Support Vector Machine

ToBig's 10기 박규리

# Support Vector Machine

알고리즘 개발자가 아닌 사용자를 중심으로

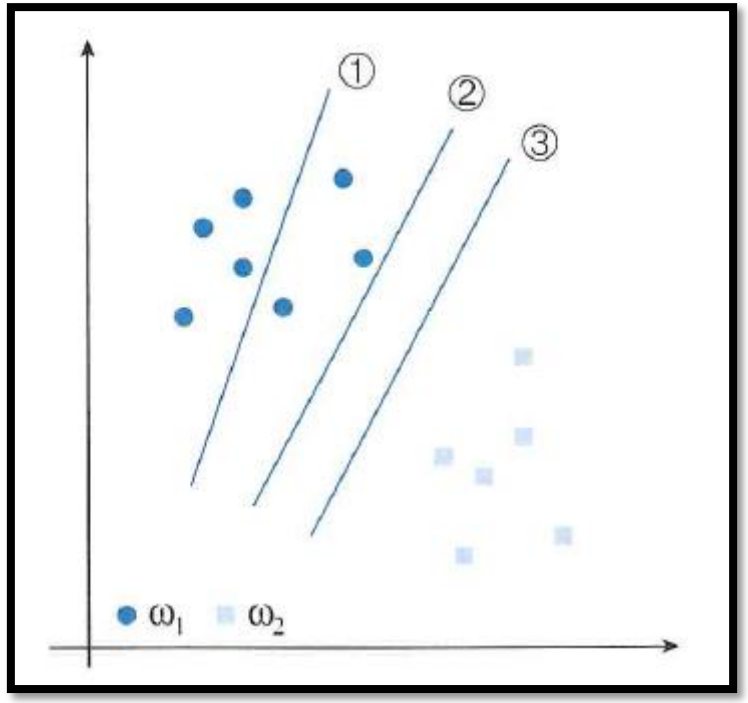
# Contents

# Linear SVM

# 선형 SVM

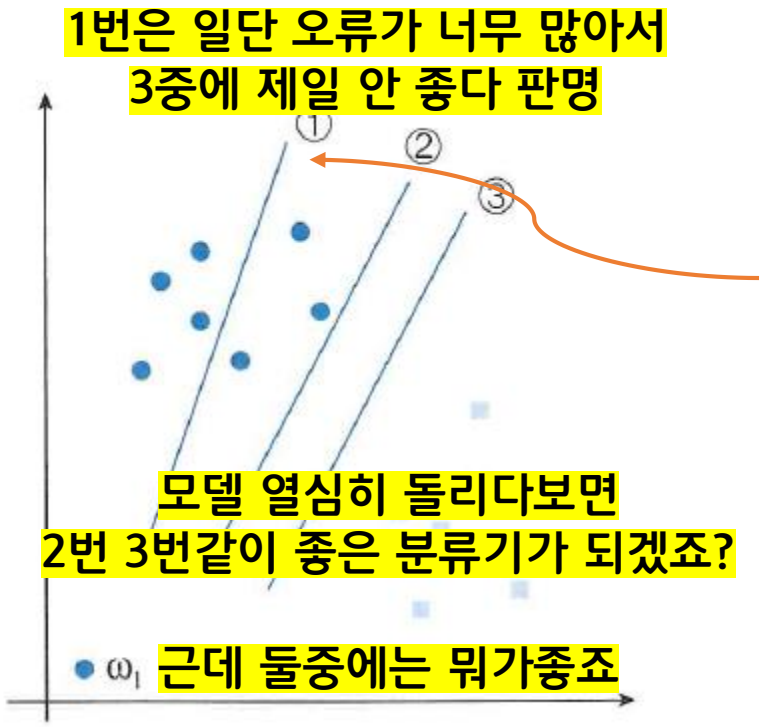
## 선형 SVM

SVM은 두 클래스를 분리하기 위한 선형 결정 함수를 찾는 것



# 선형 SVM

## 선형 SVM




SVM은 두 클래스를 분리하기 위한 선형 결정 함수를 찾는 것

① 분류기는 Train set을 **틀리게 분류**한다.

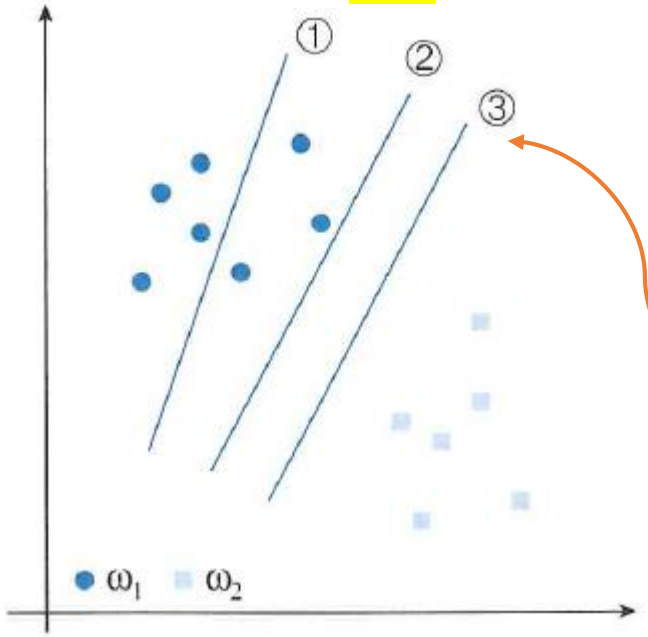
이를 여러번 학습시켜 모델링하면 ②와 ③ 분류기와 같이 될 것이다.

Train set 측면에서 보면 ②와 ③ 분류기는 오류가 0이므로 같은 성능을 가진 분류기로 볼 수 있다.

여백 : 두 데이터 군과 결정 경계와 떨어져있는 정도를 의미

## 선형 SVM

**3번이 나아보여요 딱봐도 왜죠**




SVM은 두 클래스를 분리하기 위한 결정 함수를 찾는 것

하지만, 일반화(generalization) 측면에서 보면 ② 보다 ③이 더 낫다고 할 수 있다.

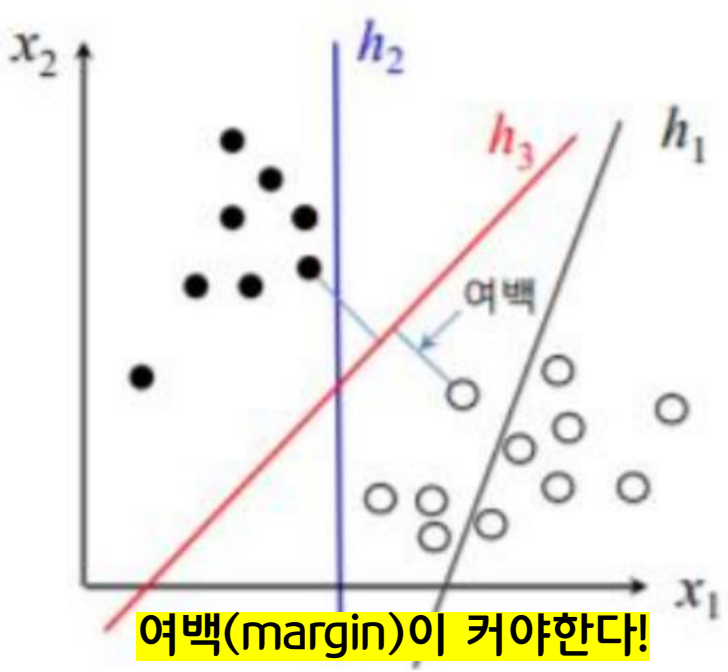
그 이유는

③ 분류기가 두 개의 클래스에 대해 여백(margin) 크기 때문이다.

여백 : 두 데이터 군과 결정 경계와 떨어져있는 정도를 의미

# 선형 SVM

## 선형 SVM




여백(margin)이 커야한다!

바로 여기서 이러한 여백, 즉 마진을 어떻게 공식화하고 이 마진을 최대화하는 결정 초평면(decision hyperplane)을 찾는 것이 바로 SVM의 발상

여백 : 두 데이터 군과 결정 경계와 떨어져있는 정도를 의미

# 선형 SVM

## 선형 SVM

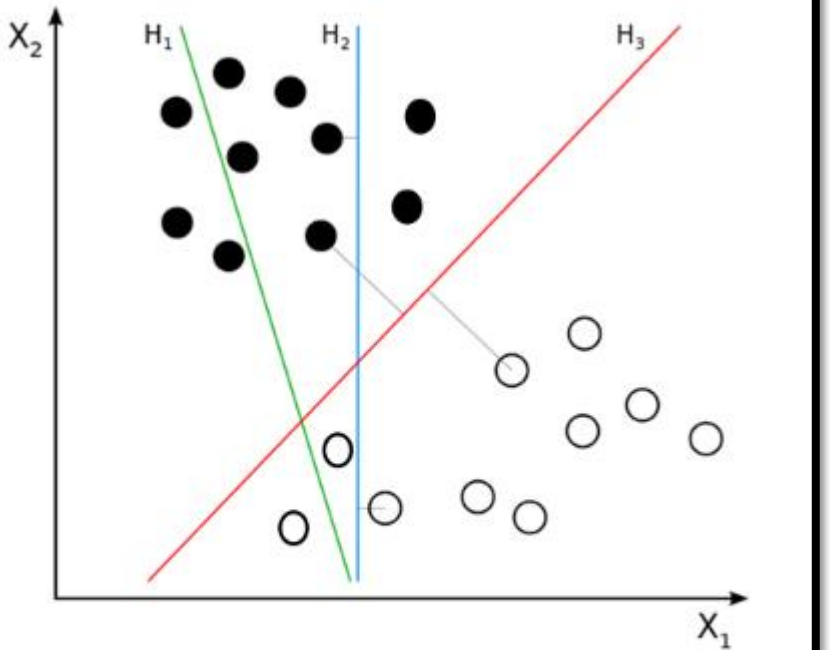



그림3. 4개의 데이터가 새롭게 추가된 상황에서 H3와 달리 H2는 데이터를 제대로 분류해낼 수 없는 것을 알 수 있다.

