

19기 정규세션

ToBig's 18기 강연자

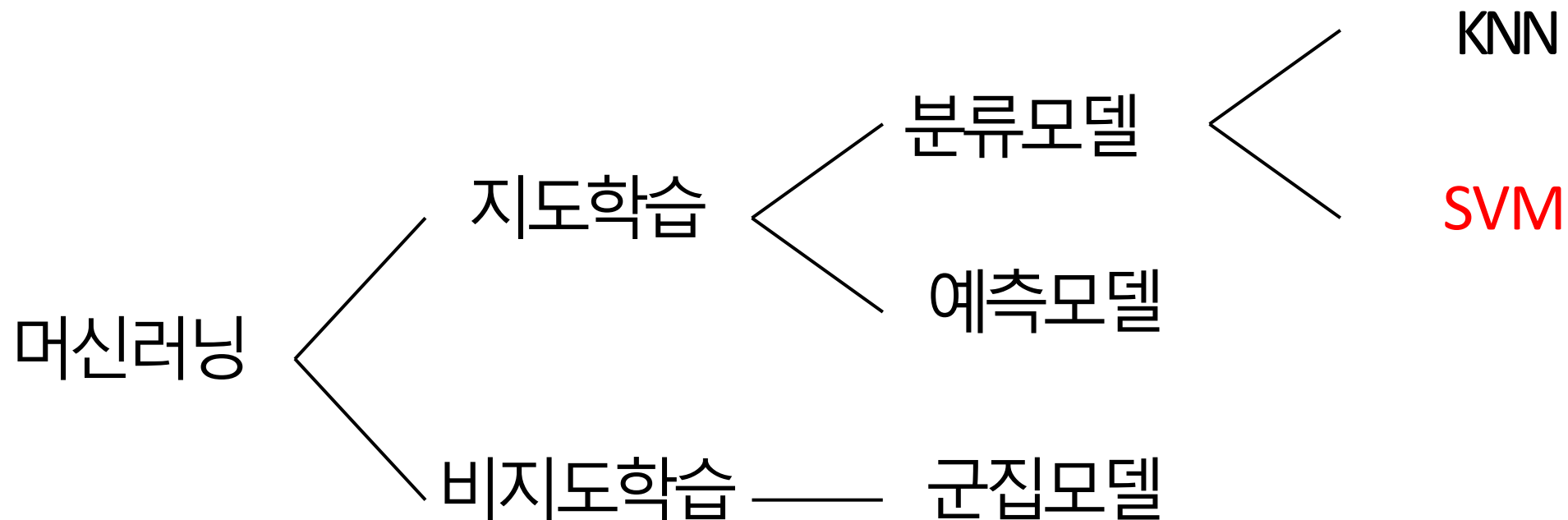
손유진

SVM

Support Vector Machine

Unit 01 | SVM

오늘 학습할 내용은?



Contents

Unit 01 | SVM

Unit 02 | Soft Margin SVM

Unit 03 | Non-Linear SVM

Unit 04 | Summary & 실습

Contents

Unit 01 | SVM

} Linear

Unit 02 | Soft Margin SVM

Unit 03 | Non-Linear SVM

Unit 04 | Summary & 실습

Unit 01 | SVM

Support Vector Machine

- : 주로 바이너리 분류를 위해 사용하는 기법 (회귀에 사용하는 경우 : SVR)
- : 딥러닝 이전에 높은 성능으로 주목받은 모델

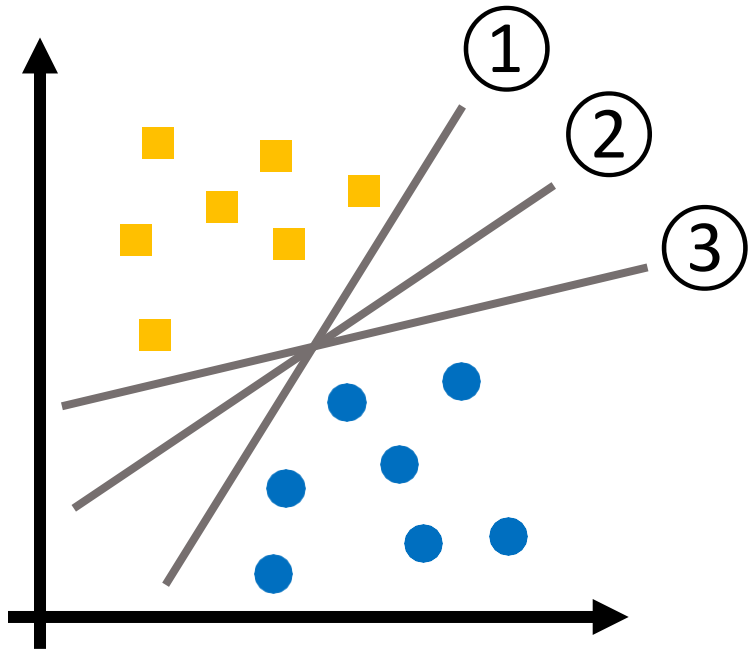
SVM의 분류

선형 여부	분류
선형	Linear svm
비선형	Non-linear svm

오분류허용 여부	분류
X	Hard margin svm
O	soft margin svm

Unit 01 | SVM

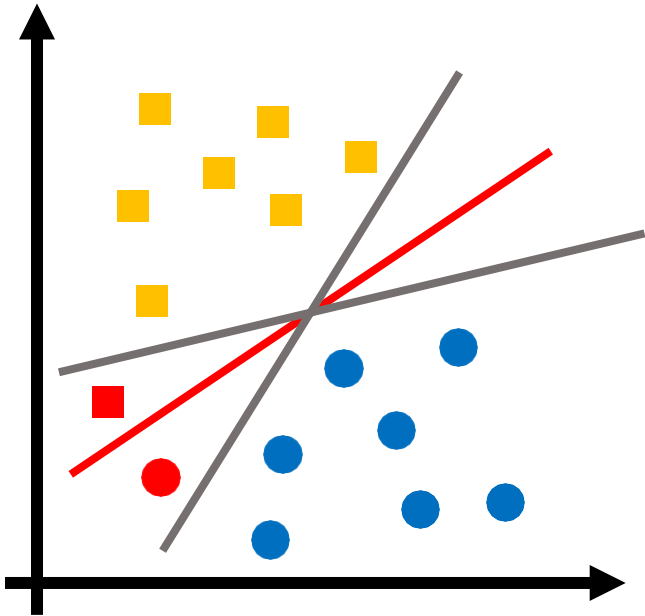
1.SVM?



SupportVector Machine관점에서,
데이터를 가장 잘 나누는 선은?

Unit 01 | SVM

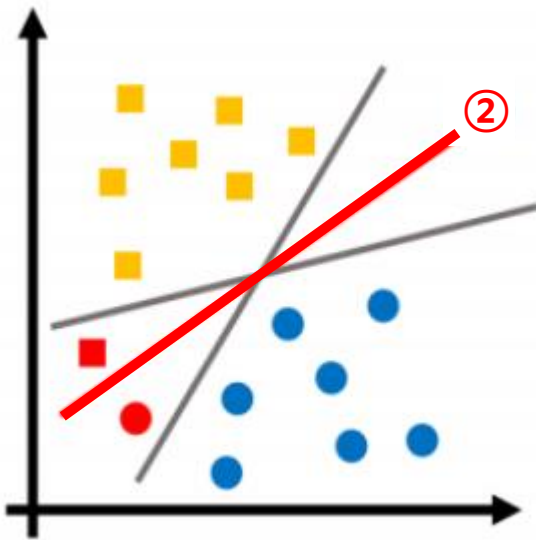
1.SVM?



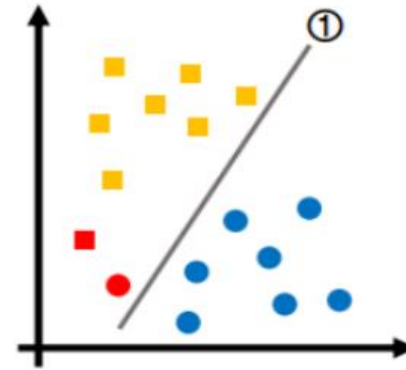
② 정답은 2번! 이유는?

Unit 01 | SVM

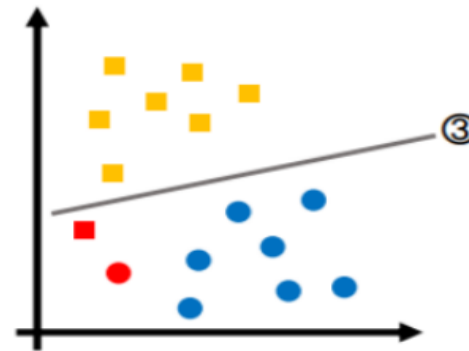
SVM (support vector machine)



