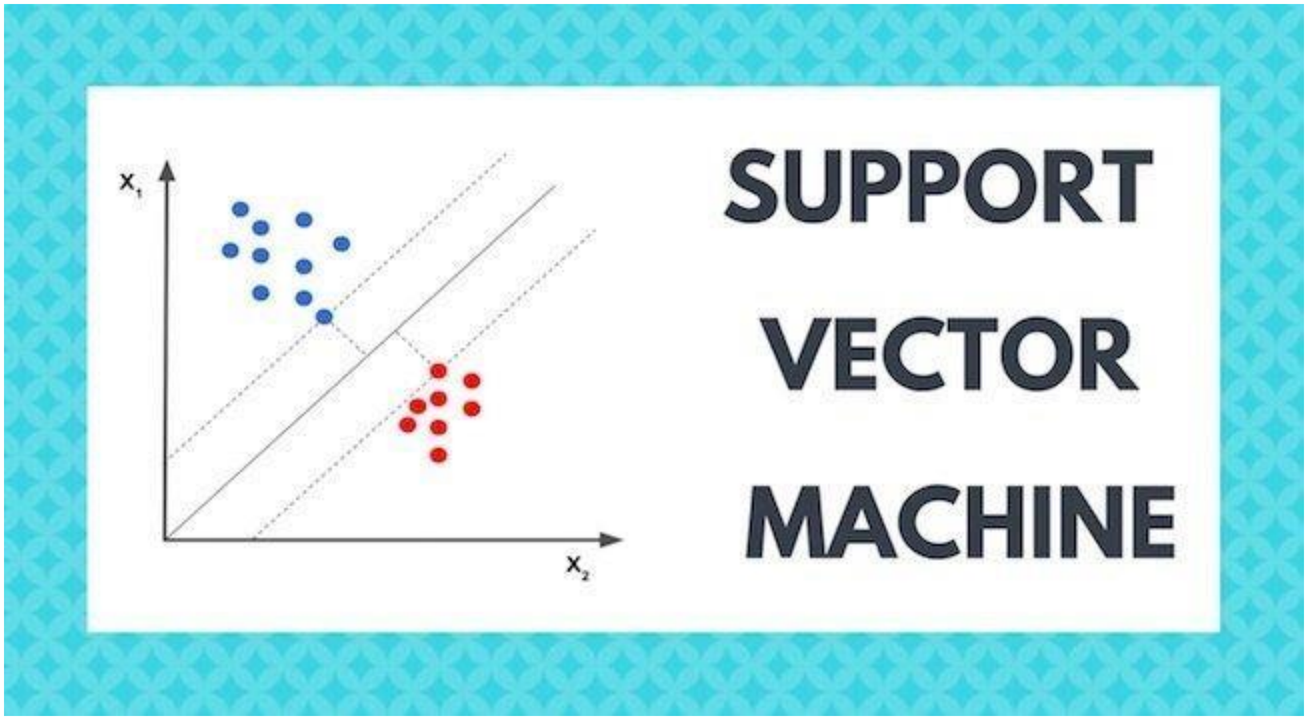
SVM (Support Vector Machine)



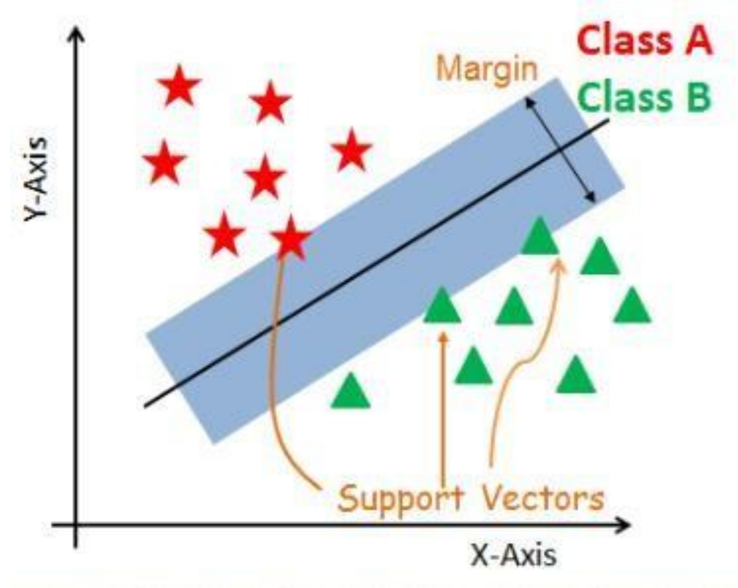
2019.10.29.이 승 한

< 목 차 >

1. SVM이란?

Support Vector Machine

- Supervised learning
- both 1) Classification & 2) Regression



- 범주형변수: Support Vector Classifier
- 연속형변수: Support Vector Regressor

Data들을 가장 잘 구분할 수 있는 경계는?

1. SVM이란?

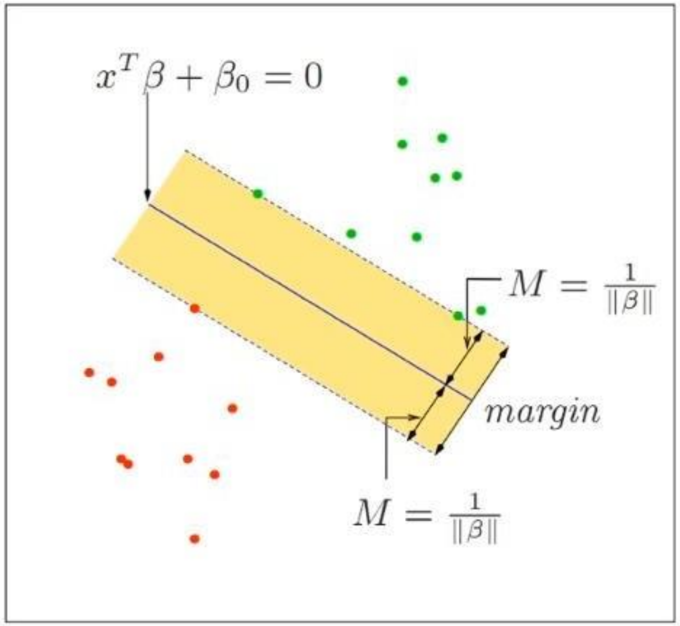
데이터의분포가정이힘들 때, 데이터를가장 잘 나누려면?