SVM은 두 클래스를 분리하기 위한 선형 결정 함수를 찾는 것

결정경계가 마진을 최대화해야, 즉 두 데이터 군과 결정 경계와 떨어져 있는 정도를 최대화해야, 이후에 새로운 데이터가 추가 되더라도 H3가 훨씬 더 안정적으로 데이터를 분류해 낼 수 있다

그래서 SVM의 목표는 마진이 가장 큰 결정 경계를 찾는 것

## 선형 SVM

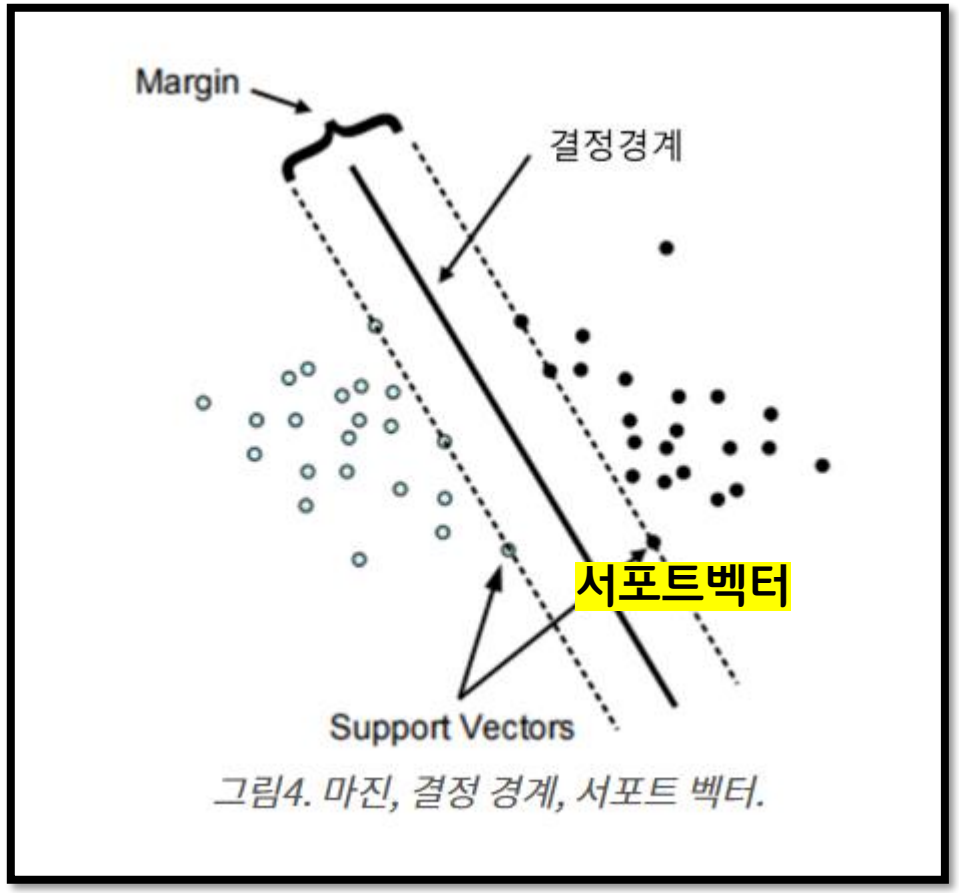


그림4. 마진, 결정 경계, 서포트 벡터.

서포트 벡터
두 클래스 사이의 경계에 위치한 데이터 포인트들
(점선 위에 있는 데이터들)

**서포트 벡터들이 결정 경계를 만드는데 영향을 준다.** 이 데이터들의 위치에 따라 결정 경계의 위치도 달라질 것이다. 즉, 이 데이터들이 결정 경계를 지지(support)하고 있다고 말할 수 있기 때문에, 서포트벡터라고 불리는 것

## 선형 SVM

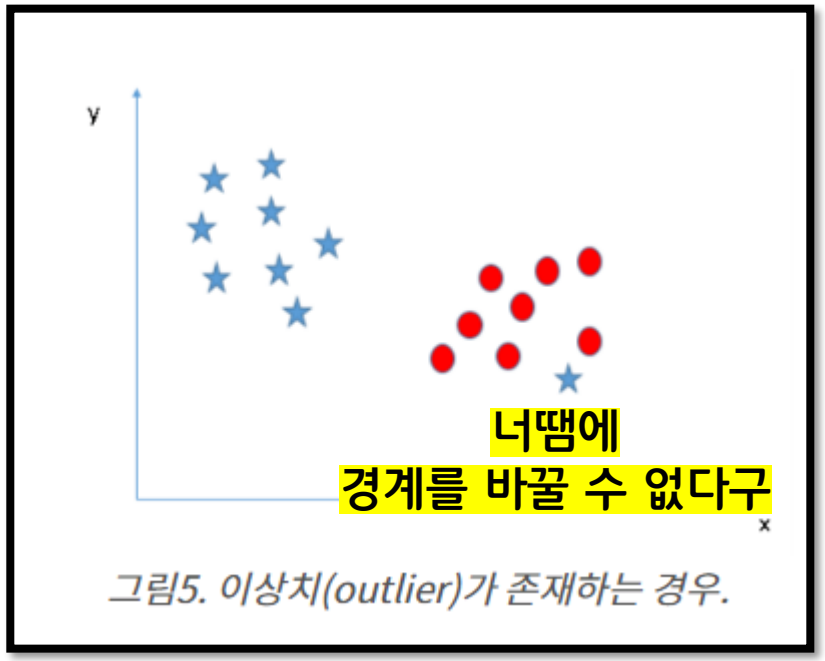


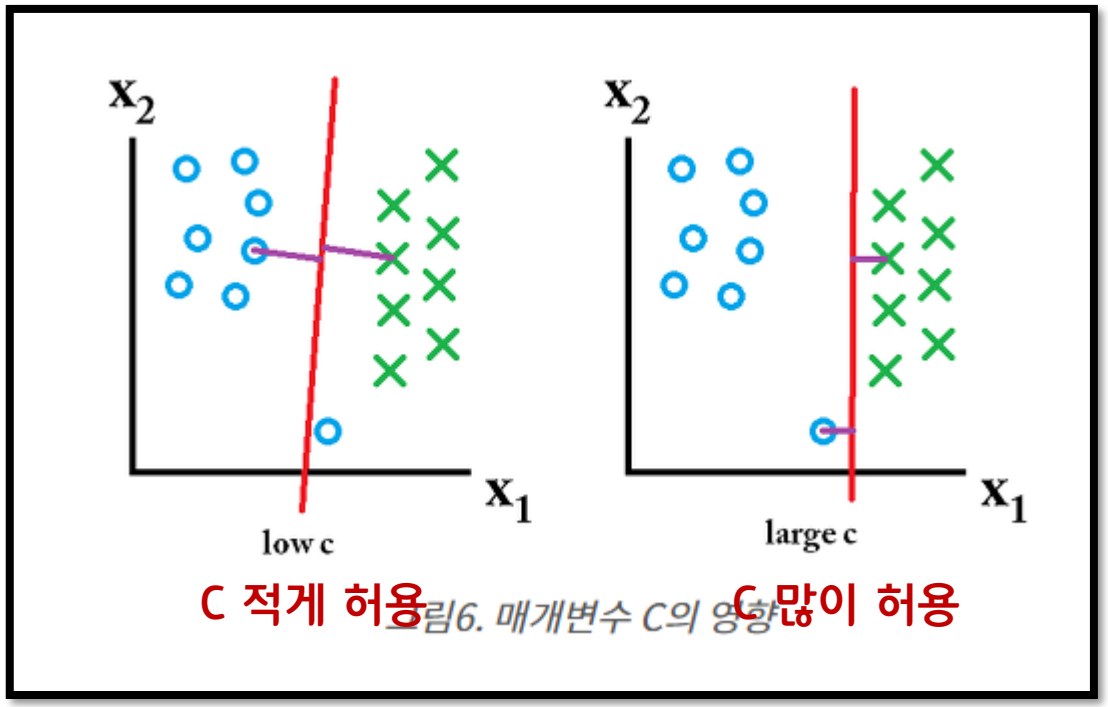

그림5. 이상치(outlier)가 존재하는 경우.

