15기 정규세션

ToBig's 14기 정재윤

SVM

Support Vector Machine

Contents

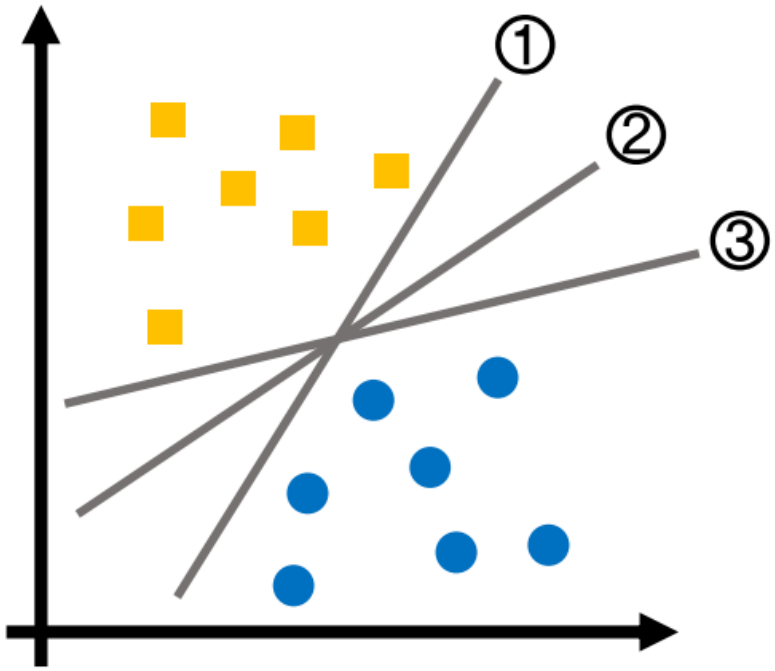
Unit 01 | Support Vector Machine이란?

Support Vector Machine

: 주로 바이너리 분류를 하기 위해 사용되는 기법!

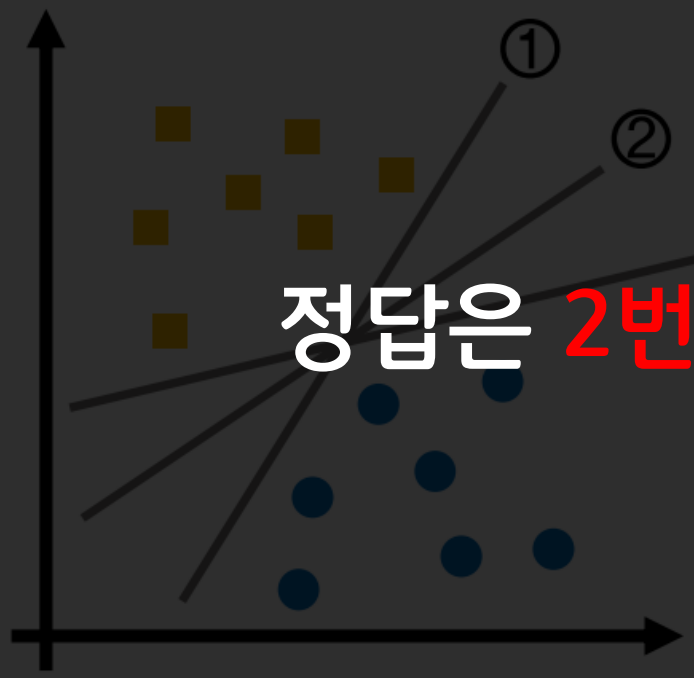
- Bell 연구소의 Vladimir Vapnik이 1963년에 개발하고, 1992년에 커널 트릭의 사용으로 발전
- Deep Learning이 사용되기 전까지 가장 성능이 좋은 모델로 평가받음. 하지만 연산이 오래 걸리는 단점
- 분류 문제에서 뛰어난 성능을 보인다. 여러분들 모두 한번쯤 접해봤을 기법

Unit 01 | Support Vector Machine이란?



데이터들을 가장 잘 나눈 선은
어떤 선일까요?

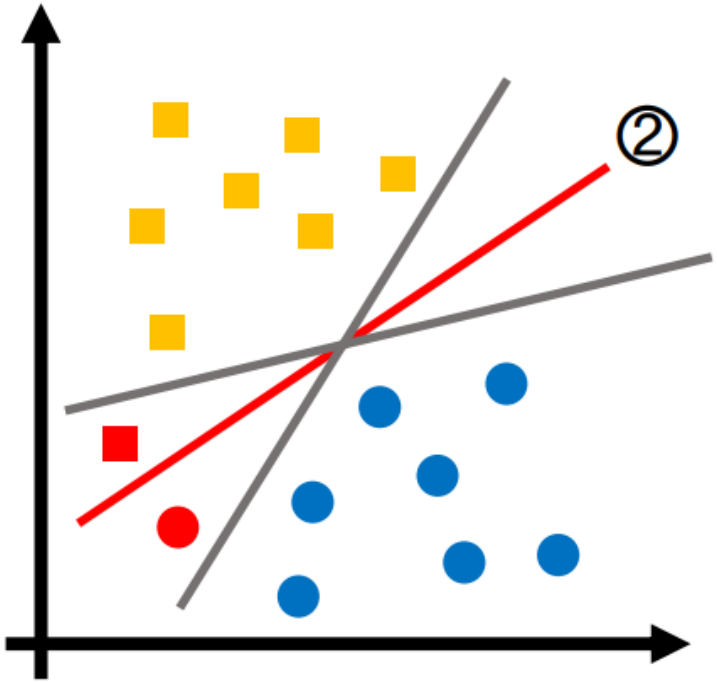
Unit 01 | Support Vector Machine이란?



정답은 2번!!! 그 이유는?? 🤧🤧🤧

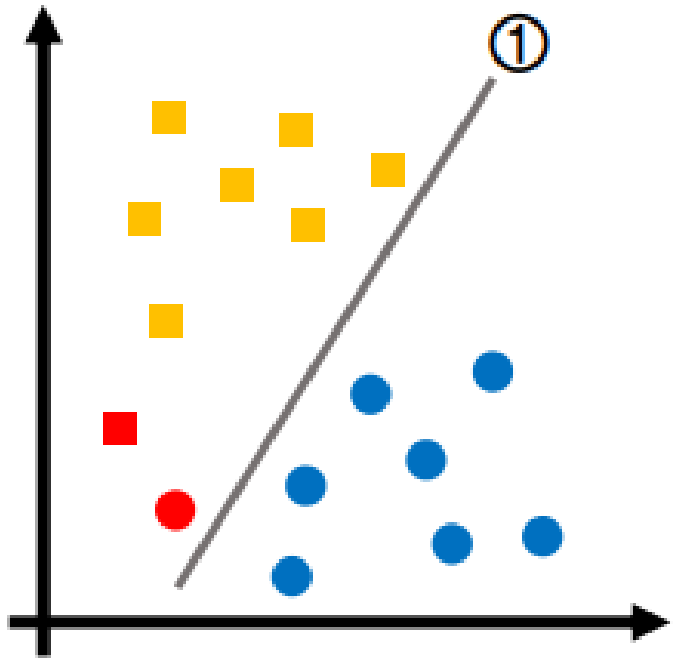
데이터들을 가장 잘 나눈 선은
어떤 선일까요?

Unit 01 | Support Vector Machine이란?



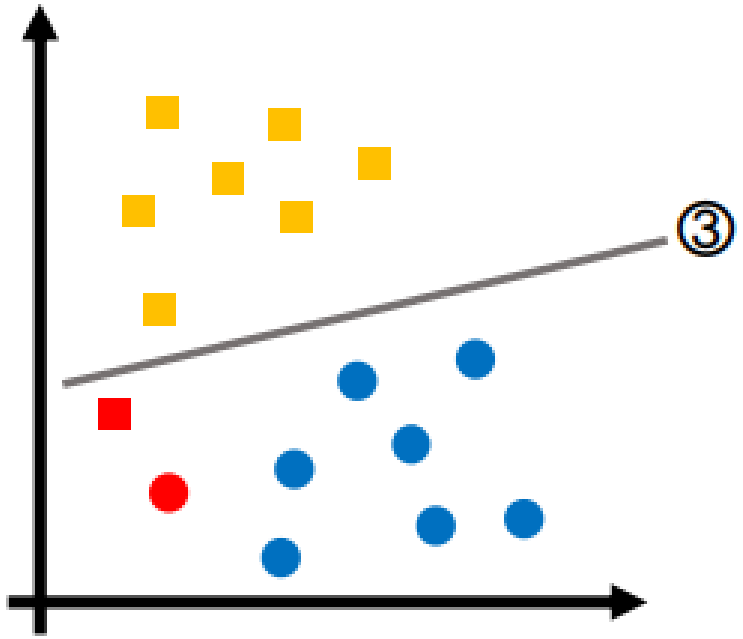
이렇게 2개의 데이터가
추가됐다고 합시다

Unit 01 | Support Vector Machine이란?