* boundary에 집중

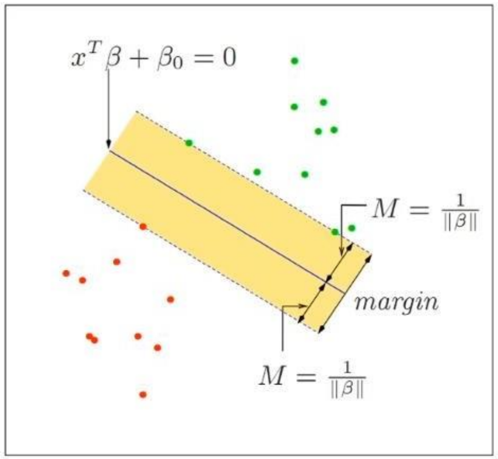
* 그림과같이margin을 최대화하는boundary 찾기

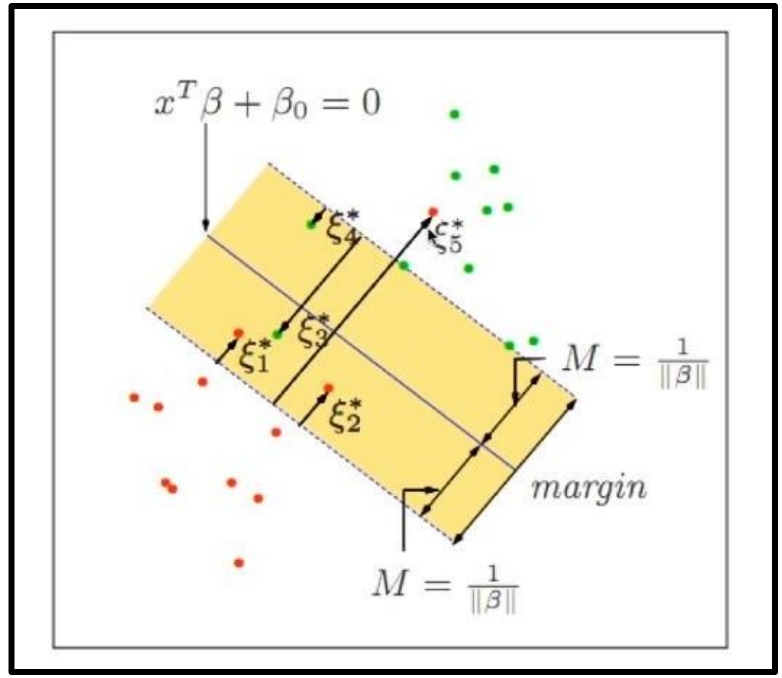
- Hyperplane : data를 구분하는경계!

*2차원은 line, 3차원 이상은hyperplane



2. SVM의 원리





정확히 구분되지 않는 경우?

적당한 error을 허용, 이를 최소화하도록 boundary 결정!

<u>Cost 산정기준</u>(SVM, SVR의 경우 약간 다름!)

- SVM : Margin안에 포함되거나, 반대방향으로 분류된 점들
- SVR : Margin 바깥에 위치한 점들

2. SVM의 원리

<u>Decision Boundary</u>

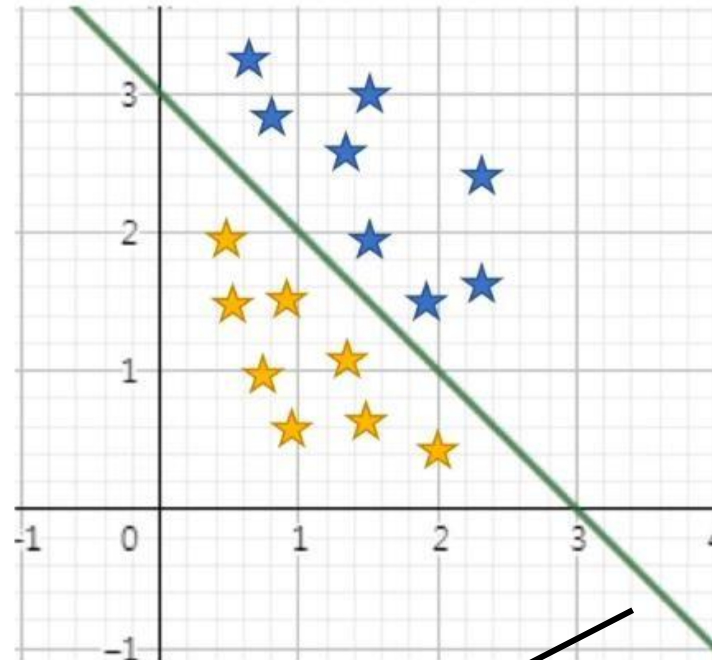
(1) Decision Boundary

$$h_a(x) = a_0 + a^T x = a_0 + a_1 x_1 + \ldots + a_p x_p$$

(2) Decision Rule

푸른 점: $h_a(x)>0$

노란 점: $h_a(x)<0$



a^T, a_0 를 조정하면, 예측값이 바뀌게 됨!