대부분의 데이터는 위에서 본 데이터와 다르게 이상적으로 분리되어 있지 않다. 많은 경우 이상치(outlier)들이 관측된다.

이러한 이상치 하나(혹은 몇 개)로 인해 결정경계를 바꾸는 일이 옳은 일일까? 라는 의문이 들 수 있다

그래서 이러한 경우를 위해 약간의 오류를 허용하는 방안을 제시하게 된다

## 선형 SVM




C 적게 허용 C많이 허용

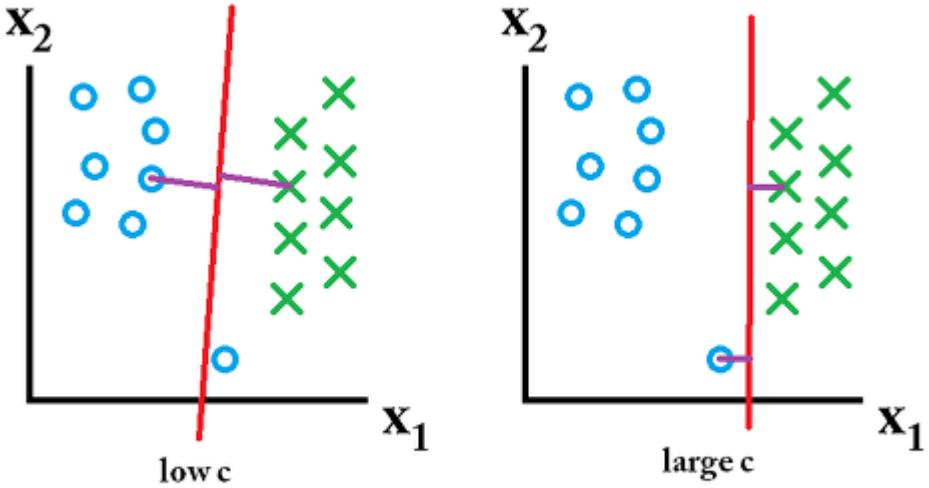
그림6. 매개변수 C의 영향

약간의 오류를 허용하는 것과 관련된 파라미터가 바로 cost(C)이다.

C는 얼마나 많은 데이터 샘플이 다른 클래스에 놓이는 것을 허용하는지를 결정한다.

# 선형 SVM

## 선형 SVM



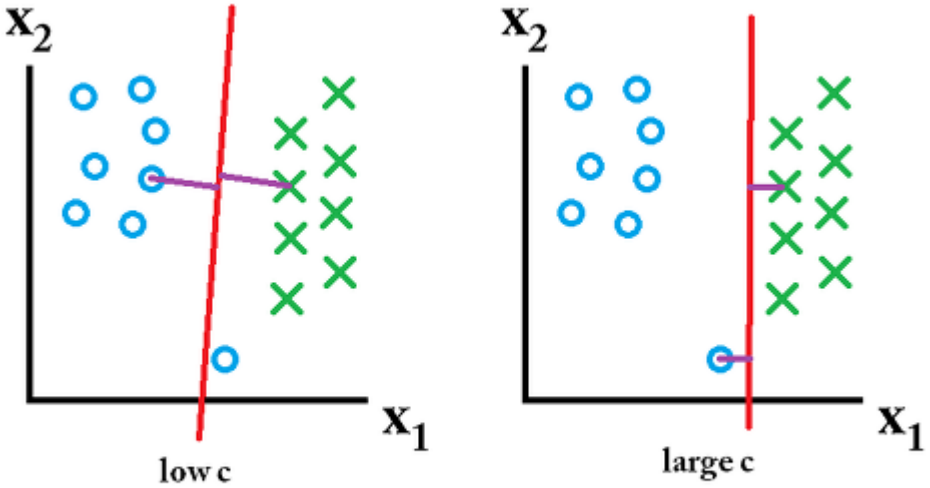

C 적게 허용 C많이 허용

그림6. 매개변수 C의 영향

C가 작을 수록 많이 허용하고, 클 수록 적게 허용한다. 다른 말로, C 값을 낮게 설정하면 이상치들이 있을 가능성을 크게 잡아 일반적인 결정 경계를 찾아내고, 높게 설정하면 반대로 이상치의 존재 가능성을 작게 봐서 좀 더 세심하게 결정 경계를 찾아낸다.

# 선형 SVM

## 선형 SVM




C 적게 허용 C많이 허용

그림6. 매개변수 C의 영향

C가 너무 낮으면 과소적합(underfitting)이 될 가능성이 커지고,

C가 너무 높으면 과대적합(overfitting)이 될 가능성이 커지니 적합한 C값을 찾아내는 것이 중요

C의 유무에 따라 하드마진(hard-margin) SVM, 소프트마진(soft-margin) SVM이라고 불린다.

# 선형 SVM

## 선형 SVM

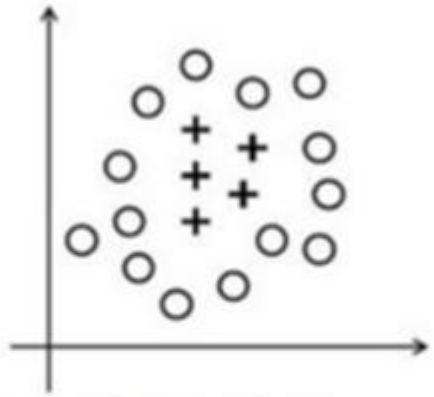


그림7. 선형 SVM으로는 제대로 분류할 수 없는 상황

그러나 선형 SVM으로는 데이터를 제대로 분류할 수 없는 상황들이 상당히 많다

왼쪽의 그림은 선형 SVM으로 데이터를 제대로 분류할 수 없다

그런 경우를 해결하기 위해 여러 방법이 제안되었는데 그 중 가장 널리 활용되고 있는 것은 바로 RBF 커널 SVM이다.

*커널의 종류는 더 있지만 RBF 커널만 소개합니다

# Non Linear SVM

# RBF Kernel SVM
# (radial basis function 가우시안 커널)

## RBF 커널 SVM

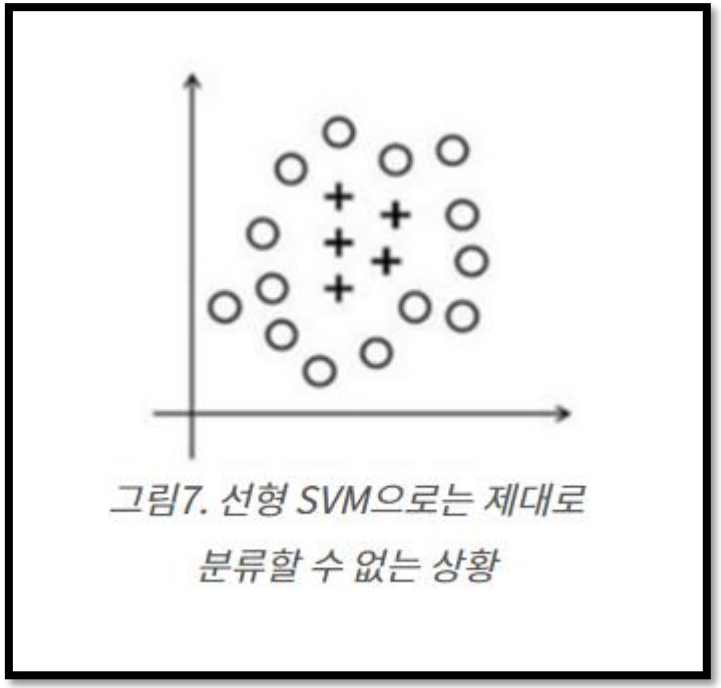


그림7. 선형 SVM으로는 제대로 분류할 수 없는 상황

SVM은 선형 SVM이든 RBF 커널 SVM이든 항상 선형으로 데이터를 분류하는 것이 기본적인 전략이다

위의 그림은 비선형으로 분류하지 않으면 분류될 수 없는 문제이다

그렇다면 이 경우는 어떻게 선형으로 분류할 수 있을까?

