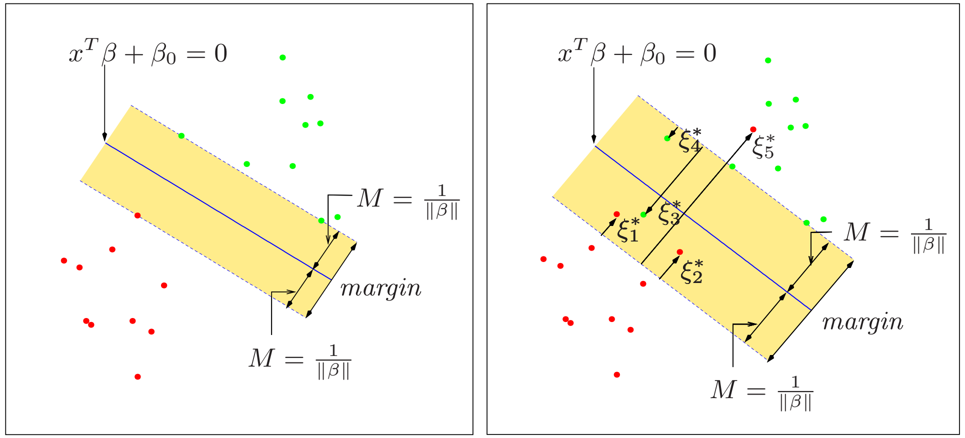
- Soft margin Support Vector Machine
- Decision Boundary는 여전히 선형

② Decision Boundary가 애초에 선형이 아닌 경우. (에러 x)

- Kernel Trick이 필요. (Support Vector Machine이 유명해진 이유.)

✓ Soft margin Support Vector Machine

➤ 아이디어



$$
\begin{aligned}
y_i(x_i^T\beta + \beta_0) &\geq M - \xi_i, \\
&\text{or} \qquad\qquad\qquad \forall i,\ \xi_i \geq 0,\ \textstyle\sum_{i=1}^N \xi_i \leq \text{constant}. \\
y_i(x_i^T\beta + \beta_0) &\geq M(1 - \xi_i),
\end{aligned}
$$

$$
\min \|\beta\| \quad \text{subject to} \begin{cases} y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i \ \forall i, \\ \xi_i \geq 0,\ \sum \xi_i \leq \text{constant}. \end{cases}
$$

✓ Soft margin Support Vector Machine

➢ **Hard margin과 마찬가지로 라그랑주 승수법을 적용해서 정리하면**

- $\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i \quad \text{subject to} \quad \xi_i \ge 0,\ y_i(x_i^T\beta+\beta_0) \ge 1-\xi_i\ \forall i,$
- $L_P = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i[y_i(x_i^T\beta+\beta_0)-(1-\xi_i)] - \sum_{i=1}^{N}\mu_i\xi_i,$

$$\beta = \sum_{i=1}^{N}\alpha_i y_i x_i,$$
$$0 = \sum_{i=1}^{N}\alpha_i y_i,$$
$$\alpha_i = C - \mu_i,\ \forall i,$$

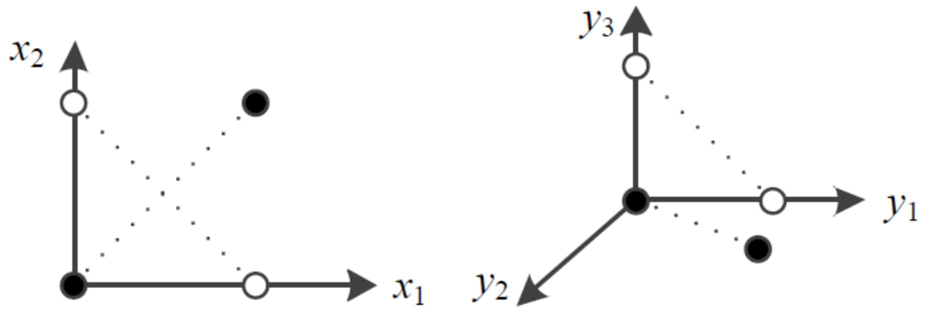
- $L_D = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N}\alpha_i\alpha_{i'}y_i y_{i'}x_i^T x_{i'},$

maximize L_D subject to $0 \le \alpha_i \le C$ and $\sum_{i=1}^{N}\alpha_i y_i = 0.$

✓ With Kernel Trick

➢ Kernel Function

$$K(x_i, x_j) = \langle h(x), h(x_i) \rangle$$



(a) 원래 공간 $\mathbf{x}=(x_1, x_2)$ (b) 매핑된 공간 $\Phi(\mathbf{x}) = (y_1, y_2, y_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i y_{i'} x_i^T x_{i'},$$

➡

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i y_{i'} \langle h(x_i), h(x_{i'}) \rangle.$$

✓ 정리

- Hard margin 일 경우 현실에선 적용이 거의 불가능. 무로 자르듯이 나누어지는 경계는 기대하기 힘들다. 따라서 soft margin 을 적용해서 패널티를 적용하게 되는데, 데이터가 실질적으로 비선형일 경우엔 패널티를 적용하기 어렵고 다른방법을 사용해야 한다. 그 방법이 차원을 높이는 방법. 선형인 데이터를 차원을 높여서 비선형으로 분류할 수 있게 만든 다음 경계 선을 찾아주는 방법인데, 차원을 높일 때 무작정 높히게 되면 계산이 무지막지하게 복잡해진다. 따라서 이걸 보완해 주는 방법이 Kernel Trick. 이 방법을 이용해서 계산을 빠르게 함으로써 분류를 할 수 있게 해준다.