(결국, a^T, a_0 를 설정하는과정이 learning 과정!)

2. SVM의 원 리

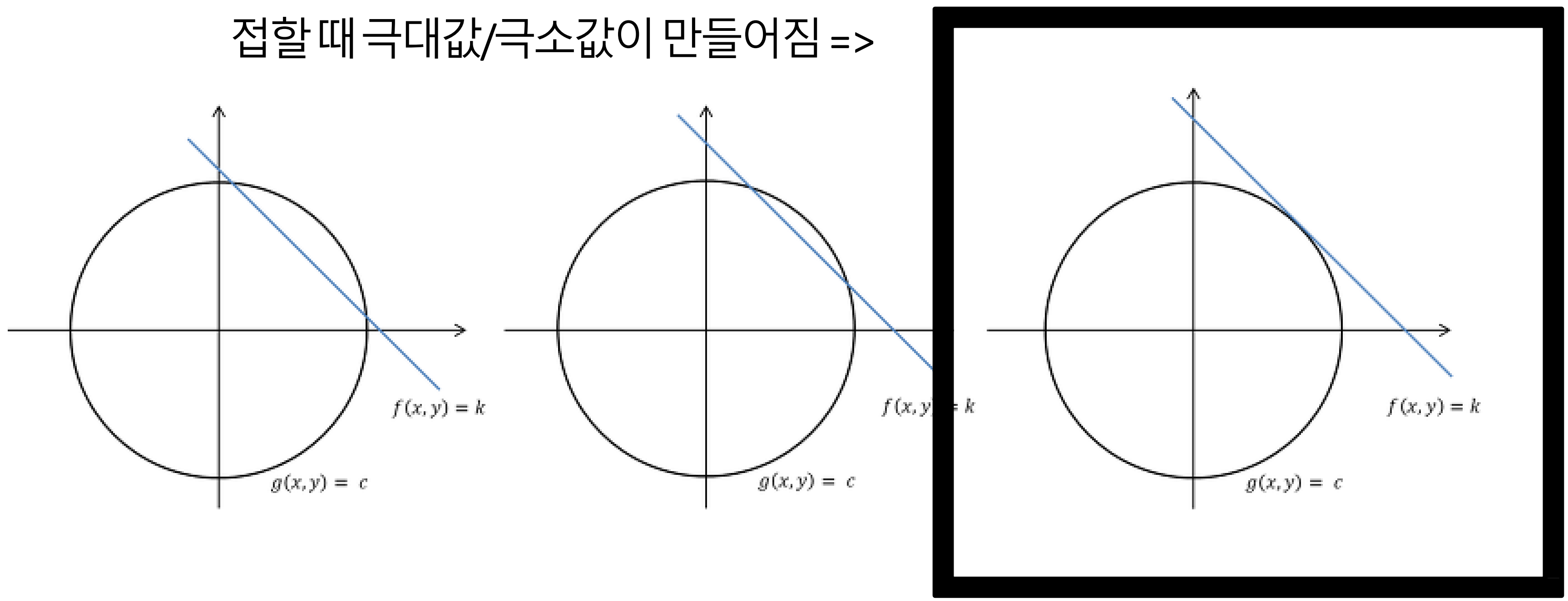
how to Optimize?

들어가기 전에...

<u>Lagrange Multiplier (라그랑즈 승수)</u>

f(x,y)를 최대화하는동시에, g(x,y)=c로 한정하고 싶은경우!

접할 때 극대값/극소값이 만들어짐 =>



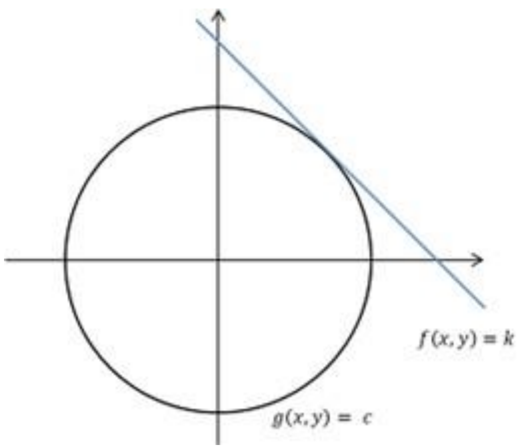
2. SVM의 원 리

Lagrange Multiplier의 기본아이디어

- f(x,y)와 g(x,y)=c가 접할 때 f(x,y)의 극대값or 극소값이발생
- f와 g의 변화량이상수배가되는시점

$$\Delta f = \lambda \Delta g$$

- $\Delta f = \left(\frac{\delta f}{\delta x}, \frac{\delta f}{\delta y}\right), \Delta g = \left(\frac{\delta g}{\delta x}, \frac{\delta g}{\delta y}\right)$



$$\Downarrow$$

$$\boxed{L(x, y, \lambda) = f(x, y) - \lambda(g(x, y) - c)}$$

즉, g(x,y)=c의 제약하에서 f(x,y)를 최대/최소화 하는 것은

위의 L값을 최대/최소화 하는 것과 같다!