# RBF 커널 SVM

## RBF 커널 SVM

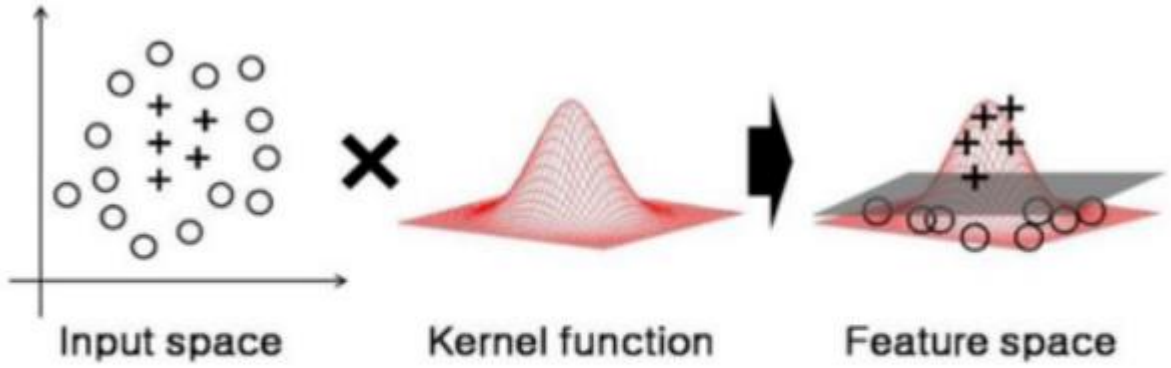



*그림8-1. 커널 기법에 대한 설명. 커널 함수를 적용하고 나니 선형적으로 분류할 수 있는 결정 경계를 찾을 수 있게 되었다.*

그래서 나온 것이 **커널 기법**이다. 커널 기법은 **주어진 데이터를 고차원 특징 공간으로 사상(변환)**해주는 것이다.

고차원 공간에 사상되고 나면 원래의 차원에서는 보이지 않던 선형으로 분류해줄 수 있는 방법이 보인다

# RBF 커널 SVM

## RBF 커널 SVM

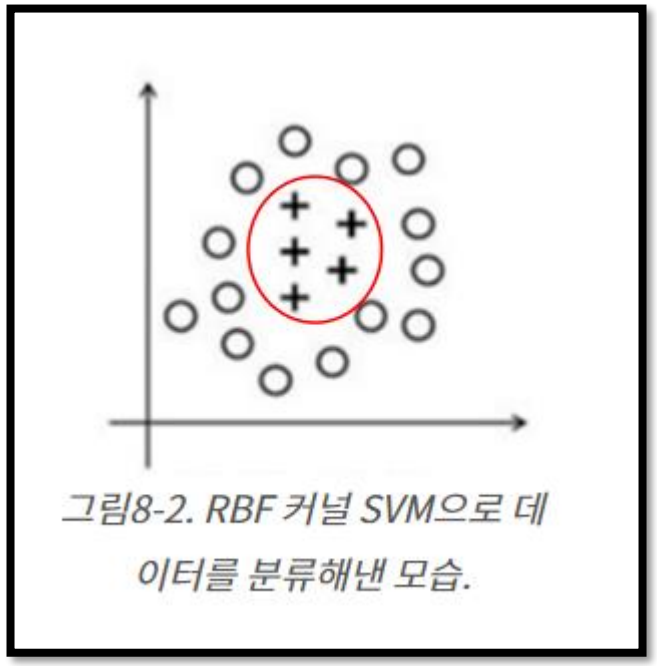
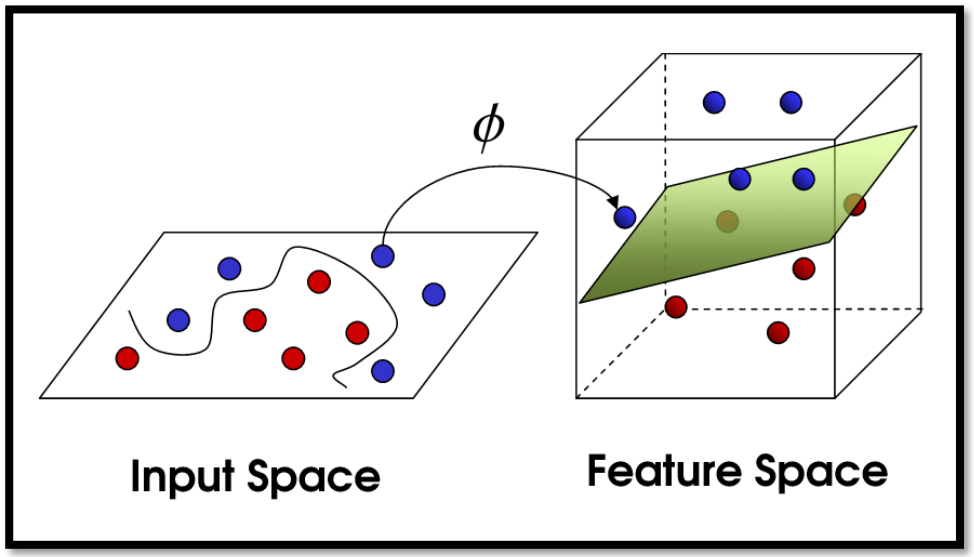


그림8-2. RBF 커널 SVM으로 데이터를 분류해낸 모습.




2차원 공간에서는 도저히 선형적으로 분류할 수 없을 것 같았던 데이터 분포가 커널 기법을 사용해서

3차원 공간으로 사상되니 분류가 가능해졌다.

3차원 공간에서 분류된 것을 다시 2차원 공간으로 매핑해서 보면 아래 그림과 같이 결정 경계가 둥그렇게 보일 것이다

# RBF 커널 SVM

## RBF 커널 SVM

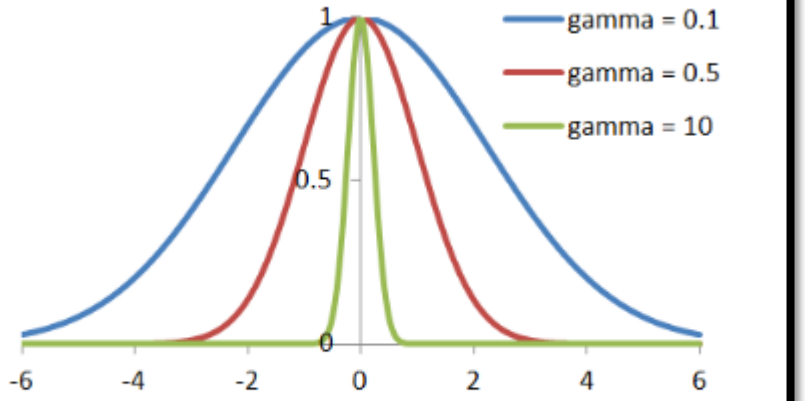



*그림8. gamma의 크기에 따른 가우시안 함수의 모양 [7]*

커널에는 Polynomial 커널, Sigmoid 커널, 가우시안 RBF 커널 등 종류가 많은데, 그 중 가장 성능이 좋아 자주 사용되는 것이 가우시안 RBF 커널이다. 각 커널마다 최적화를 도와주는 매개변수들이 따로 있다. RBF 커널의 경우 gamma라는 매개변수를 사용자가 조정 해야한다. SVM의 기본 매개변수인 C도 있으므로 총 2개의 매개변수를 설정해줘야한다.

# RBF 커널 SVM

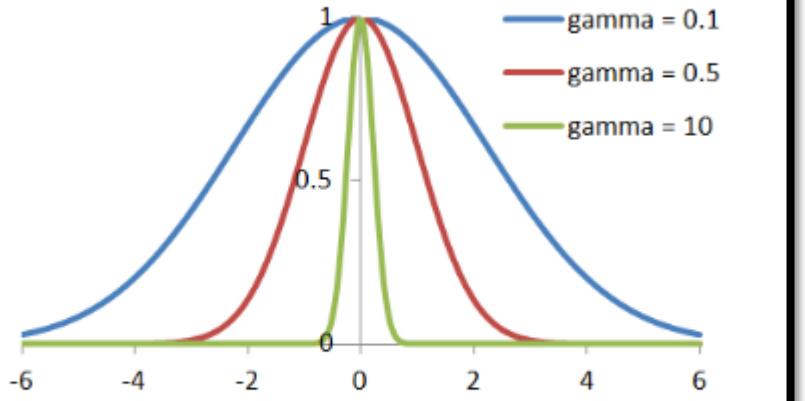



*그림8. gamma의 크기에 따른 가우시안 함수의 모양 [7]*

그렇다면 gamma의 역할은 무엇일까? gamma는 하나의 데이터 샘플이 영향력을 행사하는 거리를 결정한다. gamma는 가우시안 함수의 표준편차와 관련되어 있는데, 클수록 작은 표준편차를 갖는다. 즉, gamma가 클수록 한 데이터 포인터들이 영향력을 행사하는 거리가 짧아지는 반면, gamma가 낮을수록 커진다.