새로운 점이 추가 되었을 경우



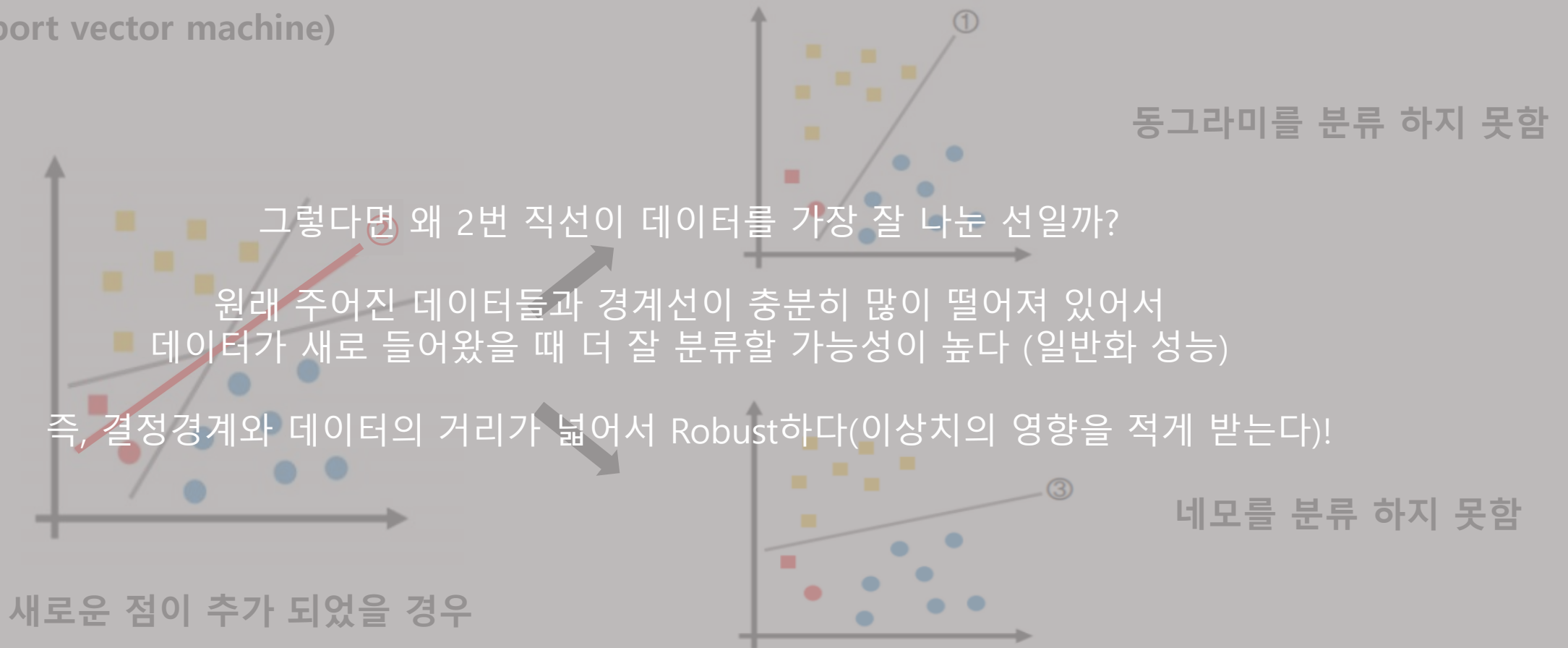
1. 동그라미를 분류 하지 못함(X)



3. 네모를 분류 하지 못함(X)

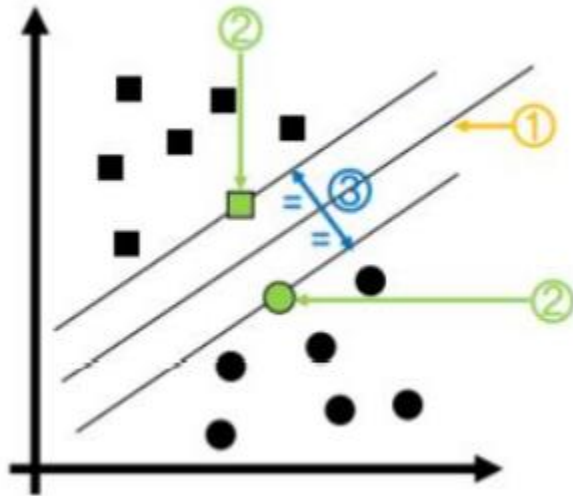
Unit 01 | SVM

SVM (support vector machine)



Unit 01 | SVM

SVM (support vector machine)



① Hyperplane

:여러 데이터를 나누는 기준이 되는 경계(초평면)

② Support Vector

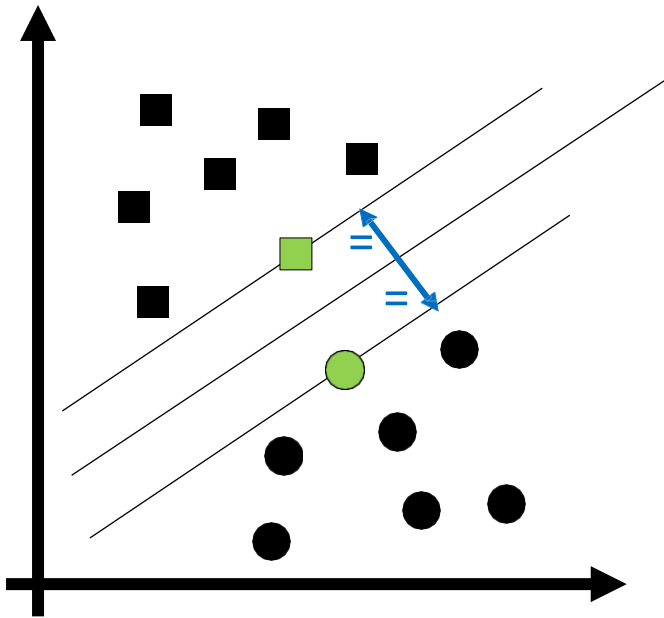
:Hyperplane과 가장 가까운 데이터

③ Margin

:결정경계와서포트 벡터 사이의거리 $\times 2$

Unit 01 | SVM

1.SVM- 정리1



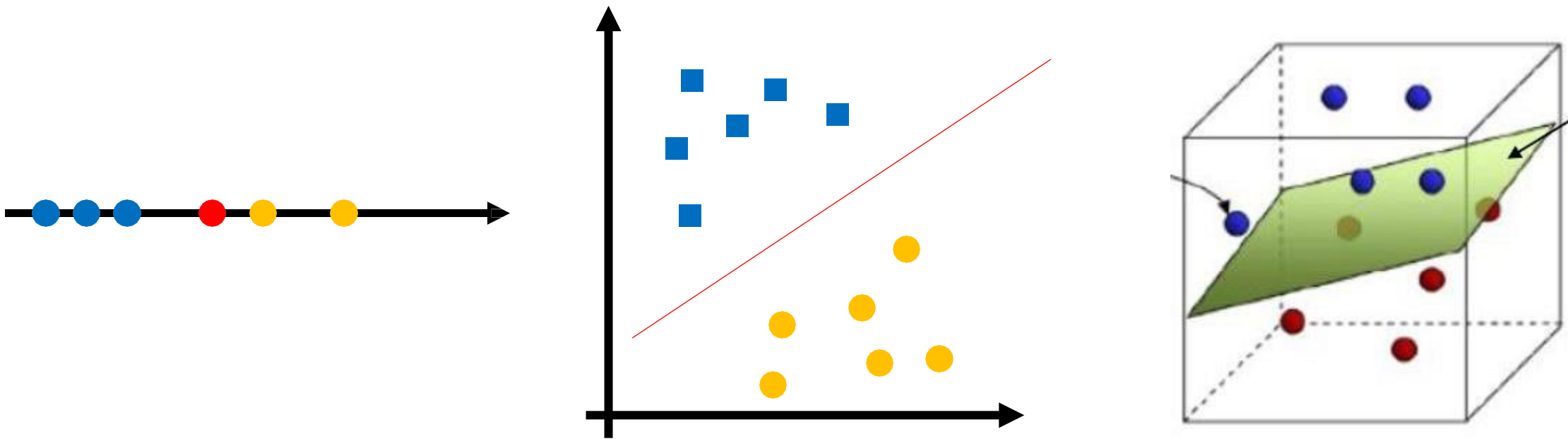
SVM의 발상?

: 마진을 구하는 방법을 공식화하고
이 마진을 최대화하는 결정 초평면
(decision hyperplane)을 찾자!

이제 수식으로 살펴볼까요?