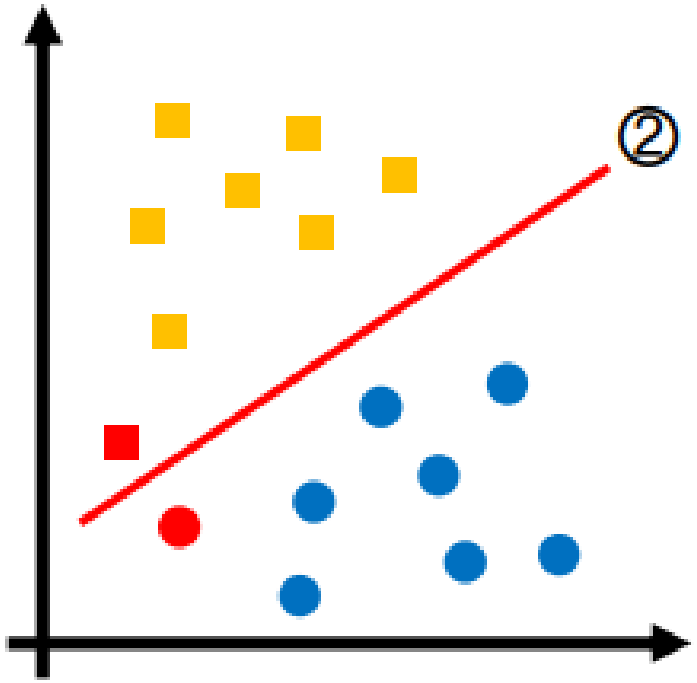
1번은 동그라미를 제대로 분류
하지 못합니다.

Unit 01 | Support Vector Machine이란?



3번은 사각형을 제대로 분류
하지 못합니다.

Unit 01 | Support Vector Machine이란?

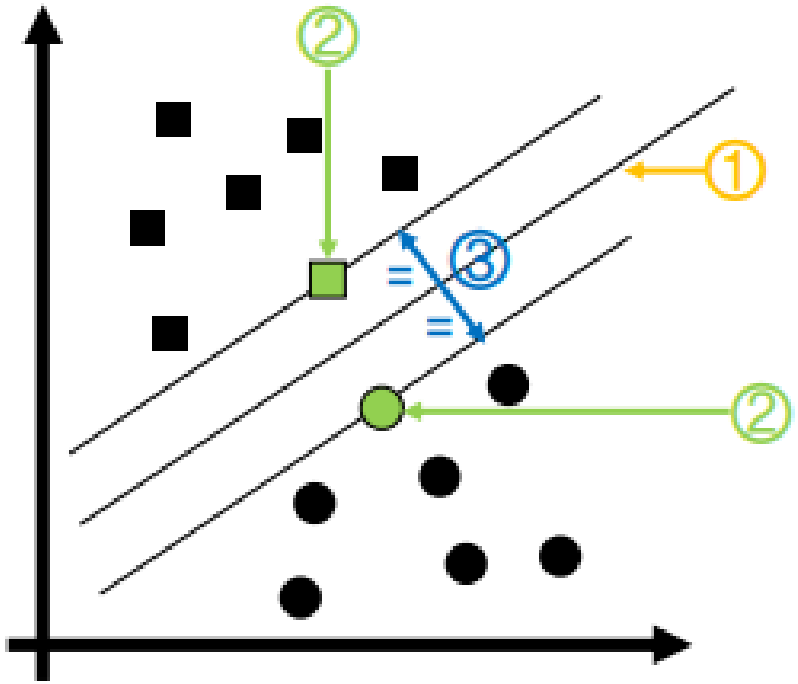


2번 직선은 데이터가 새로 들어왔을 때, 더 잘 분류할 가능성이 높다!

왜냐? 원래 있던 데이터들과 경계선이 충분히 많이 떨어져 있어서 모호한 데이터들이 들어와도 정확히 구분할 가능성이 높다!

즉, 결정경계와 데이터의 거리(여백)가 넓어서 좋다!!!

Unit 01 | Support Vector Machine이란?

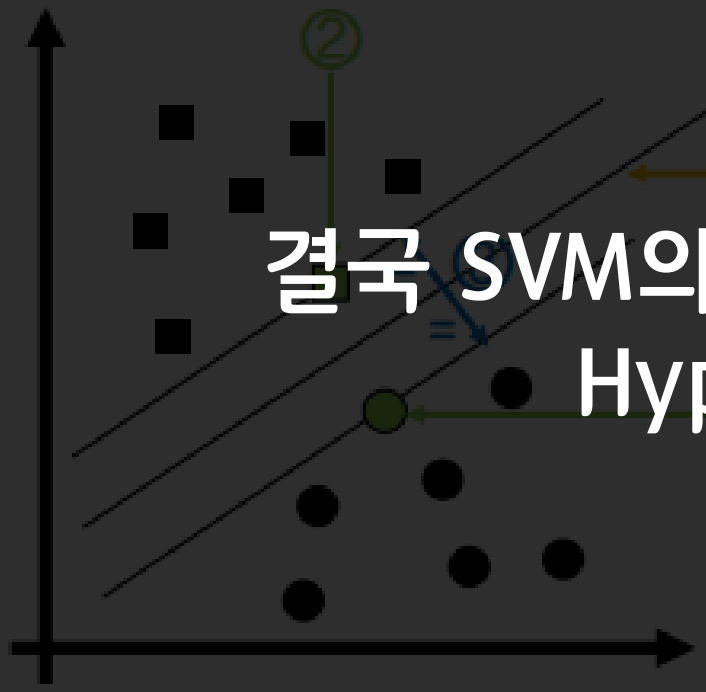


1. Hyperplane
: 여러 데이터를 나누는 기준이 되는 경계(초평면)

2. Support Vector
: Hyperplane과 가장 가까운 '데이터'

3. Margin
: 결정경계와 서포트 벡터 사이의 거리 X 2

Unit 01 | Support Vector Machine이란?



결국 SVM의 목적은 Margin을 최대화하는
Hyper Plane을 찾는 것!!

1. Hyperplane
: 여러 데이터를 나누는 기준이 되는 경계(초평면)

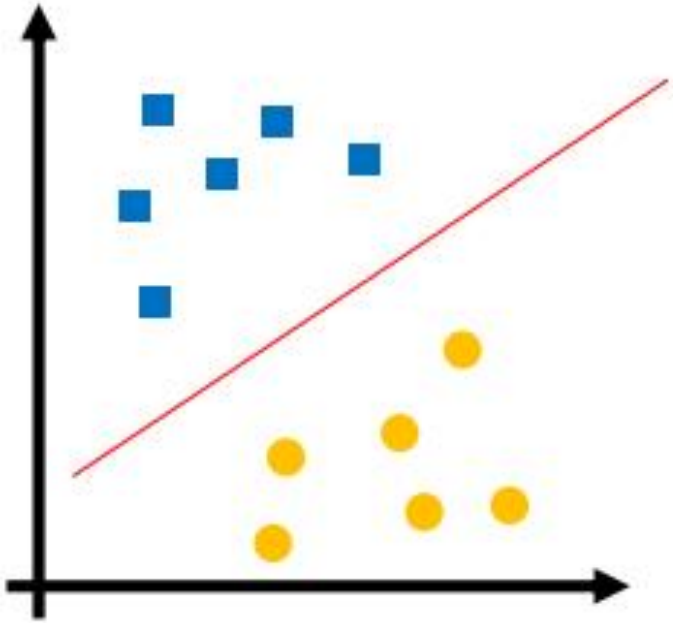
2. Support Vector
: Hyperplane과 가장 가까운 '데이터'

3. Margin
: 결정경계와 서포트 벡터 사이의 거리 X 2

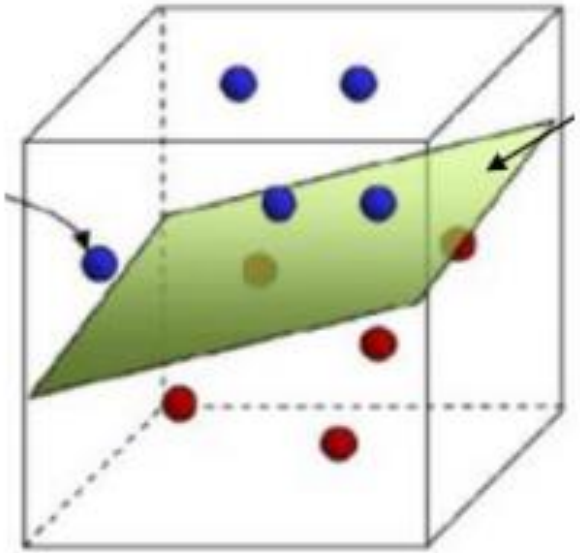
Unit 01 | Support Vector Machine이란?