# RBF 커널 SVM

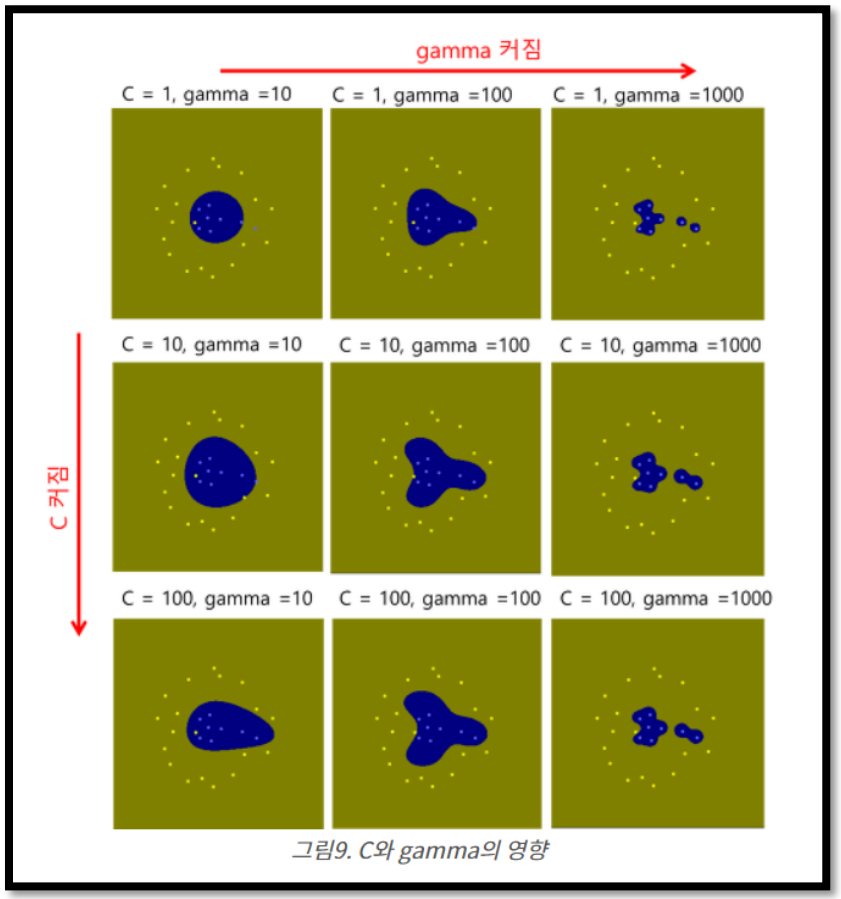



그림9. C와 gamma의 영향

우선 C부터 살펴보면, C가 커질수록 이상치의 존재 가능성을 낮게 본다고 이전에 설명했다. gamma가 10으로 동일한 첫번째 열을 보자. C=1인 상황에서는 두 개의 이상치를 인정하고 무난하게 결정 경계를 찾은 반면, C=100일 때는 하나의 이상치만 인정하면서 조금은 억지스럽게 분류해냈다.

# RBF 커널 SVM

## RBF 커널 SVM

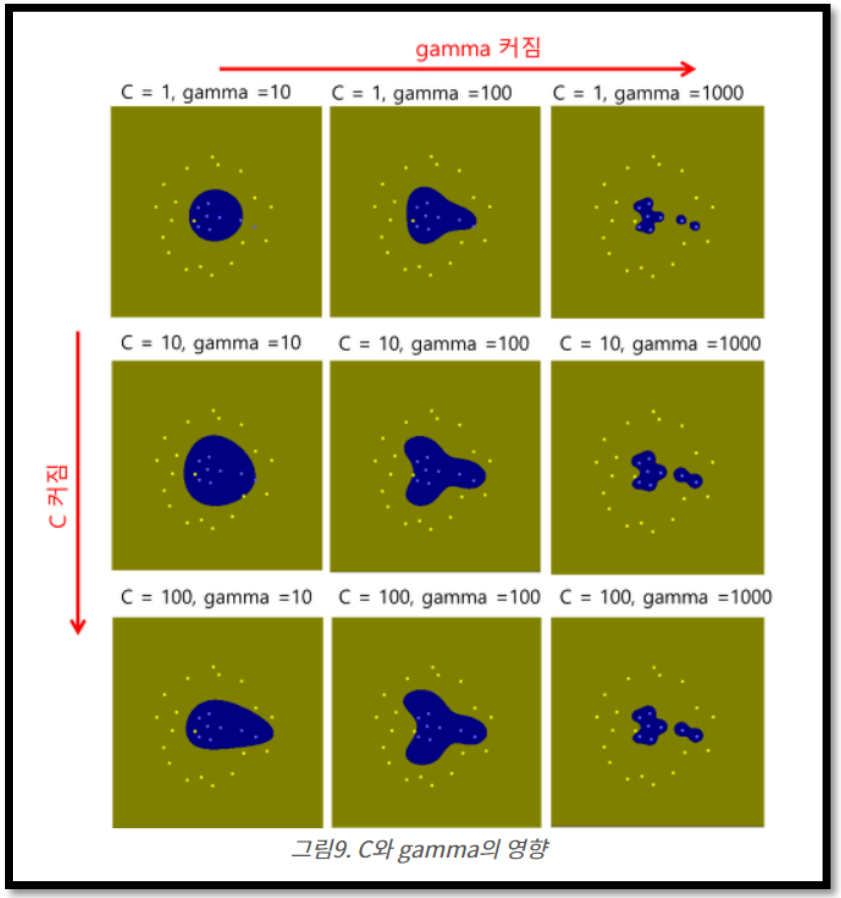


그림9. C와 gamma의 영향

이번에는 gamma의 영향을 살펴보자. C가 1로 동일하게 세팅되어 있는 첫 번째 행을 왼쪽에서부터 오른쪽으로 보면 gamma가 점점 커지는데, 결정 경계가 결정 경계 가까이에 있는 데이터 샘플들에 영향을 크게 받기 때문에 점점 더 구불구불해지는 것을 알 수 있다. 즉, gamma 매개변수는 결정 경계의 곡률을 조정한다고 말할 수도 있다. gamma의 값이 높아짐에 따라 파란색의 공간이 점점 작아졌는데, 위에서 언급한 것과 같이 각각의 데이터 포인터가 영향력을 행사하는 거리가 짧아졌기 때문이다.

# RBF 커널 SVM

## RBF 커널 SVM

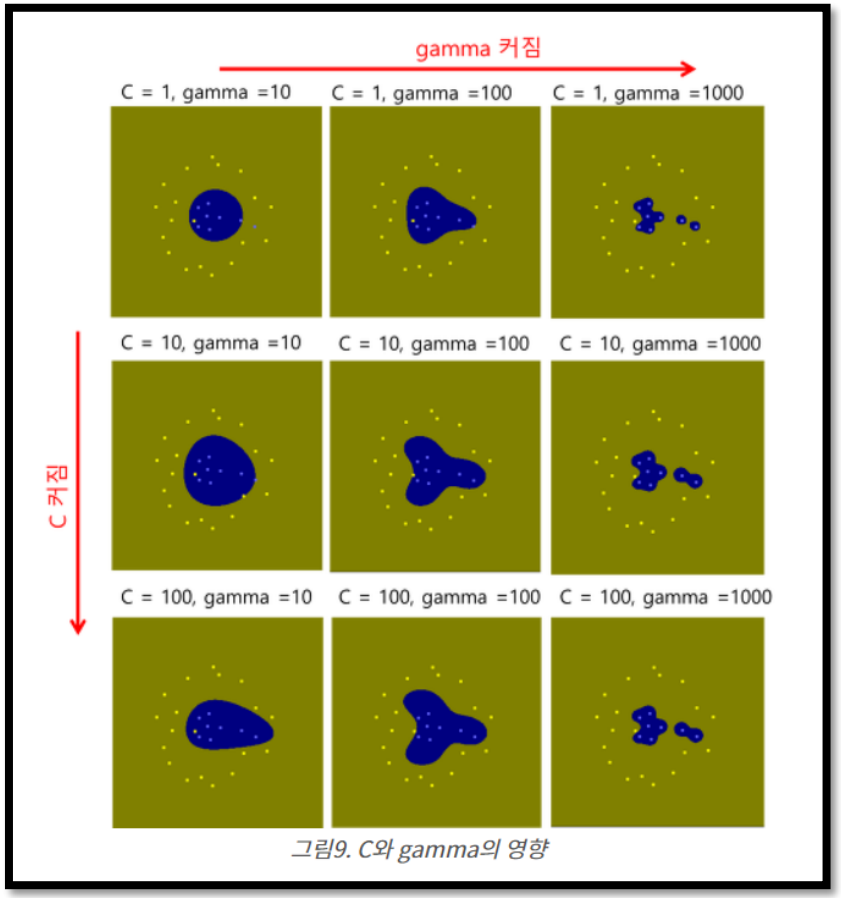


그림9. C와 gamma의 영향

매개변수 C와 마찬가지로 너무 낮으면 과소적합될 가능성이 크고, 너무 높으면 과대적합의 위험이 있다. 따라서 **두 파라미터 모두 적정값을 찾아야하는 것**이 우리 사용자들의 숙제이다.

# RBF Kernel SVM 심화
# (radial basis function 가우시안 커널)

# RBF 커널 SVM

SVM 결정경계 최적화를 위한 loss fuction

이거를 min해야하는 것만 찍어두시면 됩니다

슬랙 변수(slack variable)  Cost함수에 세타랑 x곱한 것 적용 한 함수  세타

$$\min_{\theta} C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$

실제 y값

SVM으로 잘 분류하기 위해서는

이 함수를 최적화(minimize) 해야한다!

# RBF 커널 SVM

SVM에서 제일 중요한 kernel!
왜 중요할까요?