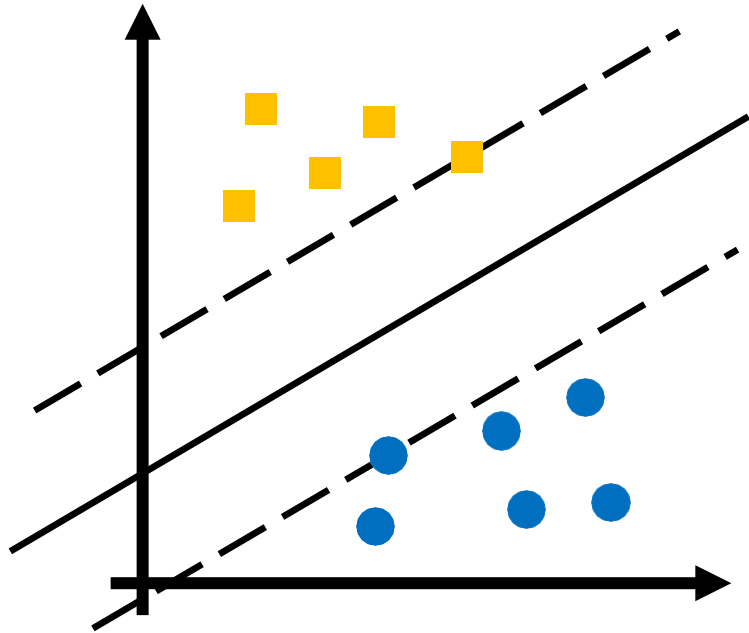
Unit 01 | SVM

1-1. 여러 차원에서 분류되는 모습

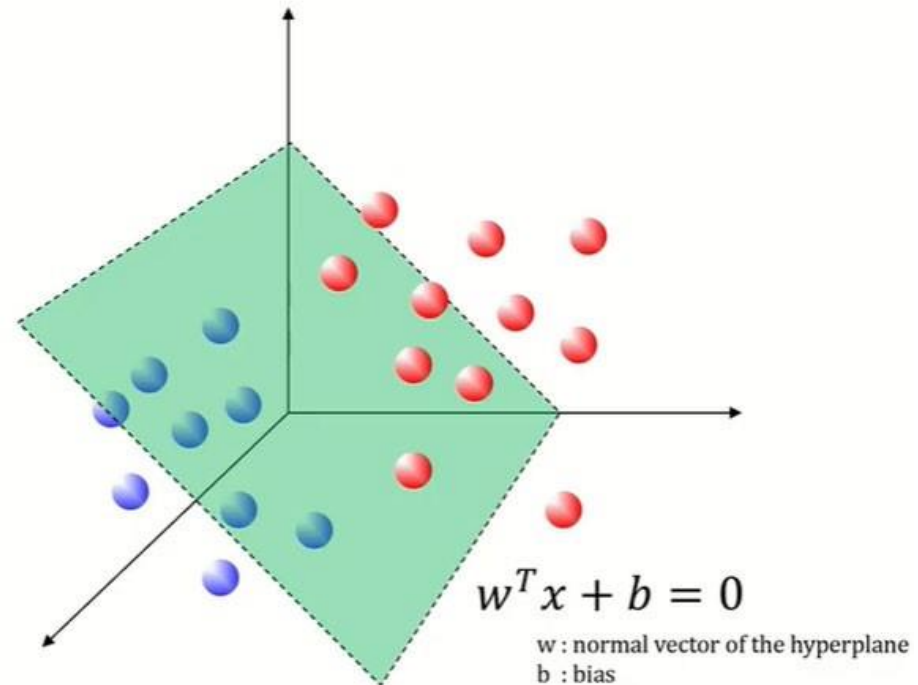


Unit 01 | SVM

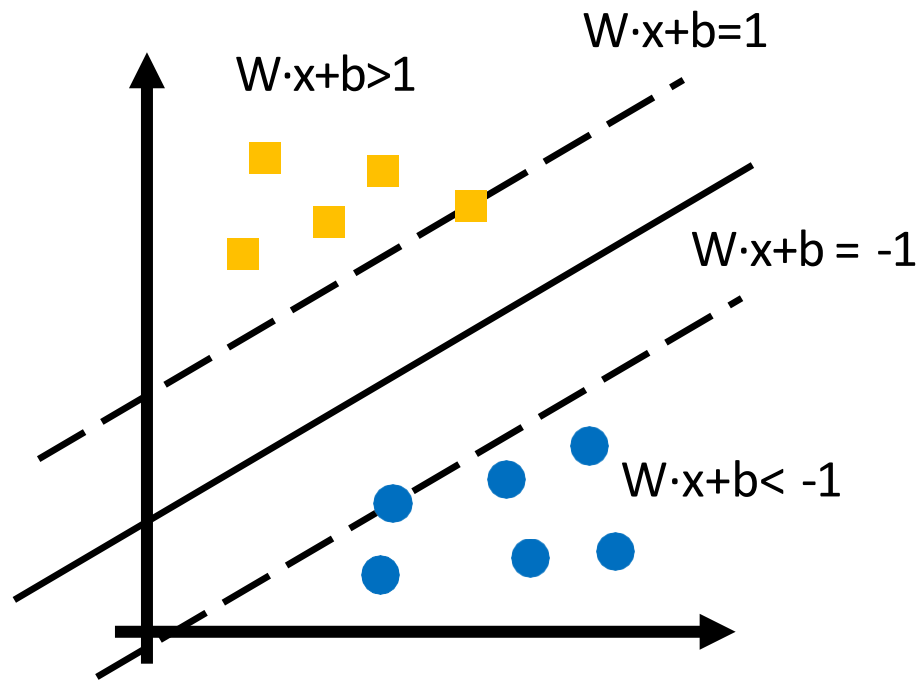
3. 수식으로 살펴보자



모든 plane은 $w \cdot x + b = 0$ 으로 표현할 수 있다!



Unit 01 | SVM

3-1. $y = \pm 1$ 분류 문제

편의상, $y = \pm 1$ 분류 문제

: \blacksquare ($+1$) class라면 $+1$ 이상,

- (-1) class라면 -1 이하의 값을 갖도록 하자

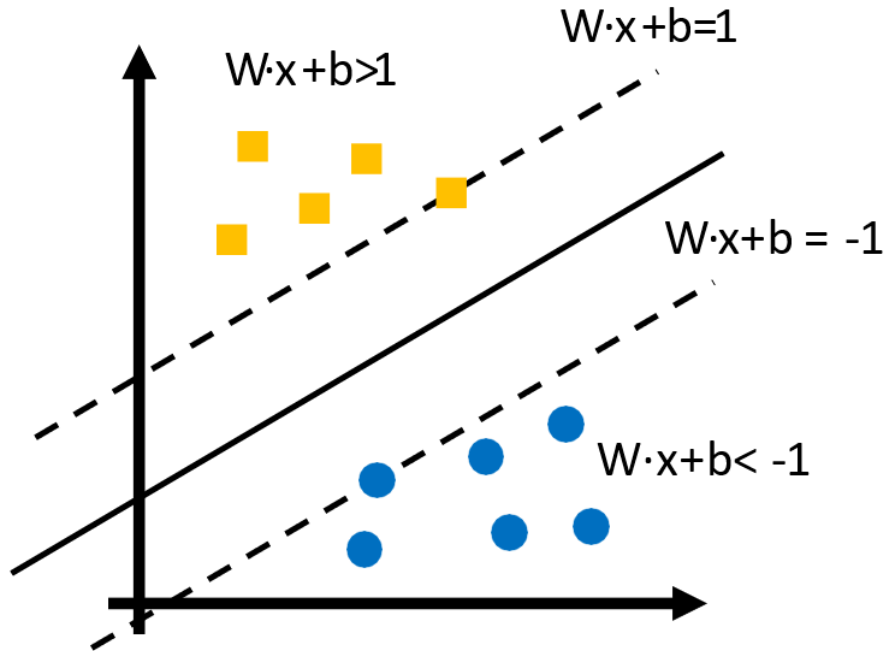
즉,

$$w \cdot x_+ + b \geq +1$$

$$w \cdot x_- + b \leq -1$$

Unit 01 | SVM

3-2. 우리의 결정규칙



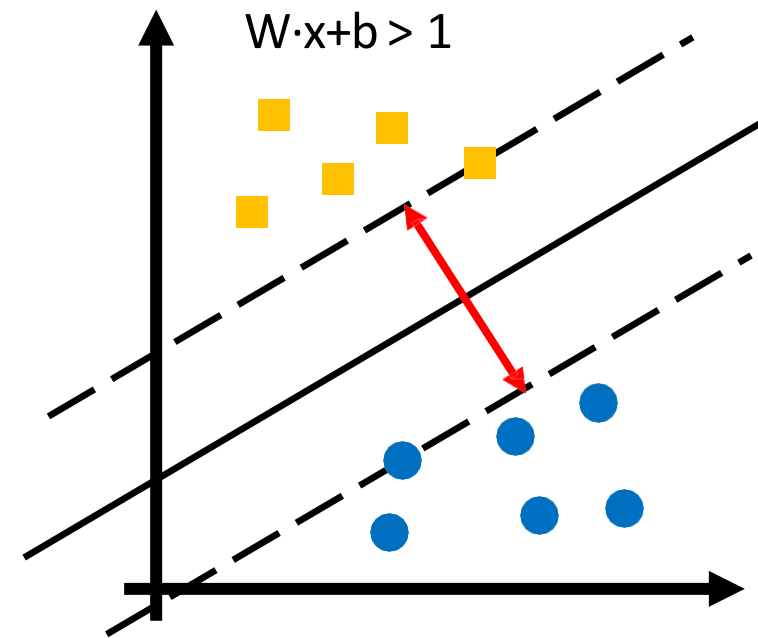
수식으로 나타내면?

$$y_i (w \cdot x_i + b) \geq 1$$

단, $y_i = \begin{cases} +1 & \text{for } \blacksquare \text{ sample} \\ -1 & \text{for } \bullet \text{ sample} \end{cases}$

Unit 01 | SVM

3-3.margin



$$x^+ = x^- + \lambda w$$

$$w^T x^+ + b = 1 \quad x^+ \text{가 plus-plane 위의 점}$$

$$w^T (x^- + \lambda w) + b = 1 \quad (x^+ = x^- + \lambda w)$$

$$w^T x^- + b + \lambda w^T w = 1$$

$$-1 + \lambda w^T w = 1 \quad x^- \text{는 minus-plane 위의 점}$$

$$\lambda = \frac{2}{w^T w}$$

The vector norm $\|W\|_p$ for $p = 1, 2, 3, \dots$

$$\|W\|_p = \left(\sum_i |w_i|^p \right)^{1/p} \quad \text{L}_2 \text{ norm} \quad \|W\|_2 = \left(\sum_i |w_i|^2 \right)^{1/2} = \sqrt{w_1^2 + w_2^2 + \dots + w_n^2} = \sqrt{W^T W}$$

길의너비(margin)

$$\begin{aligned} \text{Margin} &= \text{distance}(x^+, x^-) \\ &= \|x^+ - x^-\|_2 \\ &= \|(x^- + \lambda w) - x^-\|_2 \\ &= \|\lambda w\|_2 \\ &= \lambda \sqrt{w^T w} \\ &= \frac{2}{w^T w} \cdot \sqrt{w^T w} \\ &= \frac{2}{\sqrt{w^T w}} = \frac{2}{\|w\|_2} \end{aligned}$$

Unit 01 | SVM

지금까지의 수식을 정리하면,

1. 제약식(조건)

$$: y_i (w \cdot x_i + b) \geq 1$$

: 모든 데이터들은 결정경계 안에 잘 들어가 있어야 함

2. 목적식

: $\frac{2}{||w||}$ 가 최대가 되게 하고, 동시에 위의 제약식을 만족하는 w 와 b 를 찾자 -> margin이 최대

$$: \max\left(\frac{2}{||w||}\right) > \min(||w||) > \min\left(\frac{||w||^2}{2}\right) \quad (\text{계산의 편의를 위한 식 변형})$$

목적함수는 2차식이고, 제약식은 선형식이다 -> 2차계획법 -> convex optimization -> 전역최적해 존재

Unit 01 | SVM

3-4. 라그랑주 승수법1

: $\phi(x, y) = 0$ 이라는 제약이 있을 때, 최적화 문제를 푸는 방법

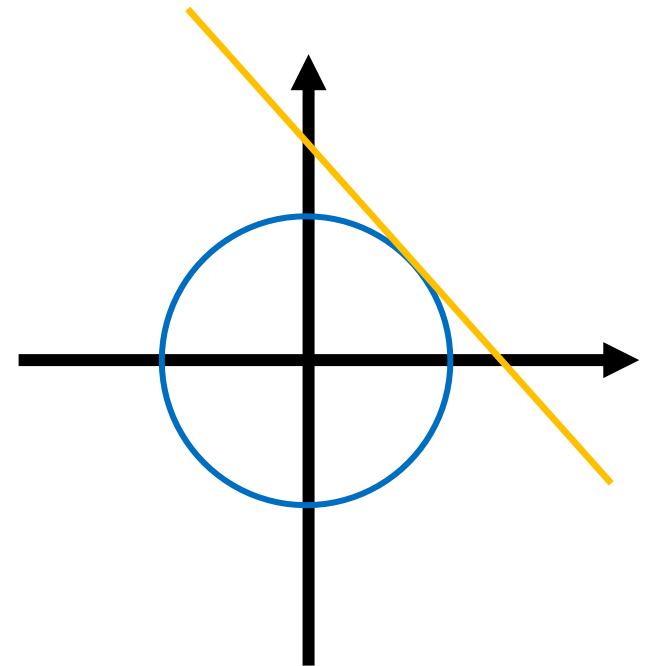
> “목적식(f) + 제약식(ϕ)”을 하나의 식으로 표현 가능

목적식 f 와 제약식 ϕ 의 그라디언트 방향이 같을 때, f 의 최적값

> $\nabla_{x,y} f = \lambda \nabla_{x,y} \phi$

> Let $L(x, y) = f(x, y) - \lambda \phi(x, y)$

> Then, $\nabla_{x,y,\lambda} L(x, y, \lambda) = 0$ 을 푸는 문제가 된다!