1차원

2차원

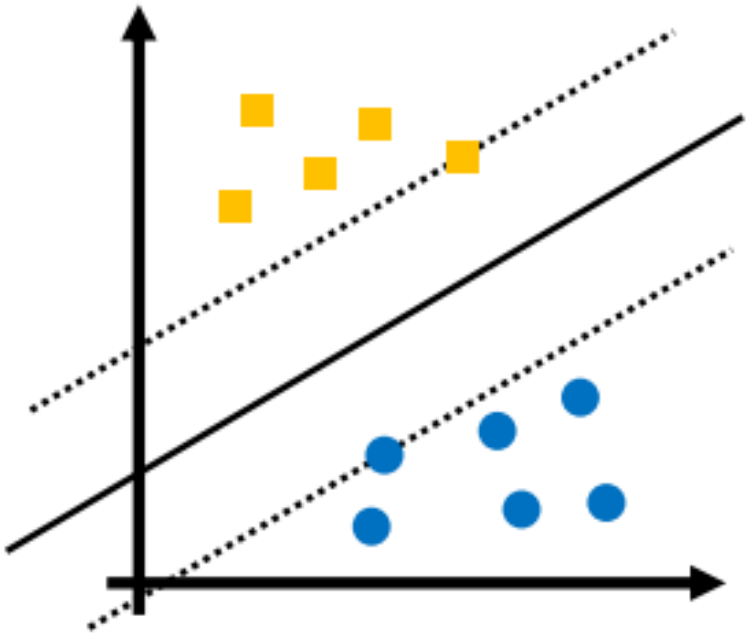


3차원



Unit 01 | Support Vector Machine이란?

차근차근 수식으로 알아보자



모든 plane은 W*X + b = 0으로 표현할 수 있다!

Ex) y = ax+b

$y = ax + b$ ⟶ 축을 x1,x2로 변경

$x_2 = ax_1 + b$ ⟶ 좌변으로 모두 넘겨준다.

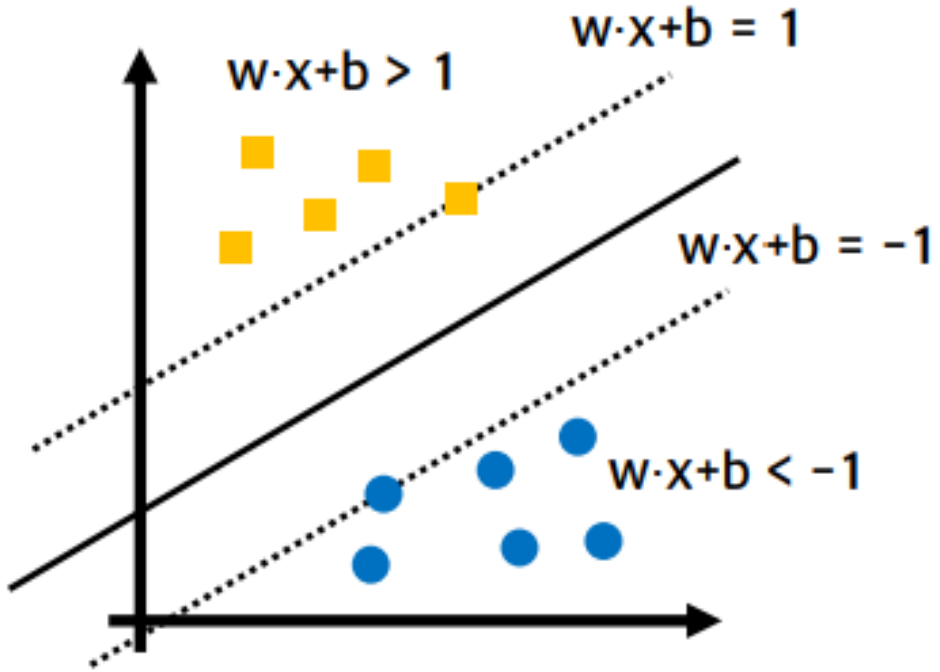
$ax_1 - x_2 - b = 0$ ⟶ 일반화를 위해 계수 변경

$a_1 x_1 - a_2 x_2 + b = 0$ ⟶ 행렬식으로 표현

$[a_1, -a_2] \begin{bmatrix} x1 \\ -x2 \end{bmatrix} + b = 0$ ⟶ $[a_1, -a_2]$ =W $\begin{bmatrix} x1 \\ -x2 \end{bmatrix}$ =X

Unit 01 | Support Vector Machine이란?

차근차근 수식으로 알아보자



편의상, y = +1 혹은 -1의 분류문제로 가정하겠습니다.

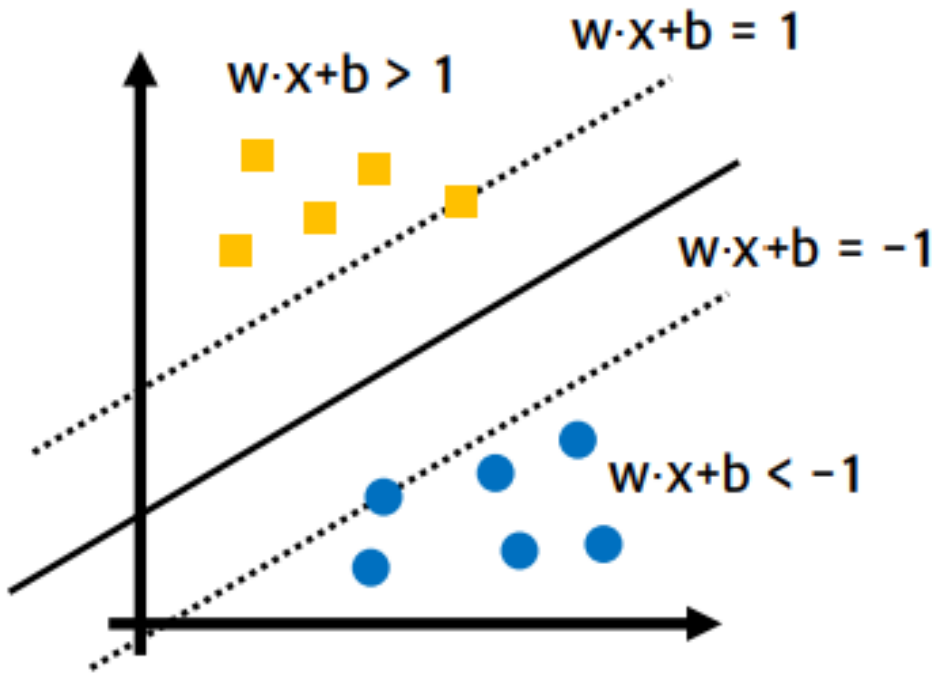
: ■ (+1) class라면 +1 이상,
● (-1) class라면 -1 이하의 값을 갖도록 하자

즉, 아래와 같은 조건으로 표현가능 합니다.

$$W X_+ + b \geq 1$$
$$W X_- + b \leq -1$$

Unit 01 | Support Vector Machine이란?

차근차근 수식으로 알아보자



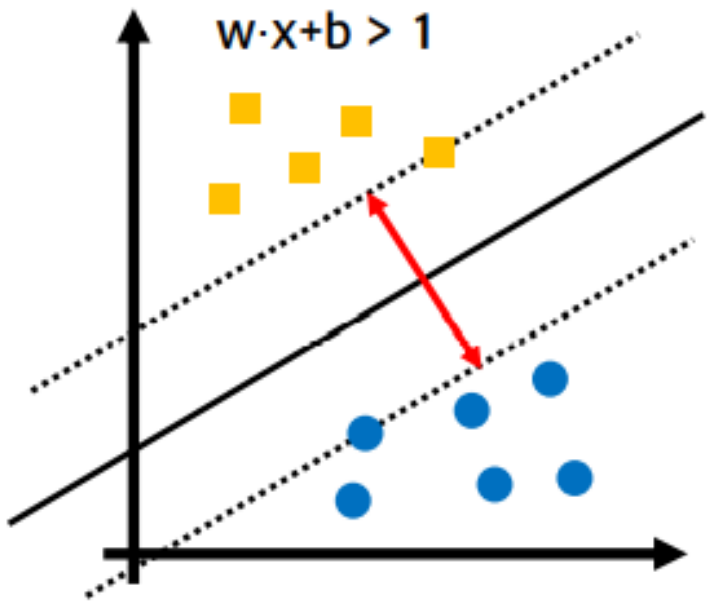
그러나 2개는 너무 번거롭습니다.
수학의 편리성을 위해서 y변수를 사용하여 아래와 같이 표현하겠습니다.

$$y_i(W X_i + b) \geq 1$$

이 때 y는 class에 따라 -1 or 1의 값을 가집니다.