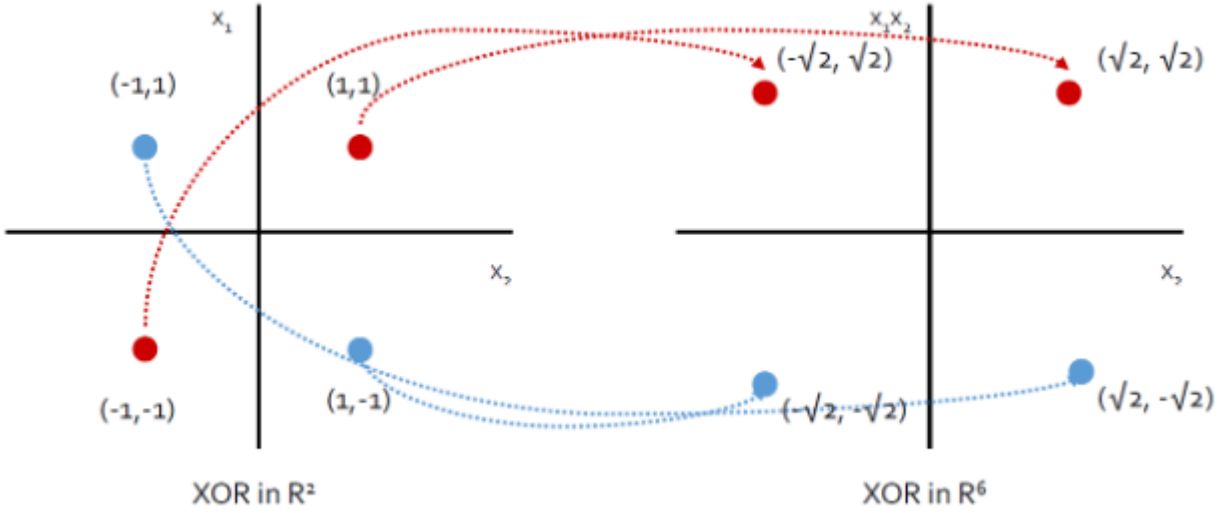
Kernel이 필요한 이유가 뭘까요?

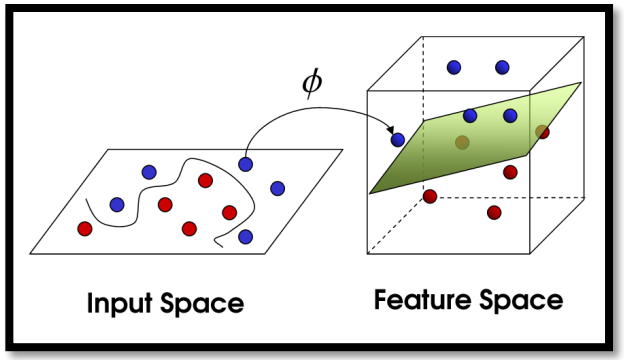
# RBF 커널 SVM

## Kernel이 필요한 이유

$$\Phi : (x_1, x_2) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, 1)$$

Input Space와 Featuer Space 사이를 매핑해주는 함수 $\Phi$
이 경우 2차원에서 6차원으로(고차원으로) 매핑(변환)




고차원으로 변환해주면
비선형문제 선형으로 분리 가능!

일단 비선형문제를 해결하려면
입력된 차원보다
고차원으로 공간을 변환해야하는 것을
기억합시다!

# RBF 커널 SVM

## Kernel이 필요한 이유

Soft-Margin SVM의 라그랑지안 Dual 식

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

이 계산에서 중요한 계산은 요거
(다른건 정해져 있음)

기억해야 하는 것은 svm 결정경계 함수 최적화를 위해 이 식을 계산 해야 한다!

$$\max L_D(\alpha_i) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \Phi(x_i)^T \Phi(x_j)$$

고차원으로 변환하는 순간
이걸 계산하는 걸로 변하는데
연산량이 기하급수적으로 늘어남
ex)2차원 내적 -> 6차원 내적)

# RBF 커널 SVM

## Kernel이 필요한 이유

가우시안 커널만 한번 살펴 봅시다

$$\max L_D(\alpha_i) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \Phi(x_i)^T \Phi(x_j)$$

깨알같은 이 식은 한번 더 보기

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

가우시안 커널은 이렇게 생겼는데,

따라서 가우시안 커널은 Input Space가 몇 차원이 됐든 무한대 차원의 Feature Space로 매핑한다는 얘기

요정도 계산만으로 말이죠

2시그마 제곱은 1이라 치고 위에껏만 봐봅시다

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

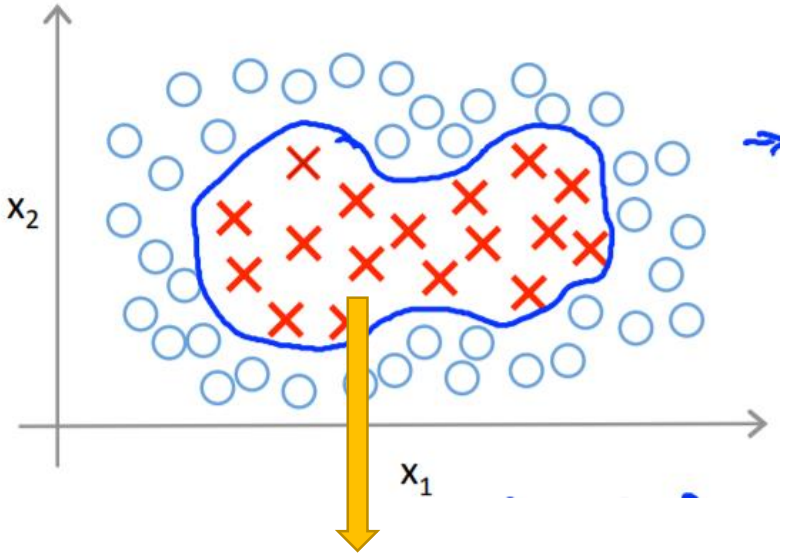
$$exp(2x_1 x_2) = \sum_{k=0}^{\infty} \frac{2^k x_1^k x_2^k}{k!}$$

테일러 정리에 의해 이렇게 된다고 함

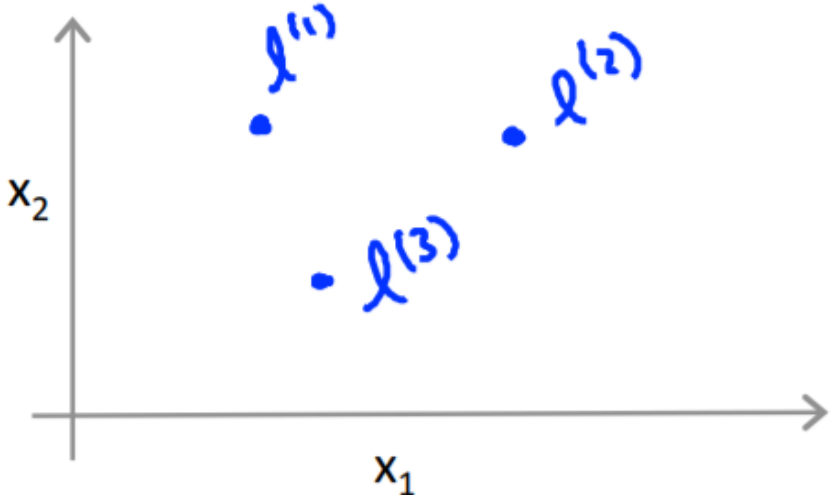
# RBF 커널 SVM

## Kernel이 필요한 이유

### 가우시안 커널 뭐하러 쓰죠




이곳으로 분류되는 곳을, y=1이라고 치고
여기 경계(파란선)안에 있는 데이터에 landmark를
찍어봅시다

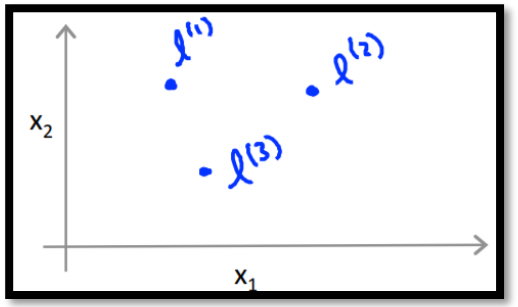



Landmark 3개 찍은 것

# RBF 커널 SVM

## Kernel이 필요한 이유

가우시안 커널 뭐하러 쓰죠




분모는 새로운 데이터 x와
Landmark1 과의 유클리디안거리 즉, 유사도

$$f_1 = similarity(x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$$
$$f_2 = similarity(x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(2)}\|^2}{2\sigma^2}\right)$$
$$f_3 = similarity(x, l^{(3)}) = \exp(\dots)$$

Kernel (Gaussian kernels)

$k(x, l^{(i)})$

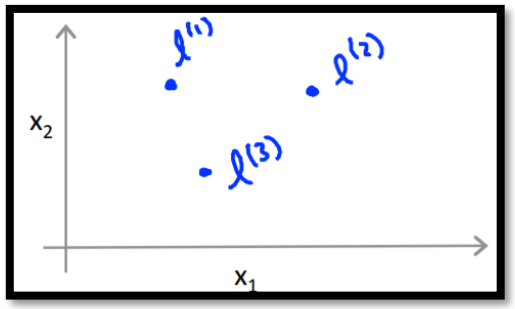
새로운 데이터 x와 모든 랜드마크와의
유사도(거리가 가까운지)를
가우시안 커널로 구한다

Cf ) 저희 분야에서는 유사도랑 거리랑 비슷한 개념이라고 하죠

# RBF 커널 SVM

## Kernel이 필요한 이유

가우시안 커널 뭐하러 쓰죠




$$f_1 = \text{similarity}(x, l^{(1)}) = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right) = \exp\left(-\frac{\sum_{j=1}^{n}(x_j - l_j^{(1)})^2}{2\sigma^2}\right)$$

판서라 보기 힘드셨죠
수식으로 제대로 쓰면 이렇게 된다고 합니다
새로운 데이터 x와 모든 landmark와의 유사도를 sum해서
2시그마로 나누고 exp해주면 f1이 됩니다

$\exp(x) = e^x$

# RBF 커널 SVM

## Kernel이 필요한 이유

두가지만 살펴봅시다 첫번째는 데이터가 랜드마크랑 거리가 가까우면???