Unit 01 | SVM

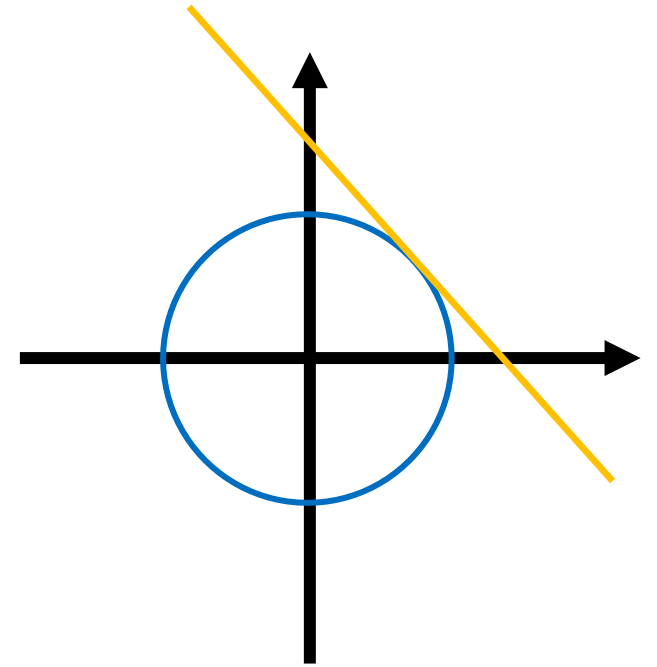
3-4. 라그랑주 승수법2

우리의 제약식과 목적식은?

1. 제약식: $y_i (w \cdot x_i + b) \geq 1$

2. 목적식: $\frac{\|w\|^2}{2}$ 의 최소화

> 문제는? 제약식이 '부등식'(라그랑주 승수법은 등식)



Unit 01 | SVM

3-5. KKT condition (Karush-Kuhn-Tucker 조건)

연립 부등식의 경우 라그랑주 승수법을 사용하되, 최적 값이기 위한 필요충분조건인 KKT조건이 붙는다.

1. 라그랑주 승수를 제외한 변수에 대한 편미분 값은 0이 되어야 한다.
2. 라그랑주 승수는 0보다 크거나 같아야 한다.
3. 라그랑주 승수와 제약식 중 하나는 무조건 0이 되어야 한다(즉, 둘의 곱은 항상 0).

> 조건을 전개해서 정리하자

Unit 01 | SVM

(w, b, α) 가 Lagrangian dual problem의 최적해가 되기 위한 조건

KKT (Karush-Kuhn-Tucker) conditions:

① Stationarity

$$\frac{\partial \mathcal{L}(w, b, \alpha)}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^n \alpha_i y_i x_i \quad \frac{\partial \mathcal{L}(w, b, \alpha)}{\partial b} = 0 \Rightarrow \sum_{i=1}^n \alpha_i y_i = 0$$

② Primal feasibility $y_i(w^T x_i + b) \geq 1, i = 1, 2, \dots, n$

③ Dual feasibility $\alpha_i \geq 0, i = 1, 2, \dots, n$

④ Complementary slackness $\alpha_i(y_i(w^T x_i + b) - 1) = 0$

Unit 01 | SVM

Original Problem

$$\text{minimize } \frac{1}{2} \|w\|_2^2$$

$$\text{subject to } y_i(w^T x_i + b) \geq 1, i = 1, 2, \dots, n$$

Lagrangian multiplier를 이용하여 Lagrangian primal문제로 변환

Lagrangian Primal

$$\max_{\alpha} \min_{w, b} \mathcal{L}(w, b, \alpha) = \frac{1}{2} \|w\|_2^2 - \sum_{i=1}^n \alpha_i (y_i (w^T x_i + b) - 1)$$

$$\text{subject to } \alpha_i \geq 0, i = 1, 2, \dots, n$$

Convex, continuous이기 때문에 미분 = 0에서 최소값을 가짐

$$\textcircled{1} \quad \frac{\partial \mathcal{L}(w, b, \alpha)}{\partial w} = 0 \quad \longrightarrow \quad w = \sum_{i=1}^n \alpha_i y_i x_i$$

$$\textcircled{2} \quad \frac{\partial \mathcal{L}(w, b, \alpha)}{\partial b} = 0 \quad \longrightarrow \quad \sum_{i=1}^n \alpha_i y_i = 0$$

$$\textcircled{1} \quad \frac{1}{2} \|w\|_2^2 = \frac{1}{2} w^T w$$

$$= \frac{1}{2} w^T \sum_{j=1}^n \alpha_j y_j x_j$$

$$= \frac{1}{2} \sum_{j=1}^n \alpha_j y_j (w^T x_j)$$

$$= \frac{1}{2} \sum_{j=1}^n \alpha_j y_j \left(\sum_{i=1}^n \alpha_i y_i x_i^T x_j \right)$$

$$= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$\textcircled{2} \quad -\sum_{i=1}^n \alpha_i (y_i (w^T x_i + b) - 1)$$

$$= -\sum_{i=1}^n \alpha_i y_i (w^T x_i + b) + \sum_{i=1}^n \alpha_i$$

$$= -\sum_{i=1}^n \alpha_i y_i w^T x_i - b \sum_{i=1}^n \alpha_i y_i + \sum_{i=1}^n \alpha_i$$

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

Unit 01 | SVM

3-6. 결론- 의미를 중심으로!

위식을 이용해서 정리하면

$\sum \alpha_i y_i = 0$ $\alpha_i \geq 0$ for all α_i 를 만족하고 동시에

$L(\alpha_i) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ 를 최대화하는 α 를 찾으면

하이퍼플레인의 방향은 $\hat{\mathbf{w}} = \sum \alpha_i y_i \mathbf{x}_i$ 이고

분류 결과는 $\text{sgn}(\sum (\hat{\alpha}_i y_i \mathbf{x}_i^T \mathbf{x}_j + b))$ 이 되니까

: w가 최대가 되는 w와 b를 찾는 문제는 $\rightarrow \alpha$ 를 찾는 문제가 된다(컴퓨터가 풀기 쉬움)

Unit 01 | SVM

x_i 가 support vector인 경우에만 $\alpha_i^* \geq 0$ 이므로

$$w^* = \sum_{i=1}^n \alpha_i^* y_i x_i = \sum_{i \in SV} \alpha_i^* y_i x_i$$

즉, support vector만 이용하여 optimal hyperplane (decision boundary) 을 구할 수 있다
(sparse representation!)

또한, 다음과 같이 임의의 support vector 하나를 이용하여 b^* 를 구할 수 있다.

$$w^{*T} + b^* = y_{sv}$$

$$w^{*T} + b^* = \sum_{i=1}^n \alpha_i^* y_i x_i^T x_{sv} + b^* = y_{sv}$$

$$b^* = y_{sv} - \sum_{i=1}^n \alpha_i^* y_i x_i^T x_{sv}$$

Contents

Unit 01 | SVM

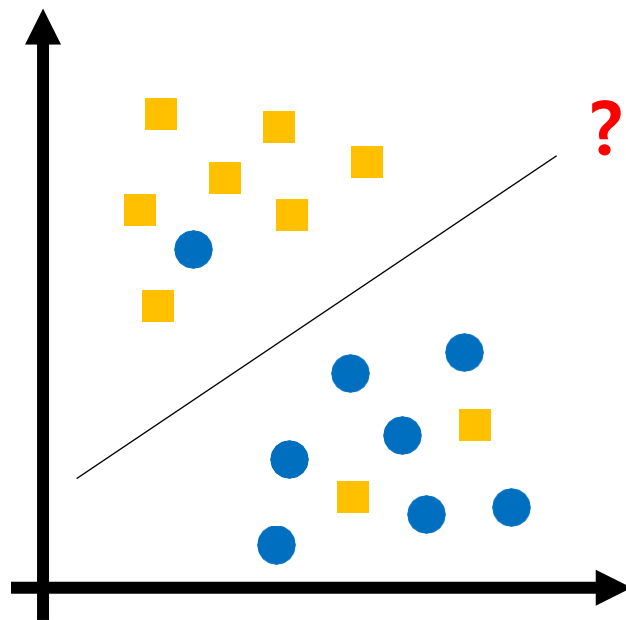
Unit 02 | Soft Margin SVM

Unit 03 | Non-Linear SVM

Unit 04 | Summary

Unit 02 | Soft Margin SVM

현실적으로 모든 데이터가 깔끔한 경계로 나뉘지는 않는다!
이런 애들은 어떻게 해줘야 할까?