Unit 01 | Support Vector Machine이란?

차근차근 수식으로 알아보자



길의 너비(margin)을 구하자

$$= (x_+ - x_-) \cdot \frac{w}{\|w\|}$$

$$= \frac{1-b-(-1-b)}{\|w\|}$$

$$= \frac{2}{\|w\|}$$

https://www.youtube.com/watch?v=I0f1o72aKDc
https://www.youtube.com/watch?v=Ls-yV4c0o8o

Unit 01 | Support Vector Machine이란?

지금까지의 과정을 중간 정리 해봅시다.

1. 제약식(조건)

$y_i(WX_i + b) \geq 1$ → 모든 데이터들은 각 영역안에 잘 나눠 들어가 있어야한다.

2. 목적

$\frac{2}{\|w\|}$ 가 최대가 되게 하는 것. 동시에 위의 제약식을 만족하는 W와 b를 찾아야 한다.

$\frac{2}{\|w\|}$ 의 최대값보다 $\frac{\|w\|^2}{2}$ 의 최소값을 찾자! (수학의 편의성, 계산을 쉽게 하기 위해)

Unit 01 | Support Vector Machine이란?

라그랑주 승수법

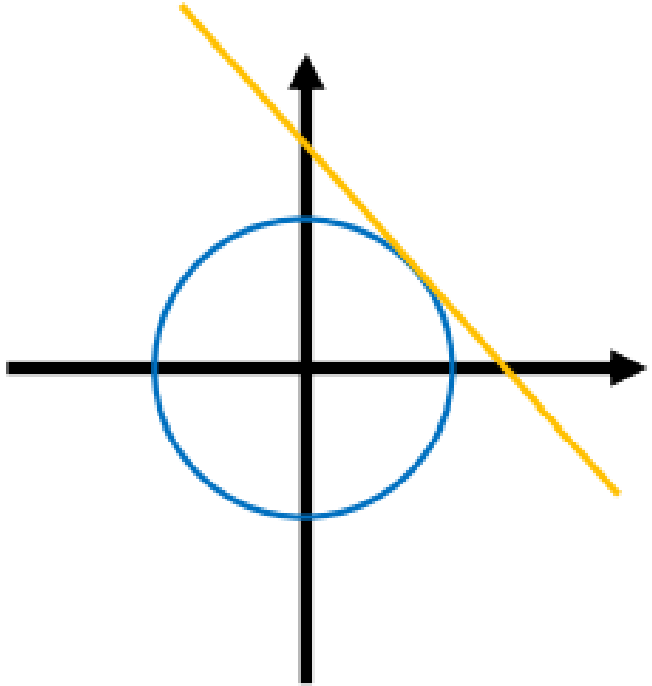
라그랑주 승수법?
어떤 f(x,y)를 최대화 or 최소화하는 문제가 있다고 합시다. 이 때 g(x,y)=0 이라는 제약식이 있을 때 이 둘을 하나로 합쳐서 계산하는 방법!

목적식 f의 gradient와 제약식 g의 gradient가 같을 때, 목적식의 최적값을 구할 수 있다! (자세한 내용은 링크의 영상을 봐주세요!)

$$\nabla_{x,y} f = \lambda \nabla_{x,y} g$$

따라서 L을 다음과 같이 정의하면 우리는 오른쪽과 같은 식을 만족하는 문제를 푸는 문제로 바꿀 수 있습니다.

$$\mathcal{L}(x, y, \lambda) = f(x, y) - \lambda g(x, y) \longrightarrow \nabla_{x,y,\lambda} \mathcal{L}(x, y, \lambda) = 0$$



https://www.youtube.com/watch?v=lmD9p6J_-TA

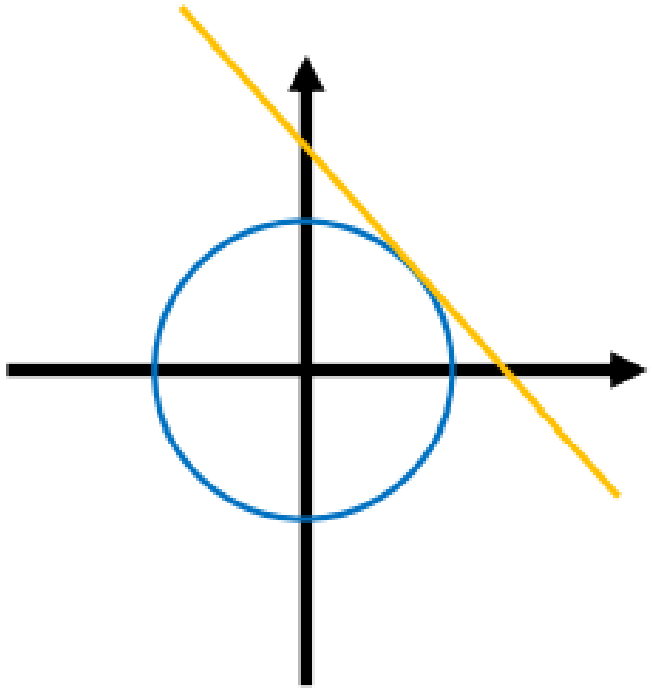
Unit 01 | Support Vector Machine이란?

라그랑주 승수법

그렇다면 우리의 목적식과 제약식은?

1. 제약식 : $y_i(WX_i + b) \geq 1$

2. 목적식 : $\frac{\|w\|^2}{2}$ 의 최소화



그런데?! 문제는 라그랑주 승수법은 등식인 데 반해, 우리는 제약식이 '부등식'…!

Unit 01 | Support Vector Machine이란?

KKT 조건

연립 부등식의 일 때도 라그랑주 승수법을 사용하되, 최적값이기 위한 필요충분조건인 KKT조건을 사용해야합니다. 조건과 결과값만 확인해보겠습니다.

(이 내용은 대학원 과정에 가까워지기도 하고, 전체 흐름에 방해될 것 같으므로 넘어 갈게요.)

1. 라그랑주 승수를 제외한 변수에 대한 편미분 값은 0이 되어야 한다.
2. 라그랑주 승수는 0보다 크거나 같아야 한다.
3. 라그랑주 승수 혹은 제약식 중 하나는 무조건 0이 되어야 한다

-> 우리의 식에 대입해보면 변수는 W,b이고, 승수는 a, 제약식은 $(y_i(W X_i + b) - 1)$ 이다.

① $\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \alpha) = 0, \frac{\partial}{\partial b} L(\mathbf{w}, b, \alpha) = 0$

② $\alpha_i \geq 0, \ i = 1, 2, \cdots, N$

③ $\alpha_i \left(y_i \left(\mathbf{w}^{\mathrm{T}} \mathbf{x}_i + b \right) - 1 \right) = 0, \ i = 1, 2, \cdots, N$

https://ratsgo.github.io/convex%20optimization/2018/01/26/KKT/

Unit 01 | Support Vector Machine이란?

결론!!

위 식들을 이용해서 제약식과 목적식을 정리하면 아래와 같이 정리됩니다!!

제약식 : $\Sigma\alpha_i y_i = 0 \ \alpha_i \geq 0 \text{ for all } \alpha_i$

목적식 : Maximize L(x,y,a) $= \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j$

제약식을 만족하는 동시에 목적식을 만족시키는(목적식을 최대화하는) a를 찾으면

우리가 구하고자 했던 W는 $\hat{w} = \Sigma\alpha_i y_i x_i$ 이고

분류결과는 $\mathrm{sgn}(\Sigma(\hat{\alpha}_i y_i \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j + b)$ 이 됩니다.

즉! Margin이 최대가 되는 w와 b를 찾는 문제는 a만을 찾는 문제로 바뀐다!

Unit 01 | Support Vector Machine이란?

결론!!

위 식들을 이용해서 제약식과 목적식을 정리하면 아래와 같이 정리됩니다!!

제약식 : $\Sigma\alpha_i y_i = 0 \ \alpha_i \geq 0 \text{ for all } \alpha_i$

목적식 : Maximize L(x,y,a) $= \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$

변수를 줄임으로써
컴퓨터가 계산하기 훨씬 편해진다!