2. SVM의 원리

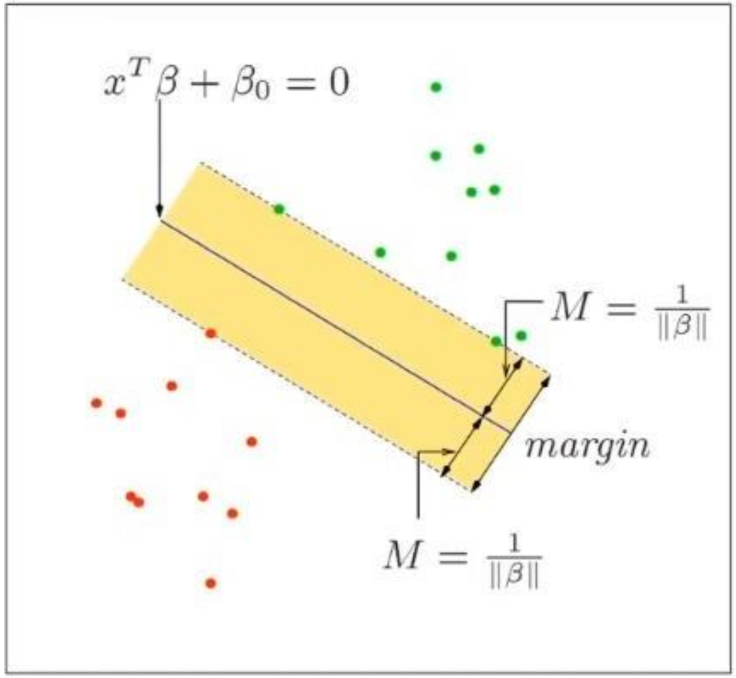
Back to SVM…

- Decision Boundary

  (아래와같은f(x) 평면을정의함)

  $$\boxed{f(x) = x^T\beta + \beta_0 = 0}$$

  유일한베타값을위해 $||\beta|| = 1$



- Decision Rule

- If $(x_i^T\beta + \beta_0) > 0$, then y = 1 otherwise, y = -1.
- $y_i(x_i^T\beta + \beta_0) > 0$
  - y = 1, $(x_i^T\beta + \beta_0) > 0$
  - y = -1, $(x_i^T\beta + \beta_0) < 0$

2. SVM의 원리

아래와같이 "Margin을최대로만드는계수값"을구하는문제

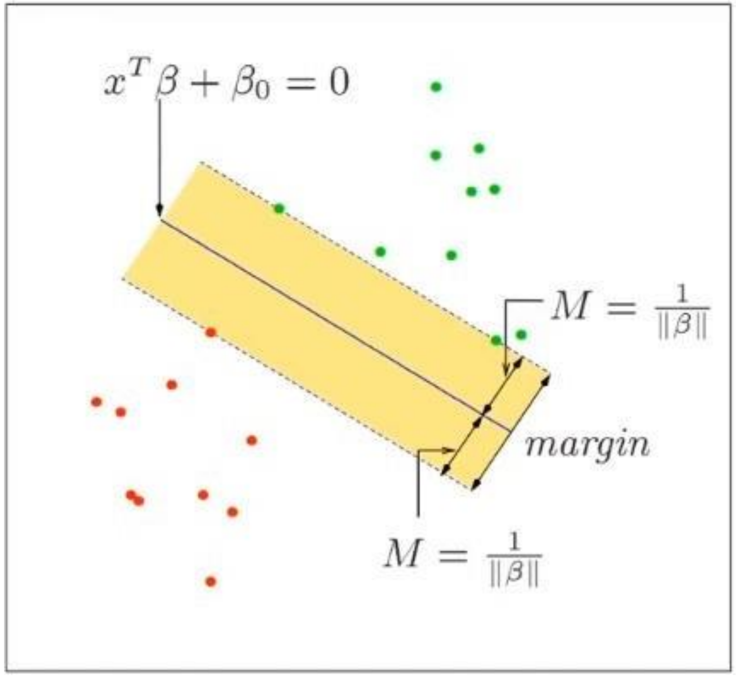
(Error 허용 X)

$$\boxed{\max_{\beta,\beta_0,||\beta||=1} M}$$

where $y_i(x_i^T\beta + \beta_0) \geq M$

$$\boxed{\min_{\beta,\beta_0} ||\beta||}$$

where $y_i(x_i^T\beta + \beta_0) \geq 1, M=1/||\beta||$



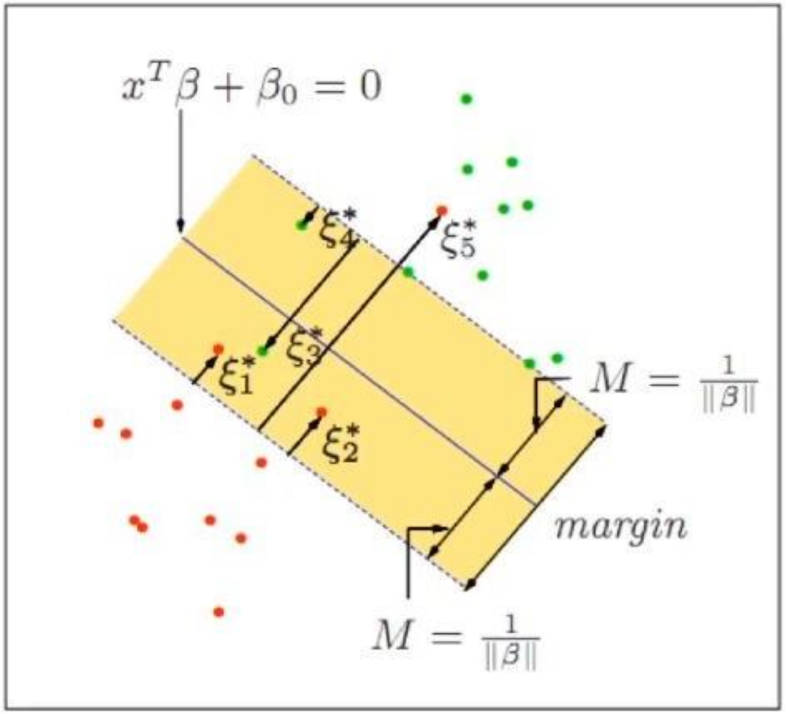
2. SVM의 원리

(Error 허용 O)