Unit 02 | Soft Margin SVM

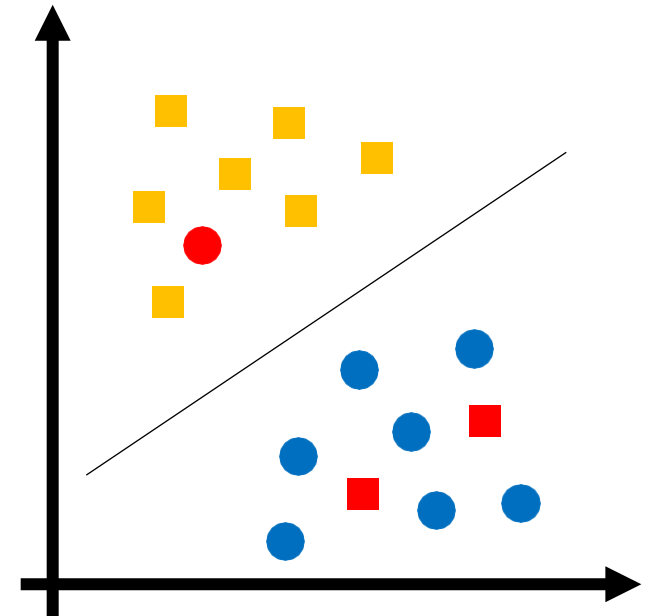
1. Soft Margin SVM

> 우리가 지금까지 했던 건 Hard Margin SVM

:error를 허용하지 않고 분류

> Soft Margin SVM

:error(오분류)를 허용하되, 패널티를
줘서 전체 error를 최소화!

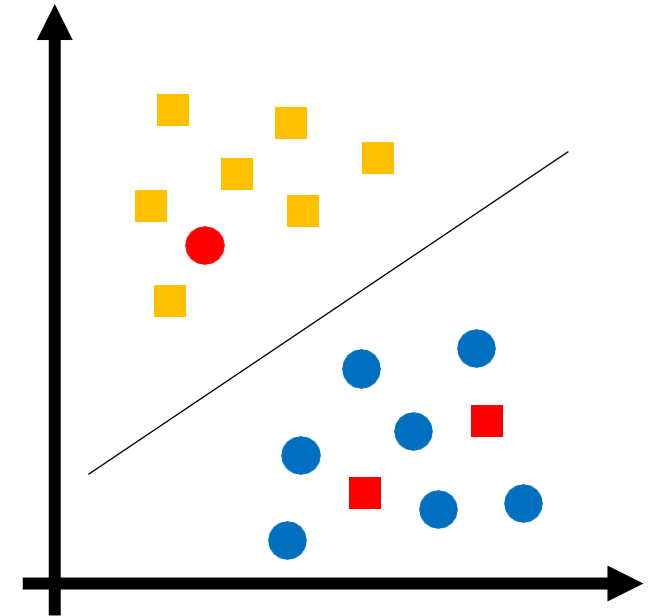


Unit 02 | Soft Margin SVM

2. Penalty

> Penalty를 주는 방법

1. 0-1 Loss
2. Hinge Loss

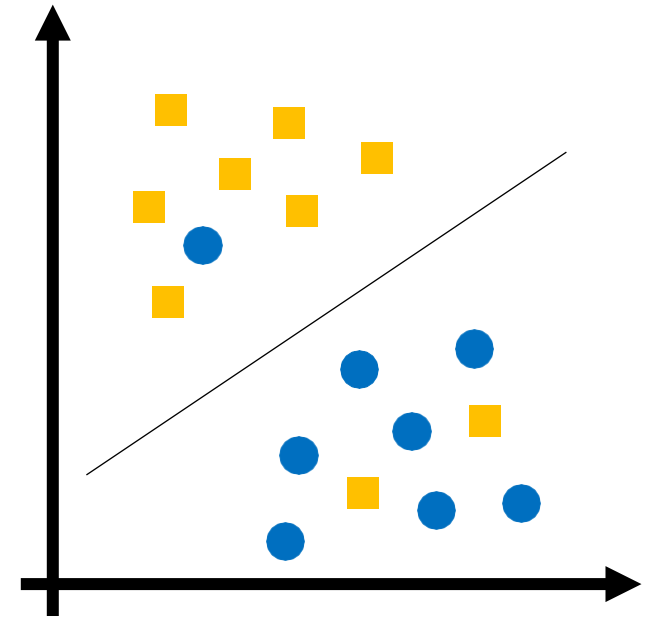


Unit 02 | Soft Margin SVM

2-1.0-1 Loss

:error가 발생한 개수만큼 패널티 계산

$$:\min ||\mathbf{w}|| + C\#\text{error}$$



Unit 02 | Soft Margin SVM

2-2. Hinge Loss

:오분류 정도에 따라 error의 크기를 다르게 하자

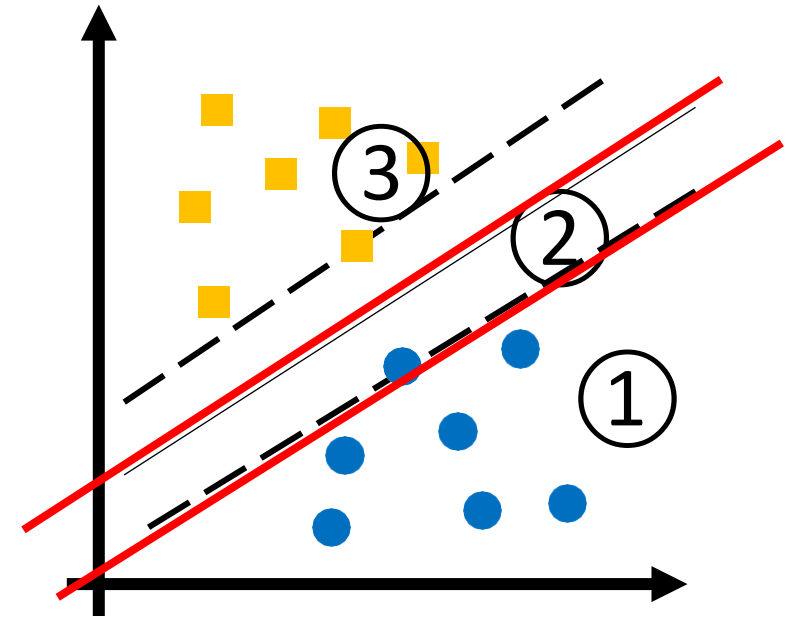
실제로는 ● 인데이터가 분류기의 각 영역에 있을 때

① error 없음 (좋은 분류) $\xi_j = 0$

② 작은 error $0 \leq \xi_j \leq 1$

③ 큰 error (잘못된 분류) $\xi_j > 1$

> slack variable(ξ_j) 사용



Unit 02 | Soft Margin SVM

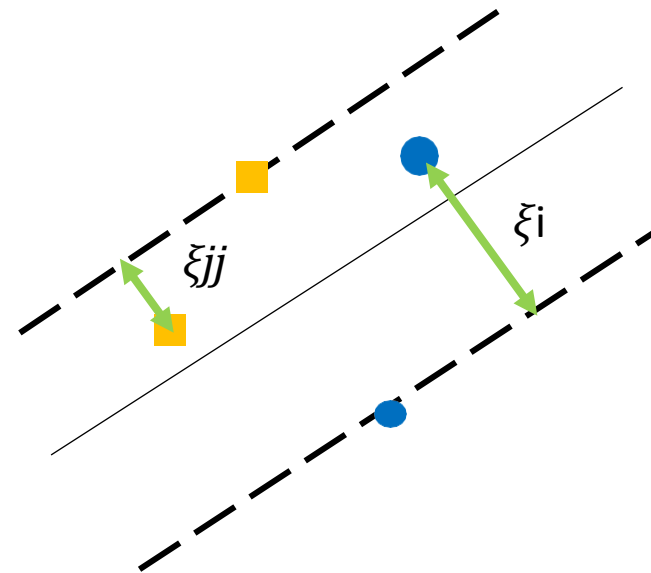
2-2. Hinge Loss

> 기존 목적함수에 error항을 추가

$$:\operatorname{argmin} ||w|| + C \sum \xi_j$$

> 여기서 c 는 하이퍼파라미터(추가자료 참고)

c 가 크다면 > error를 줄이는데에,
 c 가 작다면 > w 를 줄이는데에 신경써주자!



Contents

Unit 01 | SVM

Unit 02 | Soft Margin SVM

Unit 03 | Non-Linear SVM

Unit 04 | Summary

Unit 03 | Non-Linear SVM

1. 구분할 수 있는 linear line이 없는데
2. Outlier를 무시하지 못 하는 경우에는?

- > linear svm 불가능
- > soft margin svm 불가능

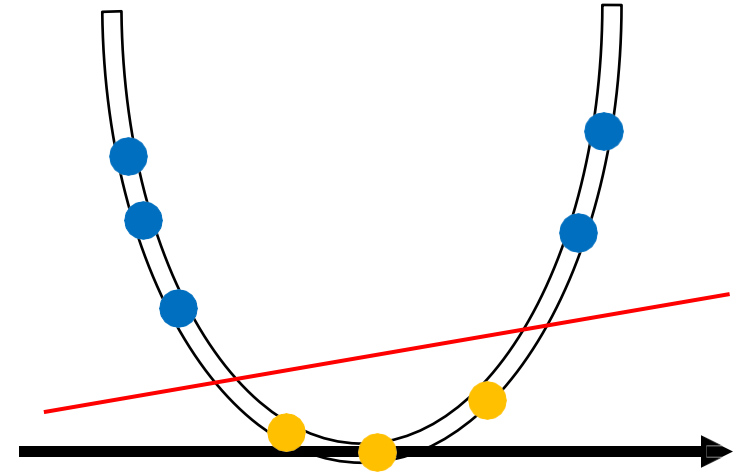


Unit 03 | Non-Linear SVM

1차원에서 선형 경계로 분류 불가능한 data를
> 2차원으로 매핑하면 분류 가능!



MAPPING



Unit 03 | Non-Linear SVM

1. Kernel

:저차원 데이터를 고차원 데이터로 매핑하는 작업

:관측치 x 들을 높은 차원으로 매핑해서 분류하자!

:SVM을 original space가 아닌 고차원의 feature space에서 학습시키면 좋지 않을까?