제약식을 만족하는 동시에 목적식을 만족시키는(목적식을 최대화하는) a를 찾으면

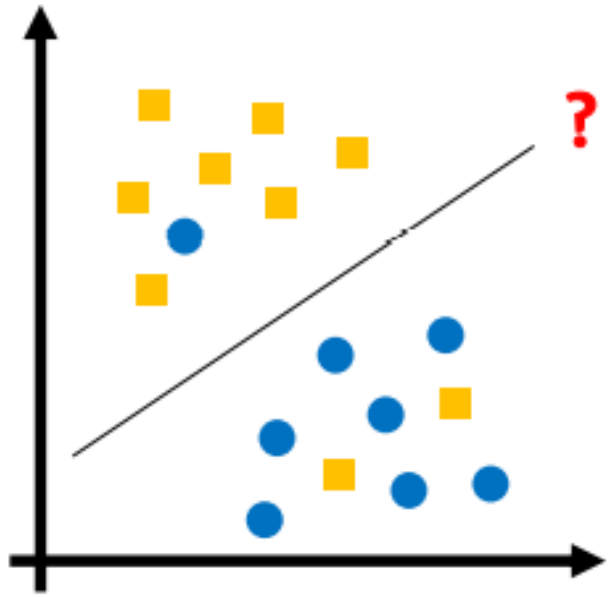
우리가 구하고자 했던 W는 $\hat{w} = \Sigma\alpha_i y_i x_i$ 이고

분류결과는 $\text{sgn}(\Sigma(\hat{\alpha}_i y_i x_i^T x_j + b)$ 이 됩니다.

즉! Margin이 최대가 되는 w와 b를 찾는 문제는 a만을 찾는 문제로 바뀐다!

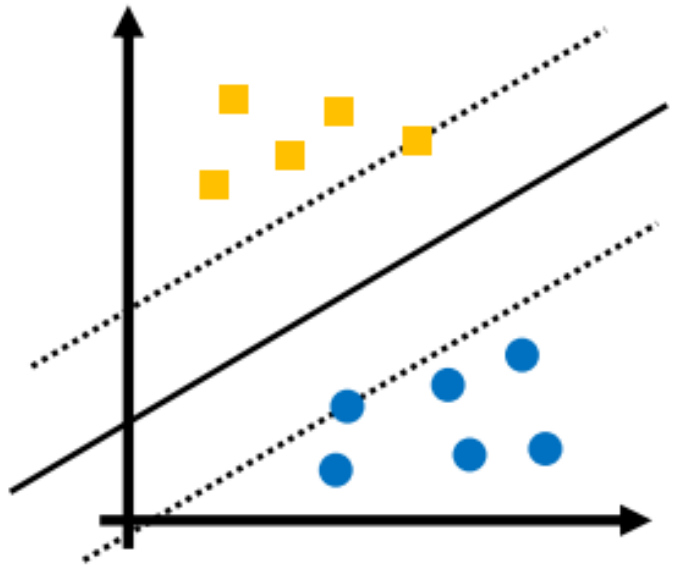
Unit 02 | Soft Margin SVM

SVM의 기본 개념은 이제 알았습니다!
그러나 현실적으로 모든 데이터들이 저렇게 이상적으로 나올까요…?
이런 데이터는 어떻게 해야할까?



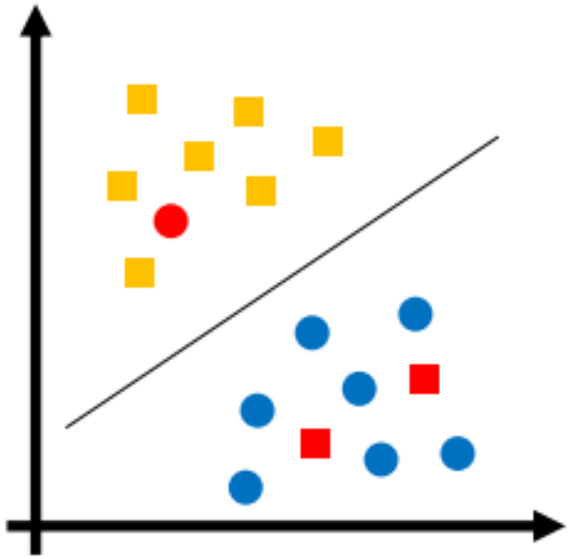
Unit 02 | Soft Margin SVM

저희가 지금까지 했던 과정은 바로 Hard Margin SVM입니다.
Error는 받아주지 않아! 모든 데이터를 정확하게 영역에 맞춰 분류!



Unit 02 | Soft Margin SVM

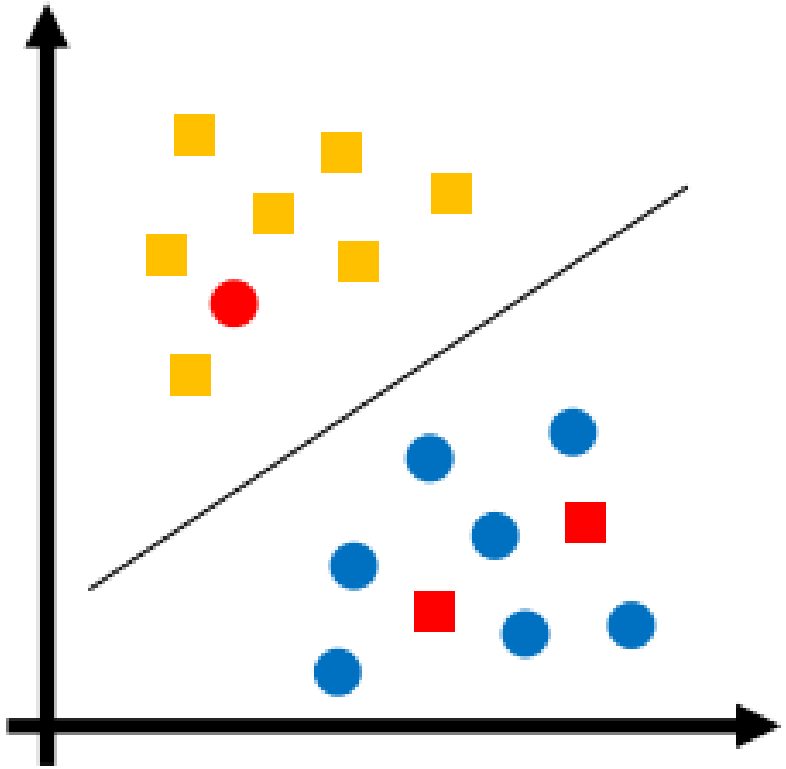
저런 데이터 몇 개 때문에 분류를 잘못했다라고 하는 건 기준이 너무 깐깐합니다.
그래서 에러를 허용하되, 이 에러들에 패널티를 부여해
에러를 최소화 하는 방법을 Soft Margin SVM이라고 합니다.



Unit 02 | Soft Margin SVM

Penalty를 부여하는 방법

0-1 Loss vs Hinge Loss

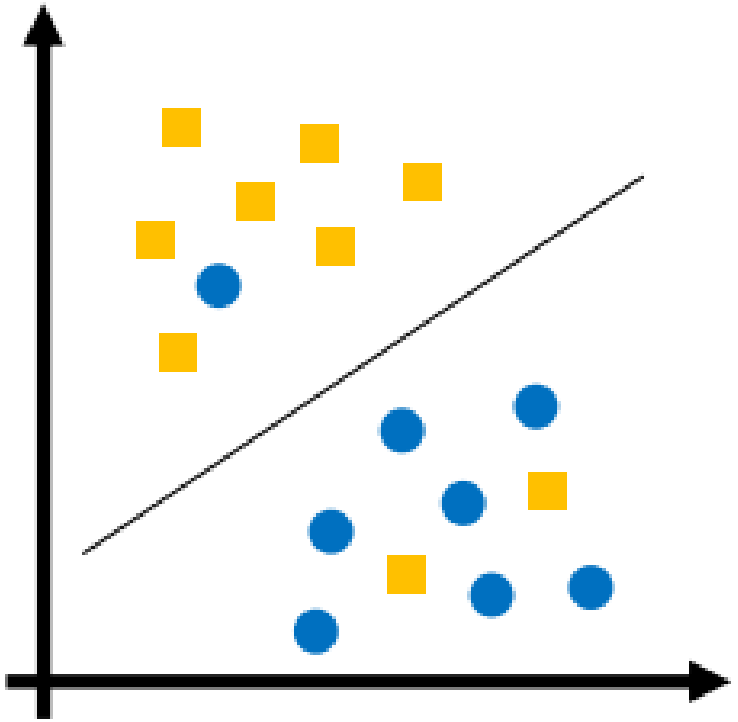


Unit 02 | Soft Margin SVM

0-1 Loss

: error가 발생한 개수만큼 패널티를 부여

: $\min||w|| + C\#error$



Unit 02 | Soft Margin SVM

Hinge Loss

: 오분류의 정도에 따라 error의 크기를 다르게 하는 것!