$$\boxed{\max_{\beta,\beta_0,||\beta||=1} M}$$

where $y_i(x_i^T\beta + \beta_0) \geq M(1-\xi_i)$

- $\xi_i \geq 0$
- $\sum_i \xi_i \leq \text{constant}$



⬇

$$\boxed{\min_{\beta,\beta_0} ||\beta||}$$

where $y_i(x_i^T\beta + \beta_0) \geq 1-\xi_i, M=1/||\beta||$

- $\xi_i \geq 0$
- $\sum_i \xi_i \leq \text{constant}$

(constant가 클수록, 많은 error을 허용하는 것!)

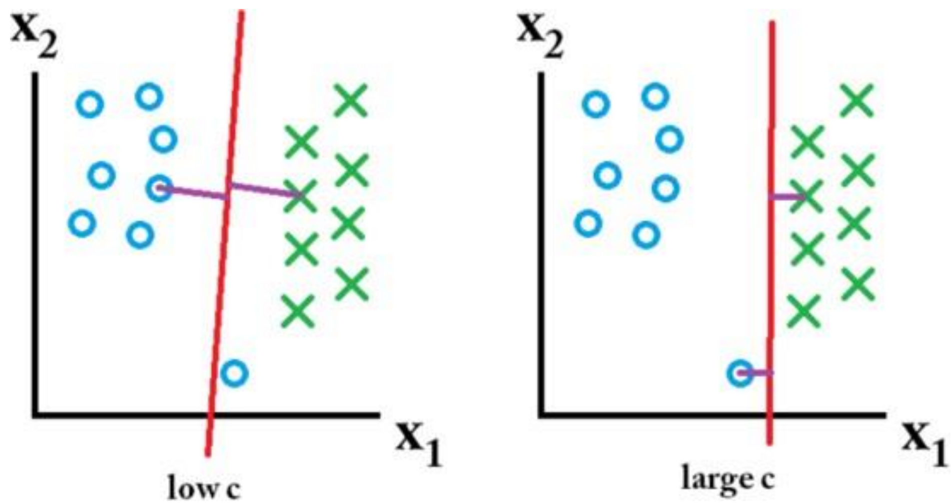
2. SVM의 원 리

Cost Function

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{subject to} \quad \xi_i \geq 0,\ y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i$$

- C가 클수록, error 적게 허용
- C가 무한대= 오분류가하나도 없는 SVM



2. SVM의 원 리

Cost Function

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$L(x,y,\lambda) = f(x,y) - \lambda(\underline{g(x,y) - c})$$

(by Lagrange Multiplier)
위 식을 최소화시키는 것은, 아래 식을 최소화 시키는 것과 같음!

$$L_P = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i \underline{- \sum_{i=1}^{N}\alpha_i[y_i(x_i^T\beta + \beta_0) - (1-\xi_i)] - \sum_{i=1}^{N}\mu_i\xi_i}$$

위 식을 최소화 시키는 β, β_0, ξ_i 찾기!

-> 이를 위해 , Lagrange Multiplier α_i, μ_i 도입!

2. SVM의 원리

Cost Function

$$L_P = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i[y_i(x_i^T\beta + \beta_0) - (1-\xi_i)] - \sum_{i=1}^{N}\mu_i\xi_i$$

β, β_0, ξ_i 에 대해 미분하면...

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

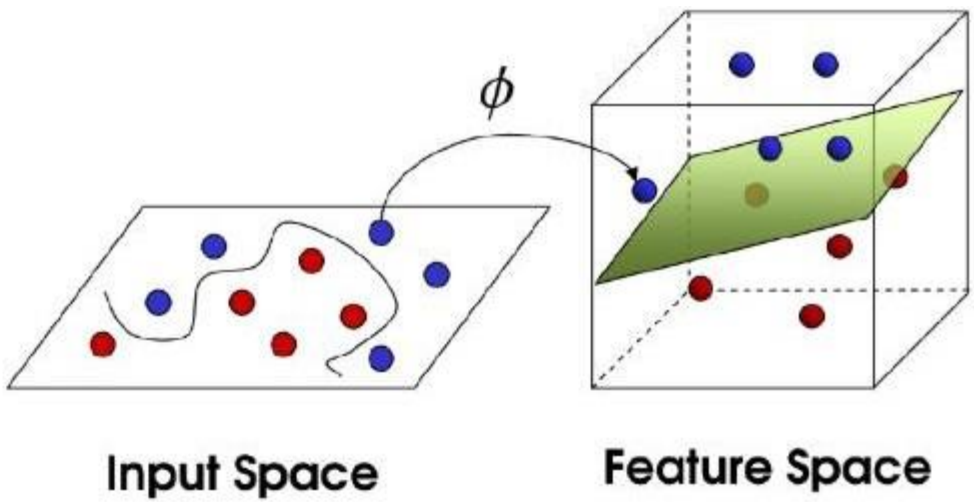
이를 대입하고 정리하면...