Original space에서는 nonlinear하게 경계를 그어야 했지만, feature space에서는 linear하게 경계선 그을 수 있다!

<https://www.youtube.com/watch?v=3liCbRZPrZA> 전체
<https://www.youtube.com/watch?v=-Z4aojJ-pdg> 9:45~10:05

Unit 03 | Non-Linear SVM

1. Kernel

목적함수를 최적화하는 수식에 내적이 있다!

$$\max L(\alpha_i) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\Rightarrow \max L(\alpha_i) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

그런데 고차원으로 매핑하니 내적 부분의 차원이 증가한다

> 연산량이 너무 많지 않아?

$$\phi : x \mapsto z = \phi(x)$$

Example

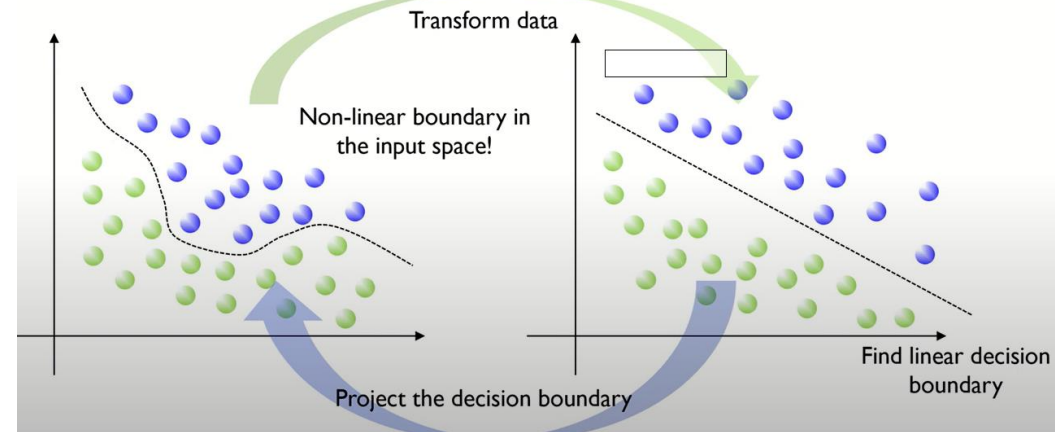
$$\phi : (x_1, x_2) \mapsto (x_1, x_2, x_1^2, x_2^2, x_1 x_2)$$

2D (Original Space) 5D (Feature Space)

$$x = (x_1, x_2, \dots, x_n) \rightarrow \phi(x) = z = (z_1, z_2, \dots, z_n)$$

Input Space R^p Feature Space R^q

$p \ll q$



Unit 03 | Non-Linear SVM

2. Kernel Trick

- > 고차원으로 매핑한 '데이터' 말고 '내적값'을 알자!
:고차원으로 매핑하되, 연산은 간단하게 할 수 있게 된다

$$:\max L(\alpha_i) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \quad \Rightarrow \quad \max L(\alpha_i) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

- > 매핑한 함수/데이터는 필요없다. 매핑한 내적값(최종결과)만 알면 된다!
- > 그 값을 쉽게 알 수 있는 'Trick'

Unit 03 | Non-Linear SVM

2. Kernel Trick

:고차원으로 보낸 뒤 벡터의 내적 연산 = 내적을 한 후 고차원으로 보내는 연산

Ex)

$$X = (x_1, x_2)$$

$$Y = (y_1, y_2)$$

$$\phi(X) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\phi(Y) = (y_1^2, y_2^2, \sqrt{2}y_1y_2)$$

$$\langle \phi(X), \phi(Y) \rangle = \underbrace{x_1^2 y_1^2} + \underbrace{x_2^2 y_2^2} + \underbrace{2x_1 x_2 y_1 y_2}$$

Question: Can we compute $x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 x_2 y_1 y_2$ without knowing the explicit functional form of $\phi(X)$ and $\phi(Y)$?

$$\begin{aligned} (X, Y)^2 &= \langle (x_1, x_2), (y_1, y_2) \rangle^2 \\ &= \langle x_1 y_1 + x_2 y_2 \rangle^2 \\ &= x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 x_2 y_1 y_2 \\ &= \langle \phi(X), \phi(Y) \rangle \end{aligned}$$

Unit 03 | Non-Linear SVM

2. Kernel Trick

> kernel을 쓰는 이유?

: 고차원으로 매핑도 하고, 연산도 간단하게 할 수 있으니까!

> Kernel Trick을 만족하는 조건도 있지만, 복잡하니 생략하고 자주 쓰는 kernel만 알자

Unit 03 | Non-Linear SVM

3. 자주 쓰는 Kernel

: 가장 많이 사용하는 kernel은

가우시안 커널! (RBF kernel)

: 각 함수마다 파라미터가 조금씩 다름!

- Linear kernel

$$K\langle x_1, x_2 \rangle = \langle x_1, x_2 \rangle$$

- Polynomial kernel

$$K\langle x_1, x_2 \rangle = (a\langle x_1, x_2 \rangle + b)^d$$

- Sigmoid kernel (Hyperbolic tangent kernel)

$$K\langle x_1, x_2 \rangle = \tanh(a\langle x_1, x_2 \rangle + b)$$

- Gaussian kernel (Radial basis function (RBF) kernel)

$$K\langle x_1, x_2 \rangle = \exp\left(\frac{-\|x_1 - x_2\|_2^2}{2\sigma^2}\right)$$



$$e^{-\gamma(a-b)^2}$$

Unit 03 | Non-Linear SVM

4. RBFkernel (=Gaussian Kernel)

:무한대의 차원으로 매핑하는 커널 (by 테일러급수)

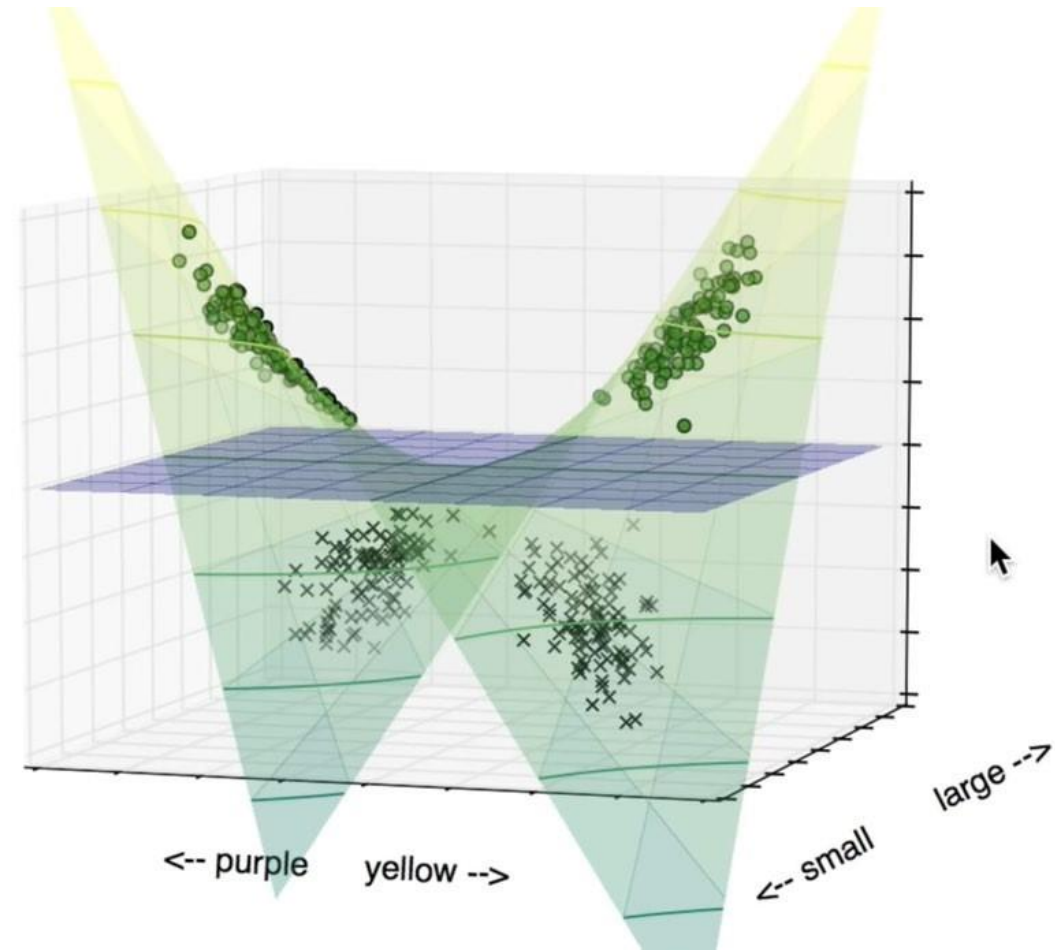
Ex) $2\sigma^2 = 1 \Rightarrow$

$$\begin{aligned} K(x_1, x_2) &= \exp \left\{ -(x_1 - x_2)^2 \right\} \\ &= \exp(-x_1) \exp(-x_2) \exp(2x_1 x_2) \end{aligned}$$

여기서, 테일러급수에 의해

$$\exp(2x_1 x_2) = \sum_{k=0}^{\infty} \frac{2^k x_1^k x_2^k}{k!}$$

> 무한대 차원의 Feature Space로 매핑!