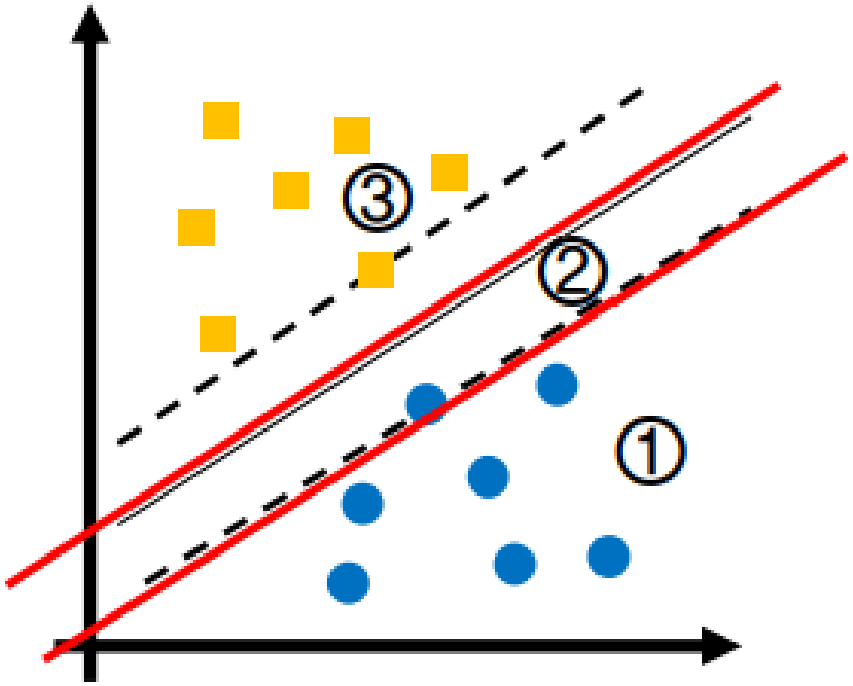
실제로는 ● 인 데이터가 분류기의 각 영역에 있을 때

① error 없음 (좋은 분류) > ξj=0

② 작은 error > $0 \leq \xi j \leq 1$

③ 큰 error (잘못된 분류) > $\xi j > 1$

> slack variable(ξj) 사용



Unit 02 | Soft Margin SVM

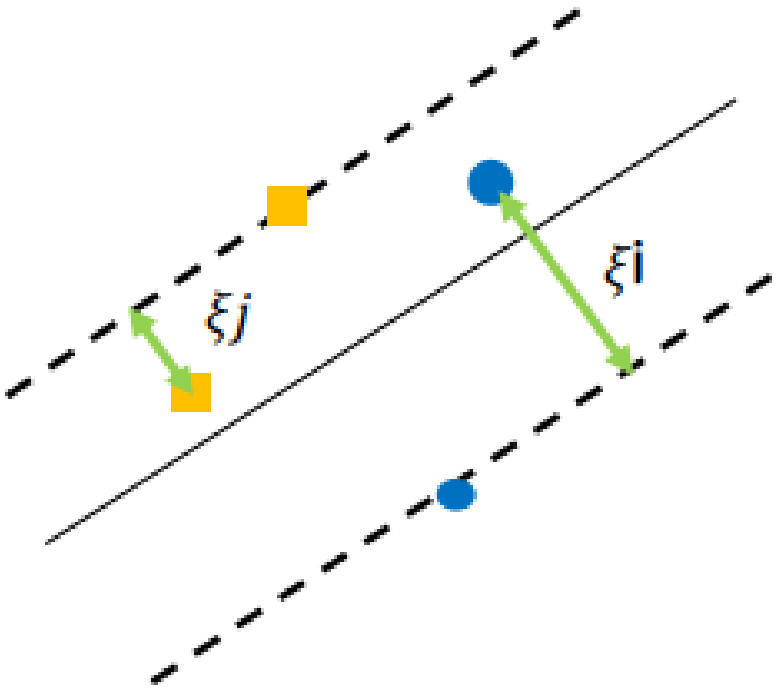
Hinge Loss

기존의 목적함수에 error항을 추가해준다!

: argmin||w|| + $c\Sigma\xi j$

여기서 c는 하이퍼 파라미터!! (추가자료 참고)

C가 크다? 에러를 줄이는게 중요! -> Hard margin
C가 작다? 에러가 있어도 큰 영향 x -> w를 줄이는게 중요



작은 C는 underfitting 가능성이 있고, 큰 C는 overfitting 가능성이 있다!

Unit 03 | Non-Linear SVM

그렇다면 이건 어떻게 분류가 가능할까요?

Unit 03 | Non-Linear SVM

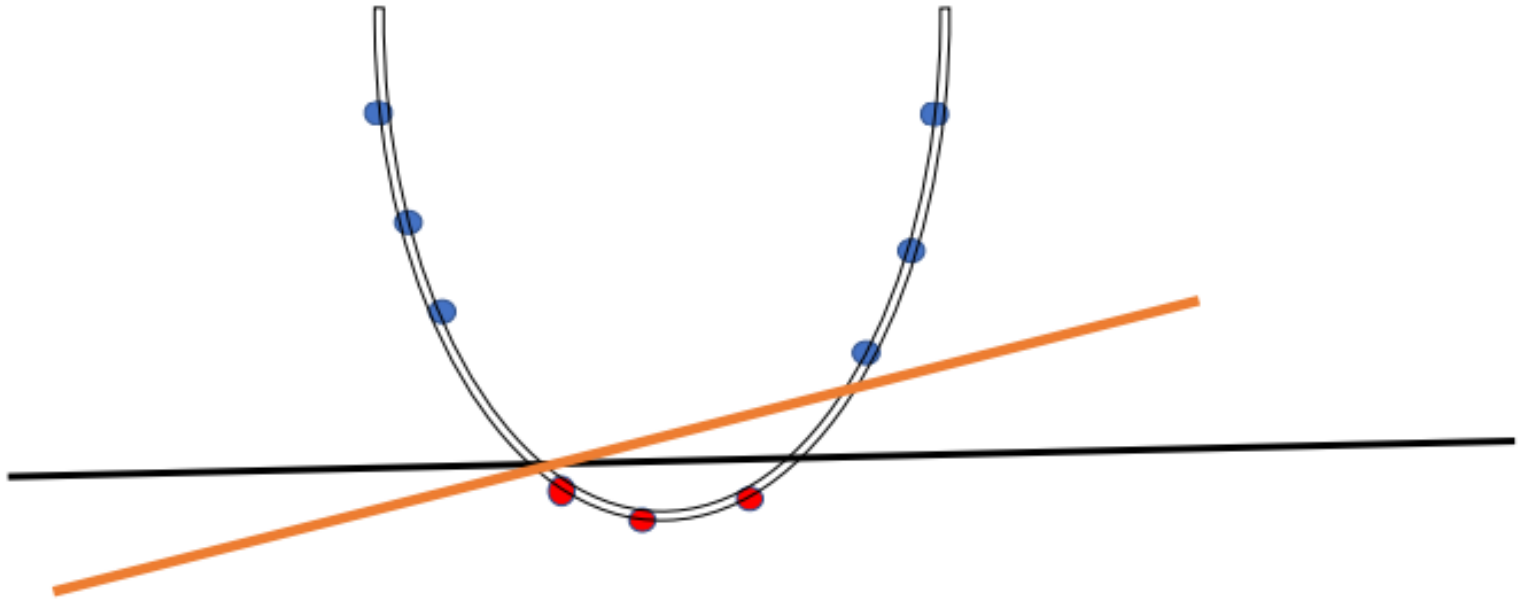
그렇다면 이건 어떻게 분류가 가능할까요?

못합니다.

Unit 03 | Non-Linear SVM

하지만 1차원에 있던 데이터들을 2차원으로 바꿔버리면 분류가 가능해진다!!

-> Mapping

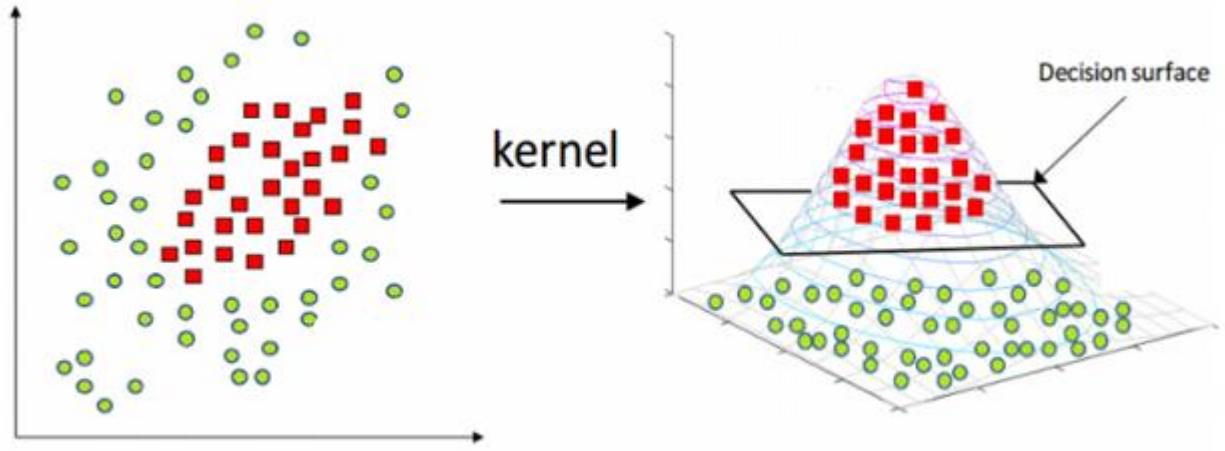


Unit 03 | Non-Linear SVM

Kernel

: 저차원 데이터들을 고차원 데이터로 매핑하는 것! -> x들을 높은 차원으로 보낸다!