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

$$\boxed{\hat{\beta} = \sum_{i=1}^{N}\hat{\alpha}_i y_i x_i}$$

3. Kernel SVM

선형관계가 아닌 경우에 사용!



https://www.youtube.com/watch?v=3liCbRZPrZA

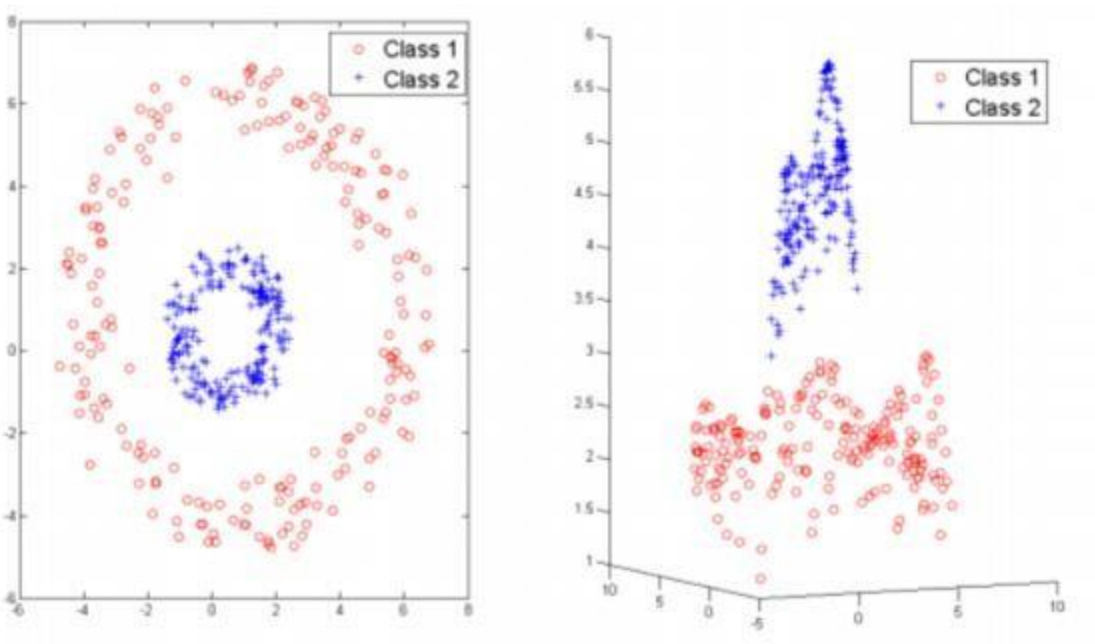
원공간(Input Space)의 데이터를, 선형 분류가 가능한 고차원 공간(Feature Spac

e)으로 mapping한 뒤 두 범주를 분류하는 초평면 찾기!

3. Kernel SVM

Kernel trick

- Feature을 더한 효과를 주지만, computation을 크게 늘리지는 않음!



실제 2차원의 data도, 실제로 feature을 추가하지 않고서도 다음과 같이 feature을 추 가한 것처럼 효과를 줄 수 있다!

3. Kernel SVM

수식적 이해

$$\hat{f}(x) = x\hat{\beta} + \hat{\beta}_0 = x\left(\sum_{i=1}^{N} \hat{\alpha}_i y_i x_i\right) + \hat{\beta}_0$$

$$\Rightarrow \quad \hat{f}(x) = \sum_{i=1}^{N} \hat{\alpha}_i y_i \underline{K(x, x_i)} + \hat{\beta}_0$$

(기 존 의 선 형 관 계 를 깨고, K(x , xi) 를 도 입 !)

Ex) 1. Polynomial Kernel

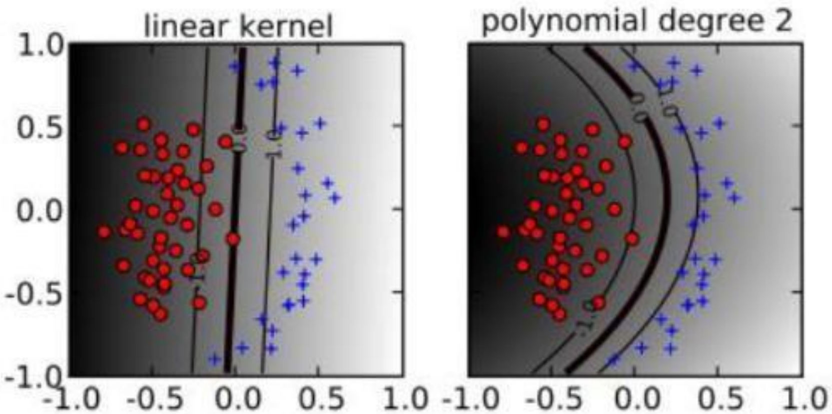
$$K(x, x') = (1 + \langle x, x' \rangle)^d$$