Unit 03 | Non-Linear SVM

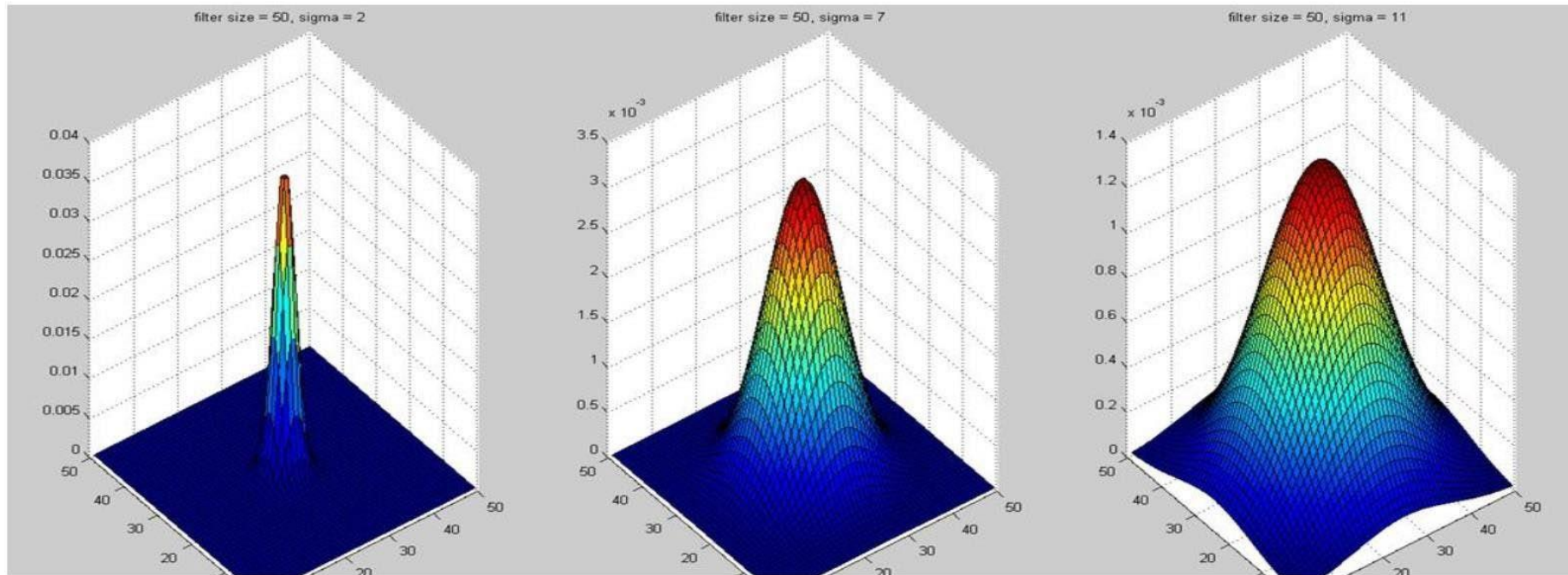
4-1. RBF 커널 - Gamma Parameter Tuning

$$K(x_1, x_2) = \exp \left\{ -\frac{\|x_1 - x_2\|_2^2}{2\sigma^2} \right\}, \quad \sigma \neq 0 \quad \longrightarrow \quad e^{-\gamma(a-b)^2}$$

$$\Rightarrow \gamma = \frac{1}{2\sigma^2} \quad \Rightarrow \quad \text{감마와 분산(표준편차)은 반비례 관계}$$

Unit 03 | Non-Linear SVM

4-1. RBF커널 - GammaParameter Tuning



Gamma 감소 = 표준편차 증가

Unit 03 | Non-Linear SVM

4-1. RBF커널 - GammaParameter Tuning

:감마가 클수록

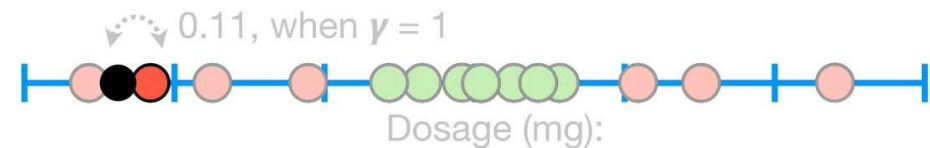
- 훨씬 인접한 것들만 같은 영역으로 본다
- =엄청 인접하지 않으면 엄청 먼 곳으로 인식한다
- =원래 차원으로 돌아왔을 때 경계가 아주 촘촘하다
- =Hyper plane이 훨씬 더 굴곡지다
- =Overfitting의 가능성이 높다

...we get **0.11** when $\gamma = 1$.

$$e^{-(2.5-4)^2} = e^{-(-1.5)^2} = e^{-2.25} = 0.11$$

When $\gamma = 2$, we get **0.01**...

$$e^{-2(2.5-4)^2} = e^{-2(-1.5)^2} = e^{-2 \times 2.25} = 0.01$$



Contents

Unit 01 | SVM

Unit 02 | Soft Margin SVM

Unit 03 | Non-Linear SVM

Unit 04 | Summary

Unit 04 | Summary

1. Hard Margin과 Soft Margin

:Hard Margin :에러를 허용하지 않음 (현실적이지 못함)

:Soft Margin :에러를 허용

:에러를 허용하는 방법은 0-1 Loss와 Hinge Loss

:에러와 마진 둘 중 무엇을 줄일 것인지를 결정하는 hyper parameter는 C

Unit 04 | Summary

2. Linear SVM과 Non-linear SVM

> Linear SVM

:선형으로 분류

:Feature가 많아 kernel을 하게 되면 차원이 높아지고 연산량이 많아지니 Linear가 효과적.

> Non-Linear SVM

:Feature가 많지 않을 땐 그걸로 분류하기 쉽지 않기 때문에 차원을 올려줌

:이때 차원을 올려주는 방법에 따라 다양한 parameter가 존재하지만

가장 많이 쓰이는 건 Gaussian Kernel에서 Gamma

Unit 04 | Summary

3. 정리

- 1) SVM 알고리즘 중 일반적으로 널리 사용되는 건 RBF 커널 SVM
- 2) 좋은 성능을 얻으려면 매개변수인 C 와 γ 를 잘 조정해줘야 함
- 3) C 는 데이터 샘플들이 다른 클래스에 놓이는 것을 허용하는 정도를, γ 는 결정경계의 곡률을 결정
- 4) 두 값 모두 커질수록 알고리즘의 복잡도는 증가
- 5) 일반적으로 grid search로 경험적으로 최적의 매개변수 값들을 찾아가는데, 당연하지만 내용을 어느정도 숙지하게 된다면 훨씬 더 빠르게 좋은 성능을 내는 매개변수 값들을 찾아낼 수 있을 것!

Unit 04 | Summary

Multi Class SVM

1. One VS one

클래스가 N 개 있을 때 모든 Class에 대해 1:1로 binary분류를 하고 제일 많이 승리한 것에 대해 투표로 결정한다.

N 개의 클래스에 대해 서로서로 Classifier를 가지고 있어야 하기 때문에 $n(n-1)/2$ 개의 Classifier가 필요하다.

참고 4. One vs One(OVO)

만약 머신 세 개 각각이 1번 머신은 A가, 2번 머신은 B가, 3번 머신은 C가 맞다고 했다면 어떻게 판별은?

One vs One 예시)

데이터 클래스가 A, B, C라 하고, 우리는 (또) 총 3개의 머신을 만든다.

1. A인지 B인지를 구분해주는 머신
2. B인지 C인지를 구분해주는 머신
3. C인지 A인지를 구분해주는 머신

새로운 데이터가 들어오고, 우리는 이 데이터를 1,2,3번 머신에 돌린 결과, 1번 머신은 A, 2번 머신은 C, 3번 머신은 A라고 했다. 그렇다면? 이 세 결과를 통해 새로 들어온 이 데이터가 A라고 판별!

Unit 04 | Summary

2. One vs rest

클래스가 N 개 있으면 모든 class에 대해 1 : $N-1$ 로 binary 분류하여 이 클래스가 맞는지 아 닌지를 판단하고 투표로 결정하고, 이 때 N 개의 Classifier가 필요하다.

Onevs Rest 예시)

데이터의 클래스로 A, B, C가 있다고 하자. 우리는 총 3개의 머신을 만든다.

1. A인지 아닌지 구분해주는 머신
2. B인지 아닌지 구분해주는 머신
3. C인지 아닌지 구분해주는 머신

새로운 데이터가 들어왔고, 우리는 이 데이터를 1, 2, 3번 머신에 돌렸다.
1번 머신은 A라고, 2번 머신은 B가 아니라고, 3번 머신은 C가 아니라고 했다.
이 결과를 통해 새로 들어온 데이터가 A라고 판별할 수 있다.

Unit 04 | 실습

▼ 서포트 벡터 머신

```
▶ import pandas as pd
import numpy as np
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score

customer = pd.read_csv("data/customerChurn.csv")
```

```
[ ] customer.head()
```

	sex	age	Recency	Frequency	Monetary	Churn
0	M	11	4	41	1268981	No
1	M	10	18	15	843996	No
2	F	38	16	41	1755623	No
3	M	13	8	14	862172	No
4	F	20	22	42	748430	No


```
[ ] X = customer.drop(['Churn', 'sex'], axis=1)
y = customer['Churn']
```

▼ 변수 표준화

```
[ ] from sklearn.preprocessing import StandardScaler



scaler = StandardScaler()
X_s = pd.DataFrame(scaler.fit_transform(X))
```

Unit 04 | 실습

 `X_s.head()`

	0	1	2	3
0	-1.555877	-1.583993	1.631371	1.130492
1	-1.667845	-0.057211	-0.330064	0.374340
2	1.467244	-0.275322	1.631371	1.996346
3	-1.331942	-1.147769	-0.405504	0.406679
4	-0.548170	0.379013	1.706811	0.204305

```
[ ] X_s.columns = ['age', 'Recency', 'Frequency', 'Monetary']
```

 `X_s['sex'] = customer['sex']` `X_s.head()`

	age	Recency	Frequency	Monetary	sex
0	-1.555877	-1.583993	1.631371	1.130492	M
1	-1.667845	-0.057211	-0.330064	0.374340	M
2	1.467244	-0.275322	1.631371	1.996346	F
3	-1.331942	-1.147769	-0.405504	0.406679	M
4	-0.548170	0.379013	1.706811	0.204305	F

Unit 04 | 실습

▼ one hot encoding

```
[ ] X_ohc = pd.get_dummies(X_s)
    X_ohc.head()
```

	age	Recency	Frequency	Monetary	sex_F	sex_M
0	-1.555877	-1.583993	1.631371	1.130492	0	1
1	-1.667845	-0.057211	-0.330064	0.374340	0	1
2	1.467244	-0.275322	1.631371	1.996346	1	0
3	-1.331942	-1.147769	-0.405504	0.406679	0	1
4	-0.548170	0.379013	1.706811	0.204305	1	0

```
rbf_svm_clf = SVC(kernel="rbf", gamma=5, C=0.001)
```