: 대부분의 문제들은 Linear하게 해결되지 않으므로, original space가 아닌 고차원의 feature space로 매핑하자!



Kernel

목적함수를 다시 한 번 봅시다! 수식에 내적 부분이 있네요!

$$: \max L(\alpha_i) = \Sigma\alpha_i - \tfrac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathrm{x_i}^{\mathrm{T}}\mathrm{x_j}$$

$$\Rightarrow \max L(\alpha_i) = \Sigma\alpha_i - \tfrac{1}{2}\Sigma\sigma\alpha_i\alpha_j y_i y_j \phi(\mathrm{x_i})^{\mathrm{T}}\phi(\mathrm{x_j})$$

그런데 고차원 mapping을 하면 내적의 차원이 증가할 겁니다.
-> 연산량이 증가할텐데?

Kernel Trick

고차원으로 매핑한 '데이터'말고 '내적값'을 알자!!

-> 즉, **매핑 후 내적**이 아닌 **내적 후, 고차원 데이터 내적값과 같게 만들기!**

: $\max L(\alpha_i) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j x_i^T x_j$ => $\max L(\alpha_i) = \Sigma\alpha_i - \frac{1}{2}\Sigma\sigma\alpha_i\alpha_j y_i y_j \phi(x_i)^T \phi(x_j)$

즉, 매핑한 함수/데이터는 불필요. 우리는 매핑한 내적값만을 알면 된다.

Kernel Trick

Ex) $\varphi < x_1, x_2 > = < {x_1}^2, \sqrt{2x_1x_2}, {x_2}^2 >$ 로 Mapping 한다고 하자..

Kernel Trick

Kernel Trick을 사용하는 이유?
: 고차원 매핑도 가능하고, 연산도 간단하게 할 수 있으니까!

이 Kernel Trick을 만족하는 조건은 복잡하고, 저희는 사용하는게 정해져 있으니 자주 사용하는 Kernel만 알아봅시다.

대표적인 kernel

$$
\begin{aligned}
linear &: \quad K(x_1, x_2) = x_1^T x_2 \\
polynomial &: \quad K(x_1, x_2) = (x_1^T x_2 + c)^d, \quad c > 0 \\
sigmoid &: \quad K(x_1, x_2) = \tanh\left\{a(x_1^T x_2) + b\right\}, \quad a, b \geq 0 \\
gaussian &: \quad K(x_1, x_2) = exp\left\{-\frac{\|x_1 - x_2\|_2^2}{2\sigma^2}\right\}, \quad \sigma \neq 0
\end{aligned}
$$

가장 많이 사용하는 Kernel은 가우시안 커널 (RBF)! 다른 것들도 튜닝이긴 하지만 굳이…

각 함수마다 파라미터가 조금씩 다릅니다

Unit 03 | Non-Linear SVM

RBF kernel(= Gaussian Kernel)

: 무한대의 차원으로 매핑하는 커널(by 테일러급수)

$$2\sigma^2 = 1 \Rightarrow \quad K(x_1, x_2) = exp\left\{-(x_1 - x_2)^2\right\}$$
$$= exp(-x_1)exp(-x_2)exp(2x_1x_2)$$

여기서, 테일러 급수에 의해

$$exp(2x_1x_2) = \sum_{k=0}^{\infty} \frac{2^k x_1^k x_2^k}{k!}$$

> 무한대 차원의 Feature Space로 매핑!

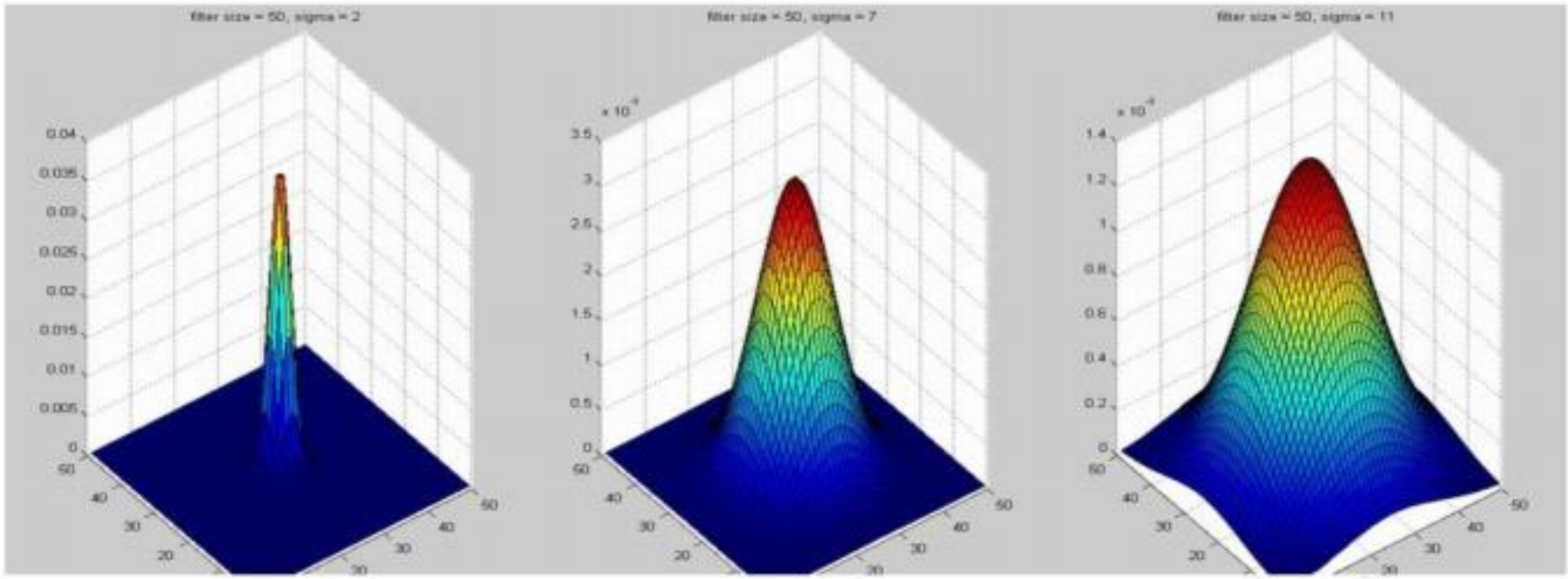
https://www.youtube.com/watch?v=5un8tY_CROE

RBF kernel – Gamma Parameter

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

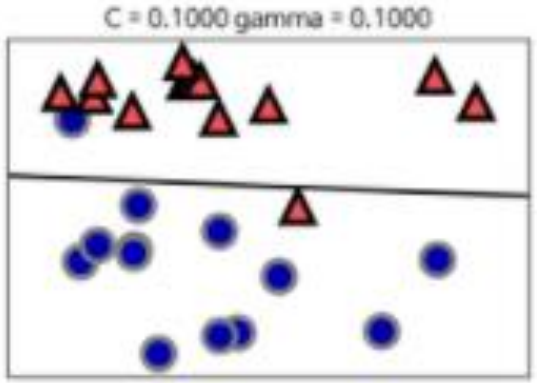
=> $\gamma = \frac{1}{2\sigma^2}$ => 감마와 분산(표준편차)은 반비례 관계