2. RBF (Gaussian) Kernel

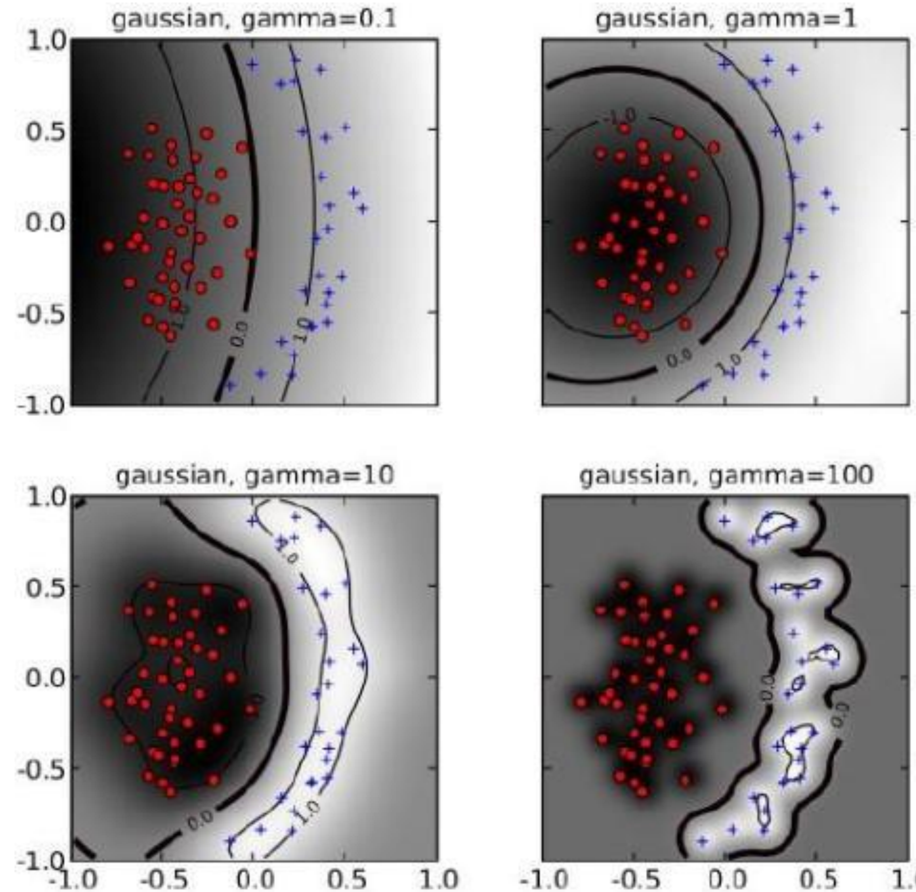
$$\exp(-\gamma \|x - x'\|^2)$$

3. Kernel SVM

Polynomial Kernel



RBF (Gaussian) Kernel



3. Kernel SVM

Hyperparameter

1.kernel : input data를 어떠한 원하는 형태로 변형시킬까?

ex) linear, polynomial, RBF ..

(polynomial & RBF는 non-linear hyperplane에 유용함. 고차원에서 다룸)

2.regularization (C) : penalty parameter

(C가 클수록 오분류에 엄격)

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i$$

3. gamma : 얼마나 fitting시킬지! rbf kernel폭의 역수에 해당함.

(지나치게 크면 overfitting 위험)

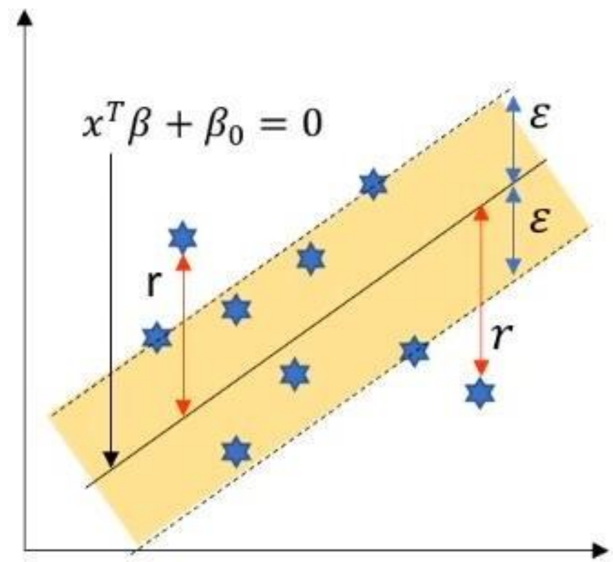
Ex) 1. Polynomial Kernel

$$K(x,x') = (1 + \langle x, x'\rangle)^d$$

2. RBF (Gaussian) Kernel

$$\exp(-\gamma\|x - x'\|^2)$$

4. SVR



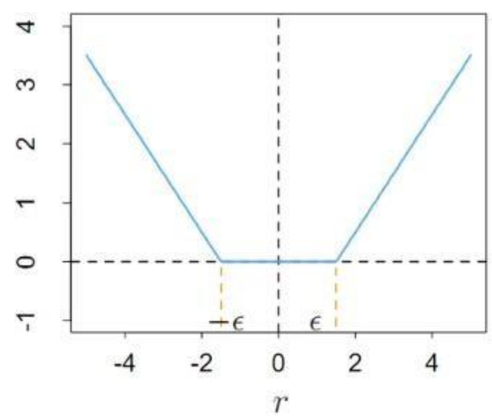
SVM과 같은 hyperplane 식

$$f(x) = x^T\beta + \beta_0 = 0$$

<u>cost function</u>

$$\boxed{H(\beta, \beta_0) = \sum_{i=1}^{N} V(y_i - f(x_i)) + \frac{\lambda}{2}\|\beta\|^2}$$

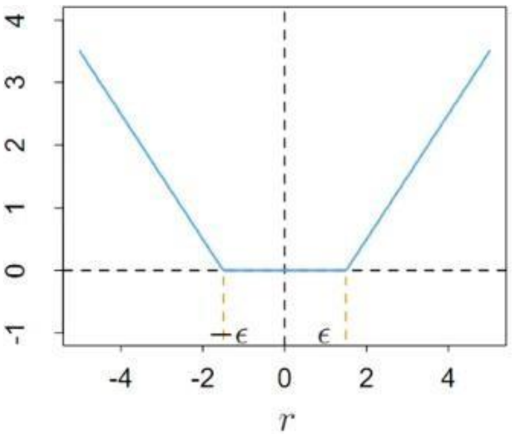
$$V_\epsilon(r) = \begin{cases} 0 & \text{if } |r| < \epsilon, \\ |r| - \epsilon, & \text{otherwise} \end{cases}$$