▼ 서포트 벡터 머신 모형 학습

```
[ ] X_train, X_test, y_train, y_test = train_test_split(X_ohc, y)
    linear_svm = LinearSVC(C=1)
    linear_svm.fit(X_train, y_train)
```

```
LinearSVC(C=1, class_weight=None, dual=True, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
```

```
[ ] print("Predictions: {}".format(linear_svm.predict(X_test)))
    print("Accuracy: {}".format(linear_svm.score(X_test, y_test)))
```

```
Predictions: ['No' 'Yes' 'No' ... 'Yes' 'Yes' 'Yes']
Accuracy: 0.9456
```

```
[ ] from sklearn.model_selection import cross_val_score
```

```
linear_svm = LinearSVC(C=1)
scores = cross_val_score(linear_svm, X_ohc, y, scoring='roc_auc', cv=5)
scores.mean()
```

```
0.9842504
```

Unit 04 | 실습

```
from sklearn.model_selection import GridSearchCV

param_grid = [{'C': [11, 12, 13, 14, 15, 16, 17, 18, 19]}]

linear_svm = LinearSVC(max_iter=10000)

grid_search = GridSearchCV(
    linear_svm,
    param_grid,
    cv=5,
    scoring='roc_auc',
    return_train_score=True)

grid_search.fit(X_ohe, y)
```

```
[ ] grid_search.best_params_
```

```
{'C': 12}
```

```
linear_svm = LinearSVC(C=grid_search.best_params_['C'], max_iter=10000)

scores = cross_val_score(linear_svm, X_ohe, y, scoring='roc_auc', cv=5)
scores.mean()
```

```
0.9842511999999999
```

참고자료

참고 1. Grid Search의 활용

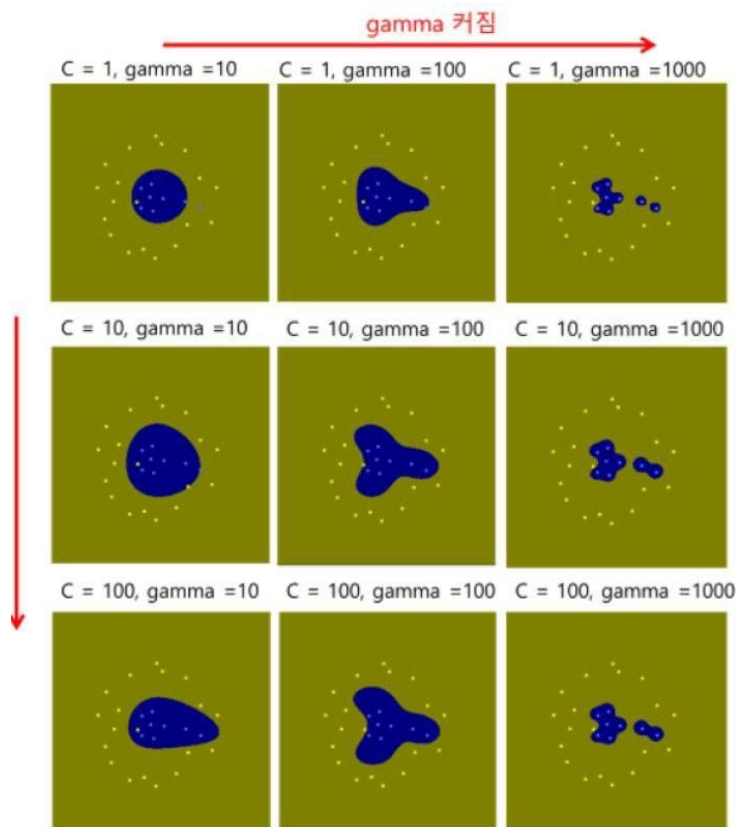


그림9. C와 gamma의 영향

하이퍼파라미터가 두 개나 있어서 감이 안 올 때는?

C: 데이터가 다른 클래스에 놓이는 걸 허용하는 정도

γ : 결정경계의 곡률을 결정 (kernel 모델의 파라미터)

> Overfitting 위험: C가 크고 Gamma가 클 때

> underfitting 위험: C가 작고 Gamma가 작을 때

참고자료

참고 2 사이킷런 SVM parameter

1. Kernel

Decision Boundary의 모양 결정

Kernel 선택 가능 (Linear, Polynomial, Sigmoid, RBF 등)

2. C

: Decision Boundary 일반화 VS training data의 정확한 분류 사이의 trade-off 조정

: C가 크면 정확하게 구분하는 데에, 작으면 smooth한 결정경계를 만드는 데에 초점을 맞춤

: C있음: soft / C없음: hard margin svm

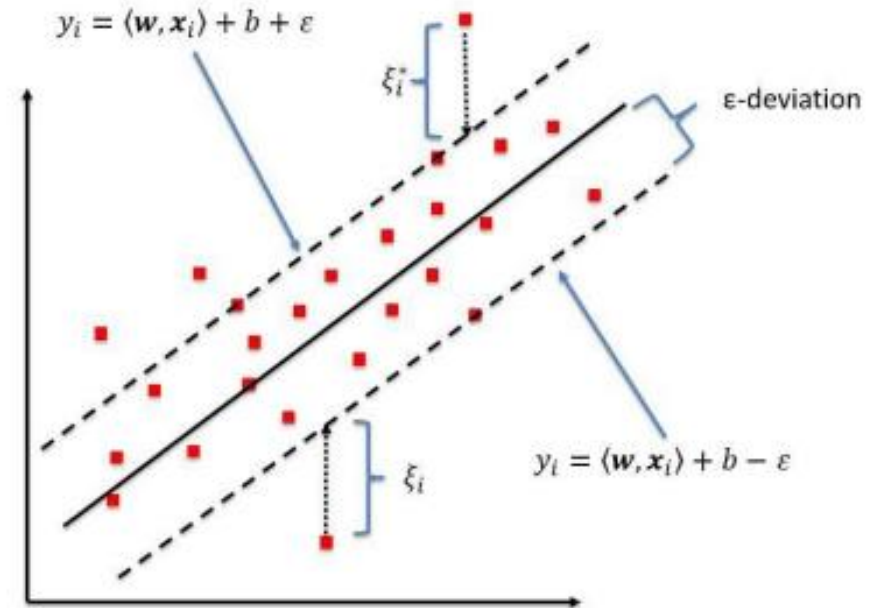
3. Gamma

: 결정경계의 굴곡에 영향을 주는 데이터의 범위(reach)를 정의

참고자료(SVR)

참고자료(SVR)

- 회귀문제의 경우는 분류문제와 반대로 제한된 마진 오류 안에서 마진 안에 가능한 많은 데이터가 들어가도록 학습함
- 마진의 넓이는 하이퍼 파라미터 ϵ (epsilon)으로 조절하며, 마진 오류는 ζ (zeta)임(scikit learn에서는 각각 epsilon과 tol로 지정함)



참고자료(SVR)

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

```
boston = pd.read_csv("data/boston.csv")
```

```
X = boston.iloc[:,1:-1]
y = boston["medv"]
X.head()
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
3	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33
4	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.43

```
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

```
from sklearn.svm import LinearSVR
```

```
svm_reg = LinearSVR(epsilon = 5, tol=0.1, C=1, max_iter=100000)
svm_reg.fit(X_train, y_train)
```

```
y_pred = svm_reg.predict(X_test)
```

```
from sklearn.metrics import mean_squared_error
np.sqrt(mean_squared_error(y_test,y_pred))
```

```
5.39020124561961
```

```
[ ] from sklearn.model_selection import cross_val_score
scores = cross_val_score(svm_reg, X, y, scoring='neg_mean_squared_error',cv=5)
np.sqrt(-scores.mean())
```

```
6.657625045369984
```

과제 설명

Assignment 1. Multiclass SVM 구현

1. **Multiclass SVM을 직접 구현하시는 것입니다.** 기본적으로 사이킷런에 있는 SVM은 **멀티클래스 SVM을 지원하지만 과제에서는 절대 쓰면 안됩니다!** Iris 데이터는 총 세 개의 클래스가 있으므로 이 클래스를 one-hot인코딩한 뒤, 각각 binary SVM을 트레이닝하고 이 결과를 조합하여 multiclass SVM을 구현하시면 됩니다
 2. 기본적으로 one vs one, one vs rest 방법이 있으며 둘 중 자유롭게 구현해주세요. 만약 투표결과가 동점으로 나온 경우(예를 들어, 각각의 SVM 결과가 A vs B 의 경우 A로 판별, B vs A 의 경우 B로 판별, C vs A 의 경우 C로 판별한 경우 투표를 통해 Class를 결정할 수 없음)
 - 1) decision_function을 활용하시거나
 - 2) 가장 개수가 많은 클래스를 사용하시거나
 - 3) 랜덤으로 하나를 뽑거나 하는 방법 등을 이용해 동점자인 경우를 판별 해주시면 됩니다.
- 공식문서를 통해 사이킷런이 어떤 방법으로 구현했는지 참고하셔도 됩니다
3. 과제코드에는 iris 데이터를 로드하고 스케일링 부분까지 구현되어 있습니다.
 4. Iris의 클래스는 3개입니다. Iris 데이터셋 뿐만 아니라 다른 데이터셋에도 적용 가능한, 클래스의 수와 무관한 Multiclass SVM을 만들어주세요.

- 투빅스 13기 이유민님 강의자료
- 투빅스 16기 김종우님 강의자료
- 투빅스 17기 유현우님 강의자료
- 고려대학교 김성범 교수님 강의자료

<https://www.youtube.com/watch?v=qFg8cDnqYCI>

-Patrick Winston, MIT OCW6.034 Fall 2010, Lec.16 Learning: Support Vector Machines

-앤드류응(Andrew Ng)교수님의Machine Learning강의

https://www.youtube.com/watch?v=Qc5IyLW_hns

<https://sooho-kim.tistory.com/85> <https://junklee.tistory.com/108>

<https://www.youtube.com/watch?v=qe3A9B76cRI&t=529s>

감사합니다