랜드마크랑 데이터가 가까우면 f1은 1이 된다(1에 근접)

If $x \approx l^{(1)}$ :

$$f_1 \approx \exp\left(-\frac{0^2}{2\sigma^2}\right) \approx 1$$

데이터가 근접하면 e의 지수가
거의 0이 되면서 e^0 = 1이 되겠죠

랜드마크랑 데이터가 멀면 f1은 0이 된다(0에 근접)

If $x$ if far from $l^{(1)}$ :

$$f_1 = \exp\left(-\frac{(\text{large number})^2}{2\sigma^2}\right) \approx 0$$

밑에 것도
데이터가 멀면 e^엄청큰숫자의 역수
이러니까 0에 가까워지겠죠

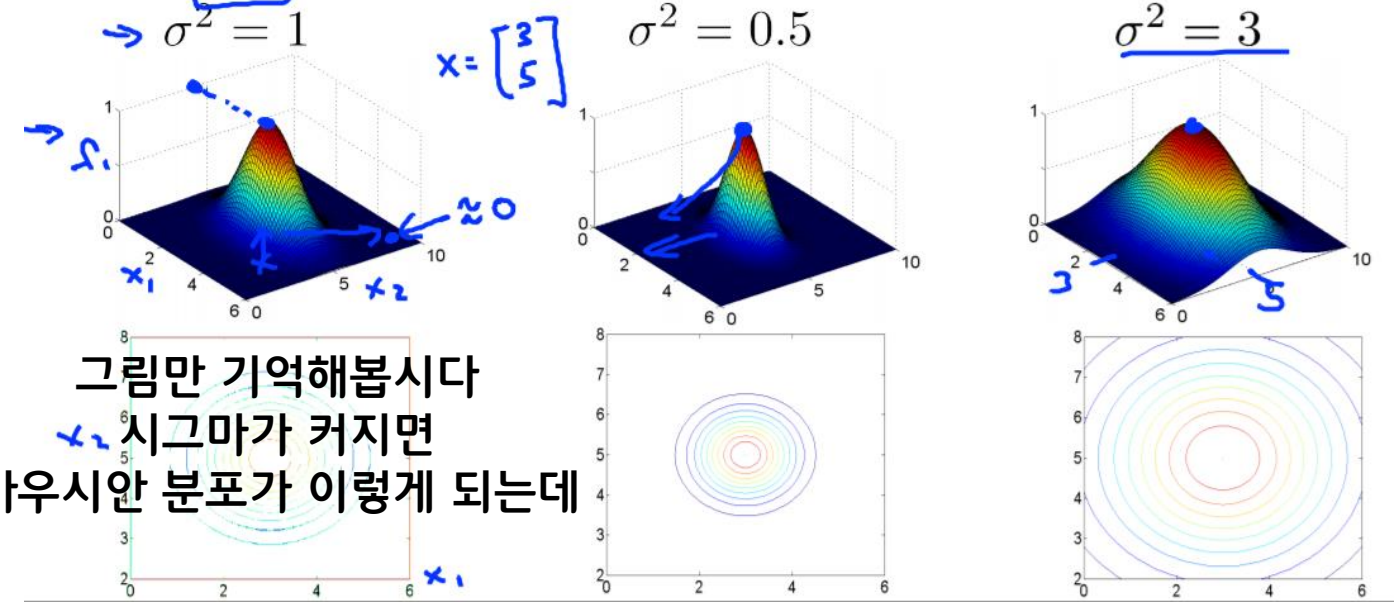
# RBF 커널 SVM

## Kernel이 필요한 이유

**두가지만 살펴봅시다 두번째는 시그마가 커지면???**

$l^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$, $f_1 = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$ **요놈이 시그마**

$\sigma^2 = 1$ $\sigma^2 = 0.5$ $\sigma^2 = 3$



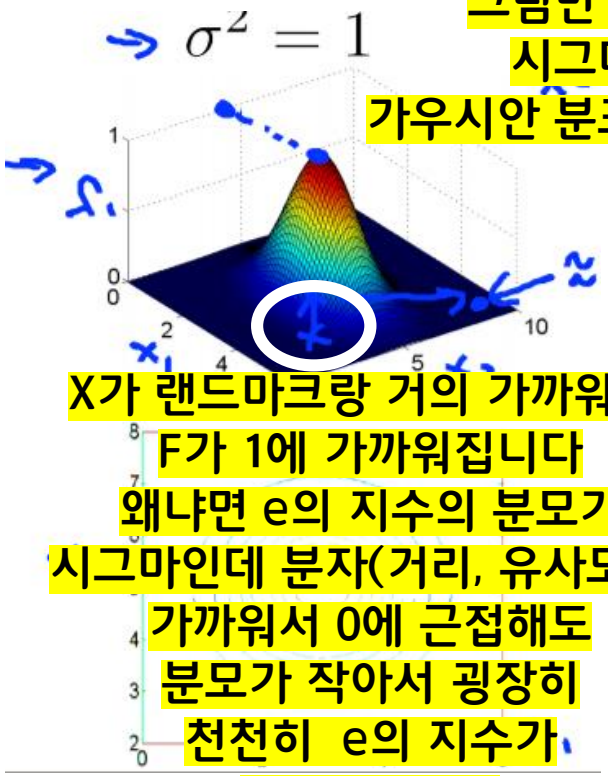
**그림만 기억해봅시다**
**시그마가 커지면**
**가우시안 분포가 이렇게 되는데**

# RBF 커널 SVM

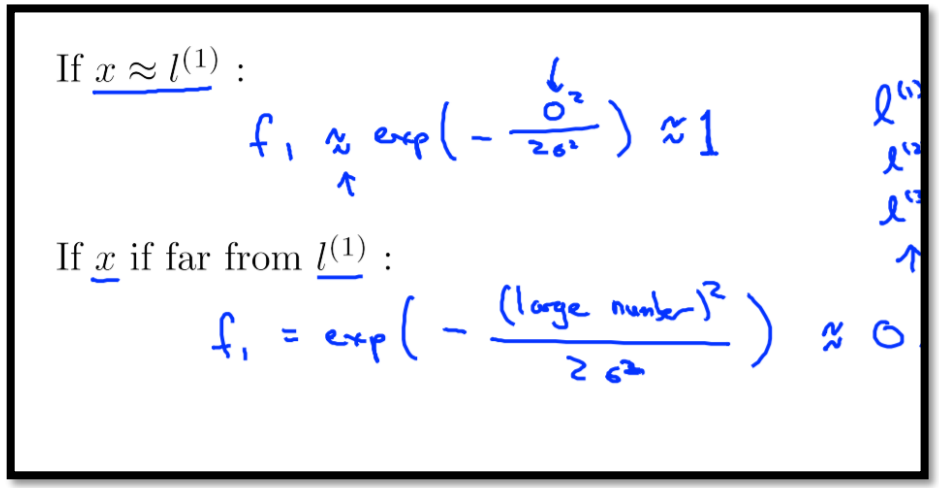
## Kernel이 필요한 이유

두가지만 살펴봅시다 두번째는 시그마가 커지면???