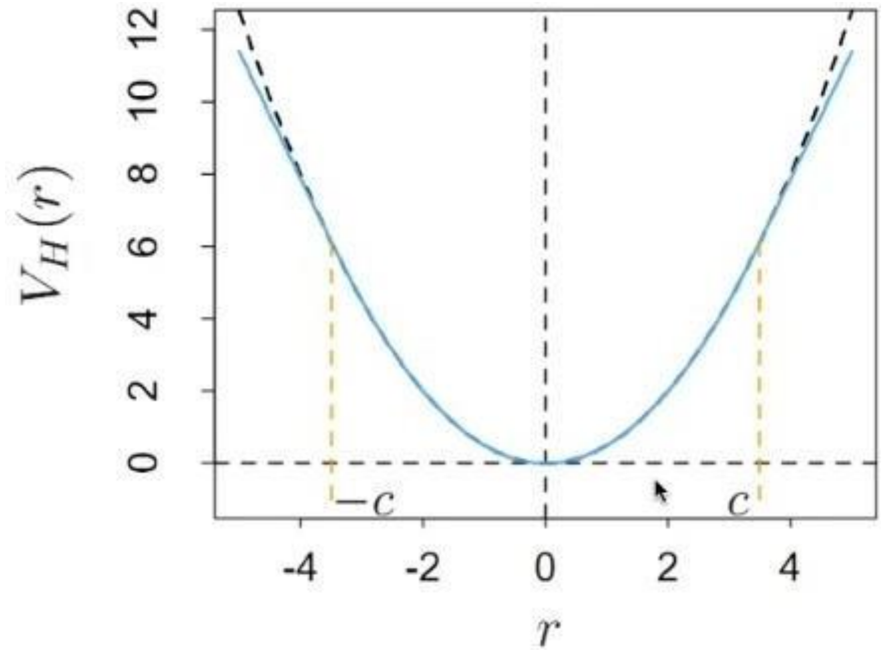


마찬가지로, 적당한 error 허용!
(그 바깥에 벗어나는 것 만 cost 계산 시 반영)

4. SVR

for 계산 상의 편의 →





$$V_\epsilon(r) = \begin{cases} 0 & \text{if } |r| < \epsilon, \\ |r| - \epsilon, & \text{otherwise} \end{cases}$$

$$V_H(r) = \begin{cases} r^2/2 & \text{if } |r| \leq c, \\ c|r| - c^2/2, & |r| > c, \end{cases}$$

4. SVR

$$\hat{f}(x) \quad = \quad \sum_{i=1}^{N}(\hat{\alpha}_i^* - \hat{\alpha}_i)\underline{\langle x, x_i\rangle} + \beta_0$$

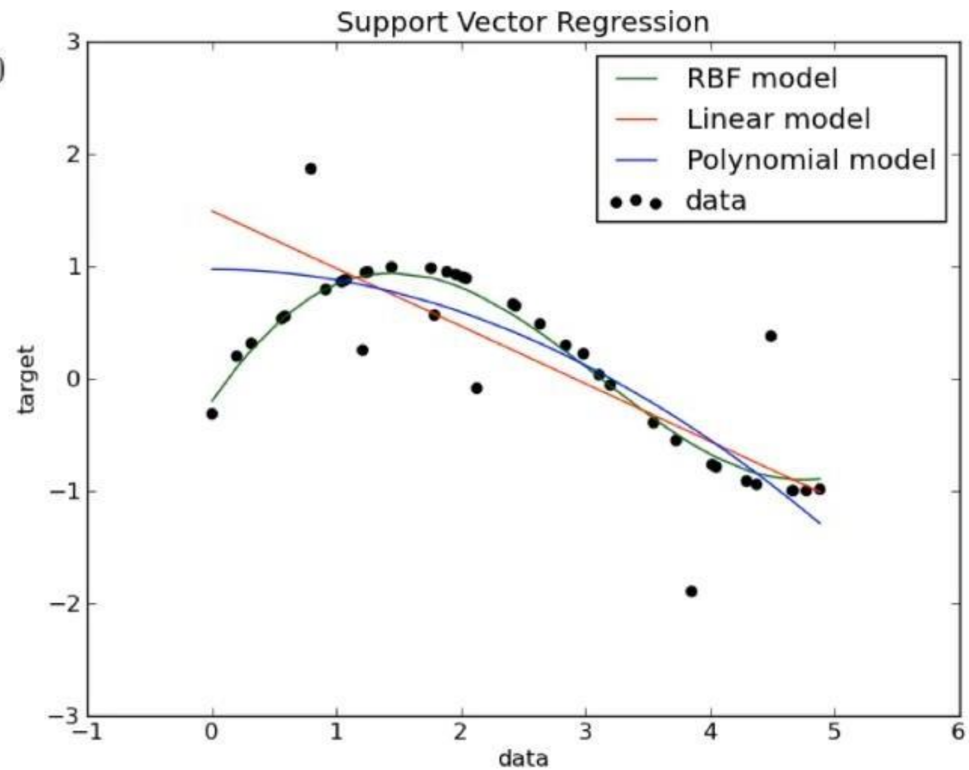
SVM과 마찬가지로 Kernel 적용 가능!

RBF

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/s^2)$$

Polynomial

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^d$$



5. Summary

[4줄 요 약]

1. SVM의 장.단점

- 장) data의 분포 가정이 힘든 경우good
- 단) C값을 직접 결정해야

2. Hyperplane을 통해data를 구분함

(적정한error을 허용하는flexible한 model)

3. Kernel을 사용하여 보다 복잡한 model 생성 가능

4.Classification 뿐만 아니라, Regression 에서도 가능하다! (SVR)

(but 주로 classification에서 많이 이용)

Thank You