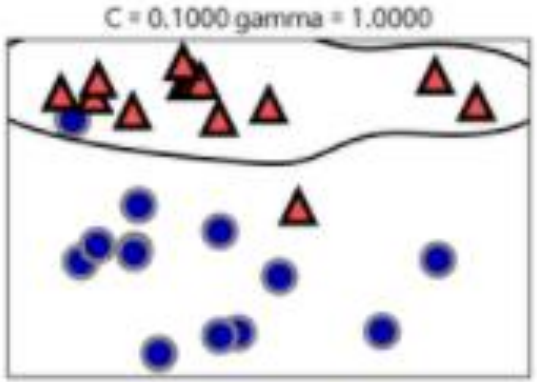
RBF kernel – Gamma Parameter

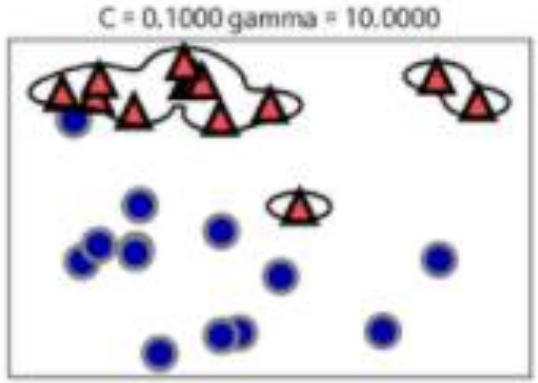


RBF kernel – Gamma Parameter

따라서 감마가 클수록 인접한 것들만 같은 영역으로 본다!
-> 엄청 인접하지 않으면 엄청 먼 곳으로 인식한다.
-> 다시 원래 차원으로 돌리면 경계가 촘촘하다.
-> Hyper plane이 훨씬 굴곡지다
-> Overfitting 가능성이 높다







Unit 04 | Summary

1. Hard margin과 Soft margin

- Hard Margin : 에러를 허용하지 않음 (현실 적용은 좀…)

- 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 많은데 Kernel을 하게 되면 차원이 높아지고 연산량이 증가하니 Linear가 효과적!

- Non-Linear SVM
: Feature가 많지 않을 때는 그걸로 분류하기가 쉽지 않아서 차원을 올려준다. 이 때 차원을 올려주는 방법에 따라 다양한 parameter가 존재하지만 가장 많이 쓰이는 건 Gaussian Kernel이고 그 중 Gamma!

Unit 04 | Summary

정리

1. SVM 알고리즘 중 일반적으로 많이 사용되는 건 RBF 커널 SVM
2. 좋은 성능을 얻기 위해선 C와 gamma를 잘 조정해야한다.
3. C는 데이터 샘플들의 오류를 허용하는 정도, gamma는 결정경계의 곡률을 결정
4. 두 값 모두 커질수록 알고리즘 복잡도가 올라간다!
5. 일반적으로 grid search를 사용해 최적값을 찾아가는데, 어느 정도 익숙해지면 훨씬 더 빠르게 좋은 성능을 내는 최적값을 찾아낼 수 있다!

Unit 04 | Summary

Multi Class SVM

1. One vs One
클래스가 N개 있을 때, 모든 Class에 대해 1:1로 binary분류를 하고 제일 많이 승리한 것에 대해 투표로 결정한다.
N개의 클래스에 대해 서로서로 Classifier를 가지고 있어야 하기 때문에 n(n-1)/2개의 Classifier가 필요하다.

2. One vs Rest
Class가 N개 있으면 모든 Class에 대해 1:N-1로 binary 분류하여 이 클래스가 맞는지 아닌지를 판단하고 투표로 결정하고, 이 때 N개의 Classifier가 필요하다.

Unit 04 | Summary

드디어 과제…

1. 위에서 언급한 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 C의 경우 A로 판별, B vs A C의 경우 B로 판별, C vs A B의 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 판별해주시면 됩니다. 공식문서를 통해 사이킷런이 어떤 방법으로 구현했는지 참고하셔도 됩니다

3. 과제코드에는 iris 데이터를 로드하고 one hot 인코딩한 부분까지 구현되어 있습니다. 또한 decision function을 호출해서 사용하는 예시를 포함했으니 참고하시면 됩니다.

Reference

투빅스 12기 박진혁님, 13기 이유민님 svm 강의자료
고려대학교 김성범 교수님 svm 강의
KAIST 문일철 교수님 svm 강의
Patrick Winston, MIT OCW 6.034 Fall 2010, Lec.16 Learning: Support Vector Machines
앤드류응 교수님의 Machine Learning 강의

http://jaejunyoo.blogspot.com/2018/01/support-vector-machine-1.html
https://ratsgo.github.io/machine%20learning/2017/05/23/SVM/
https://ratsgo.github.io/machine%20learning/2017/05/29/SVM2/
https://ratsgo.github.io/convex%20optimization/2018/01/26/KKT/
https://ratsgo.github.io/convex%20optimization/2018/01/25/duality/
http://hleecaster.com/ml-svm-concept/
https://excelsior-cjh.tistory.com/165

참고자료

참고자료 1. Grid Search의 활용

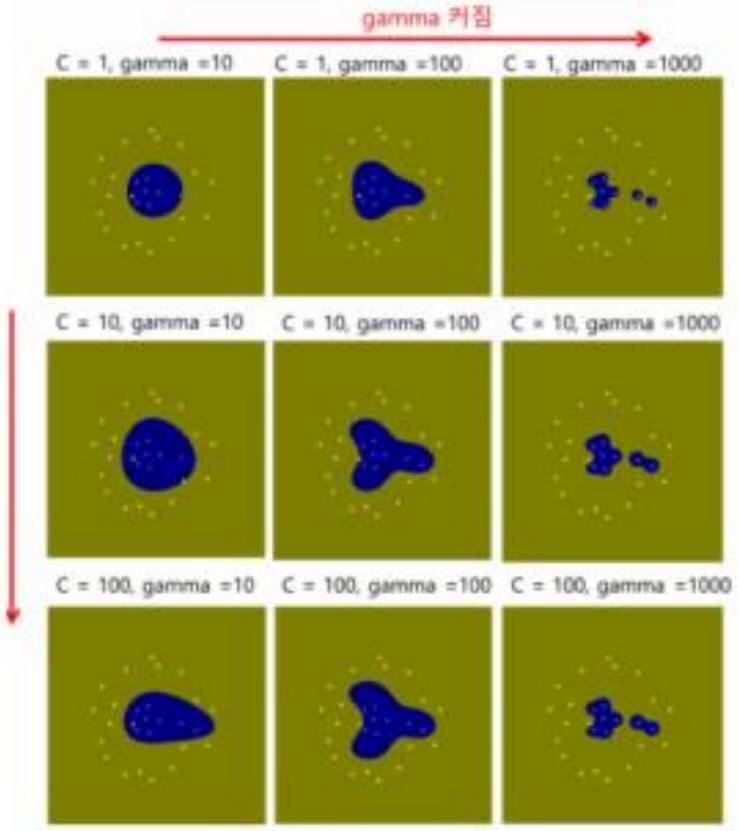


그림9. C와 gamma의 영향

하이퍼 파라미터가 두 개나 있어서 감이 안 올 때는?

C : 데이터가 다른 클래스에 놓이는 걸 허용하는 정도

γ : 결정경계의 곡률을 결정 (kernel모델의 파라미터)

> Overfitting 위험 : C가 크고 Gamma가 클 때

> underfitting 위험 : C가 작고 Gamma가 작을 때

참고자료 2.

앤드류응: Andrew Ng 교수님께서는

1) 피쳐가 데이터 개수에 비해 많으면 커널 없는 SVM 즉, linear kernel을 쓰고(피쳐 개수10000개정도)

2) 피쳐가 적고 데이터 개수는 보통이면 가우시안 커널을 적용한 SVM 즉, RBF커널 SVM을 쓰라고 하셨고(피쳐 개수 1000개정도, 데이터개수10 -10000개정도)

3) 피쳐가 적고 데이터가 많으면 커널 없는 SVM이나 로지스틱 회귀를 쓰라고 하셨으나, 이 세번째 부분은 커널 있는 SVM은 데이터가 많아지면 연산이 많아져 부담이 되기 때문이라고 하셨다.

따라서 연산의 부담이 좀 많아도 괜찮으면 RBF 커널도 무방할 듯 하다

(지극히 교수님의 경험에 의한 개인적인 생각!)

참고자료 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

: C 있음: soft / C 없음: hard margin svm

3. Gamma

: 결정경계의 굴곡에 영향을 주는 데이터의 범위(reach)를 정의

Reference

참고자료 One vs rest

One vs 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 돌린다.
1번은 A라고, 2번은 B가 아니라고, 3번은 C가 아니라고 했다.
이 결과를 통해 우리는 새로 들어온 데이터가 A라고 판별할 수 있다.

Reference

참고자료 One vs One

One vs One 예시)
데이터의 클래스로 A, B, C가 있다고 하자. 우리는 총 3개의 머신을 만든다.

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

새로운 데이터가 들어왔고, 우리는 이 데이터를 1,2,3번 머신에 돌린다. 1번은 A라고, 2번은 C라고, 3번은 A라고 했다. 이 결과를 통해 우리는 새로 들어온 데이터가 A라고 판별할 수 있다.

들어주셔서 감사합니다.