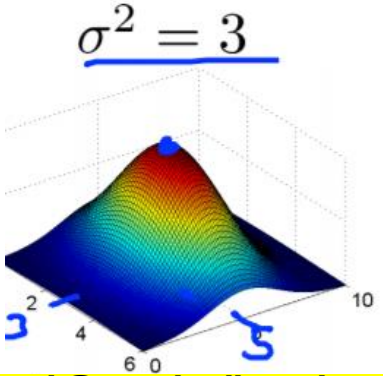
그림만 기억해봅시다
시그마가 커지면
가우시안 분포가 이렇게 되는데




X가 랜드마크랑 거의 가까워야!
F가 1에 가까워집니다
왜냐면 e의 지수의 분모가
시그마인데 분자(거리, 유사도)가
가까워서 0에 근접해도
분모가 작아서 굉장히
천천히 e의 지수가
작아지거든요




이 식이랑 천천히 대조해보면서 생각해보면 댐




이 넘은 x가 랜드마크랑
조금만 가까워도
F가 1이 쉽게 되요
그냥 분포가 퍼져있으니까
좀만 가까워도 넌 나랑 같음!
이렇게 치부해버리는거죠

# RBF 커널 SVM

RBF 커널 SVM

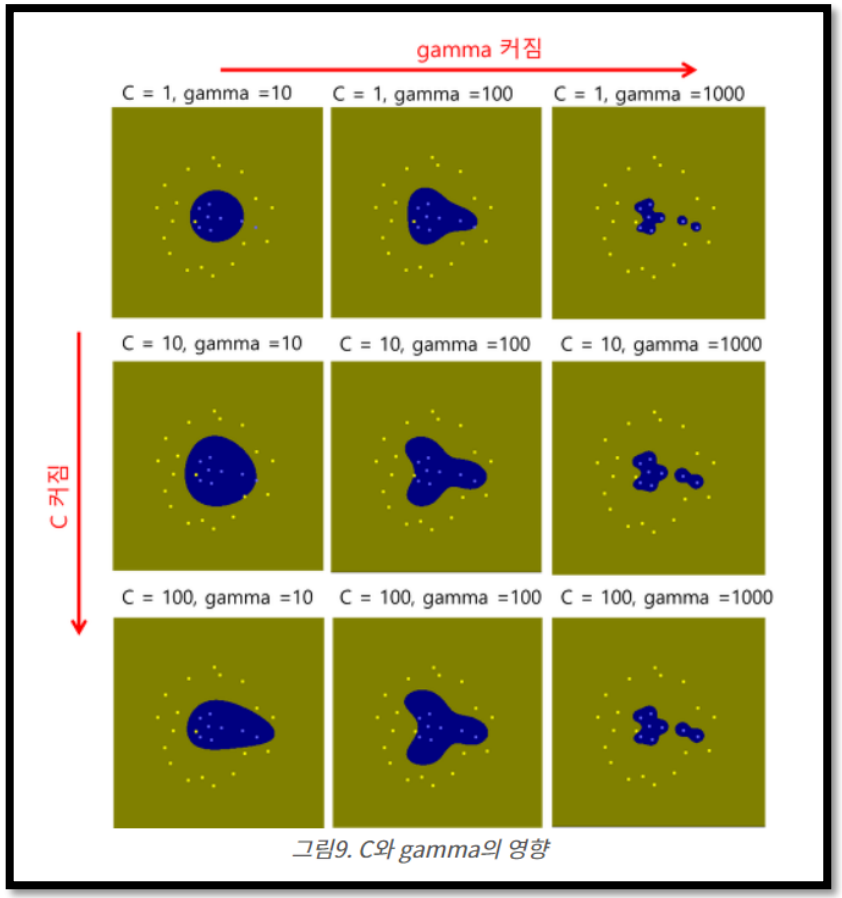


그림9. C와 gamma의 영향

시그마란 놈은 나중에
하이퍼 파라미터 감마라는 놈인데요

감마가 커질수록 -→그림에서 이쪽으로 갈 수록
경계가 구불구불 해지는 것을
볼 수 있어요

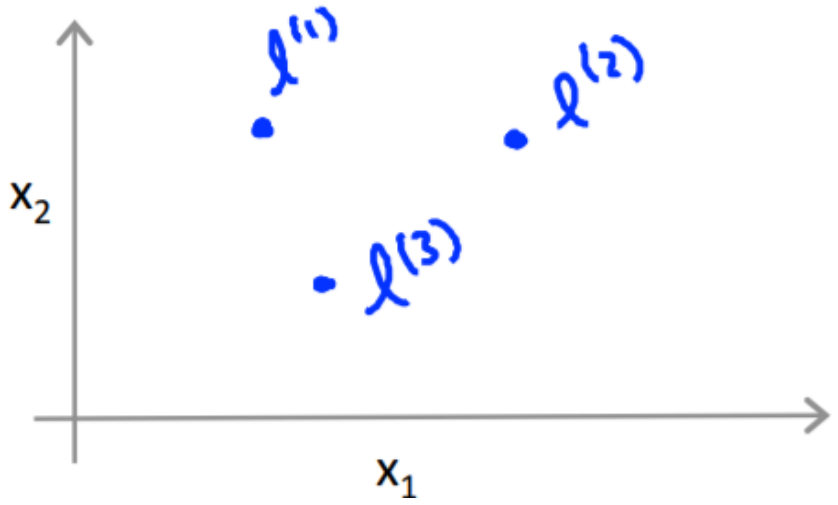
이것은 거리가 좀만 가까워도 넌 나랑
같은놈이야 이래서,
세세한 지역까지 자신의 구역으로 만들어버리거든요

그래서 감마가 높으면 과적합 가능성도
높아진다고 합니다!

# RBF 커널 SVM

## Kernel이 필요한 이유

**가우시안 커널 뭐하러 쓰는지 이제 아시겠나요**




요기선 Landmark 3개만 찍었지만
어디가 landmark인지 첨엔 모르잖아요?
그래서 처음에는 모든 데이터를 landmark로 찍어봅니다!

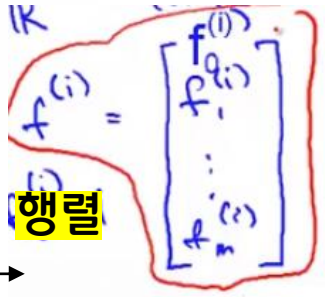
**데이터와 레이블이 주어지면**

Given $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$,
choose $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \ldots, l^{(m)} = x^{(m)}$

**랜드마크를 데이터 개수만큼 찍어보아서**

$$f_1 = \text{similarity}(x, l^{(1)})$$
$$f_2 = \text{similarity}(x, l^{(2)})$$
$$\ldots$$

**데이터 개수만큼 랜드마크를 찍고**
**함수 다 구해야 하는데**



**행렬**

***그래서 데이터 너무 많으면 연산량 쫌 많아짐**

# RBF 커널 SVM

## Kernel이 필요한 이유

판서 수식 죄송합니다…사람마다 쓰는 기호가 조금씩 달라서 이 방법밖엔..

랜드마크 찍은다음 함수구해서 결론적으로 요 함수 minimize해야되요

$$\min_{\theta} C \sum_{i=1}^{m} y^{(i)} cost_1(\theta^T f^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^{m} \theta_j^2$$

아까랑 세타랑 x말고
세타랑 f랑 곱하는거 달라졌음

아까랑 달라진건 요기가 n이 아니라 m이 됨
데이터 갯수만큼 랜드마크랑 유사도 계산해야 됨

함수 최적화는 아까 라그랑주 뭐시기가 해줌

$$\min_{\theta} C \sum_{i=1}^{m} \left[ y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$

전에 보여준 것

$$f_1 = similarity(x, l^{(1)})$$
$$f_2 = similarity(x, l^{(2)})$$
$$\dots$$

# RBF 커널 SVM

## 결론

**왜 그런지는 각자 논리적으로 생각해보기로 해요**

C가 크면

Large'C: 'Lower'bias, 'high'variance

지금 주어진 데이터는  잘 적합(잘 맞춤)는데 이 데이터에만 잘 맞는 모델이 될 수도 있다는 뜻

C가 작으면

Small'C: 'Higher'bias, 'low'variance.'

모델 자체의 정확도는 떨어질 수 있는데 좀 더 일반화 된 모델이 될 가능성은 높다는 뜻

# RBF 커널 SVM

## 결론

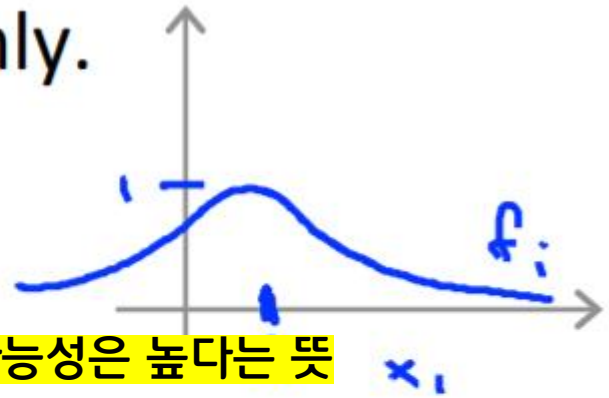
**왜 그런지는 각자 논리적으로 생각해보기로 해요**

시그마가 크면

Large $\sigma^2$: Features $f_i$ vary more smoothly.

→ Higher bias, lower variance.

$\exp\left(-\frac{\|x-l^{(i)}\|^2}{2\sigma^2}\right)$




모델 자체의 정확도는 떨어질 수 있는데 좀 더 일반화 된 모델이 될 가능성은 높다는 뜻

시그마가 작으면

Small $\sigma^2$: Features $f_i$ vary less smoothly.

Lower bias, higher variance.


지금 주어진 데이터는  잘 적합(잘 맞춤)는데 이 데이터에만 잘 맞는 모델이 될 수도 있다는 뜻

## RBF 커널 SVM

**결론**

**왜 그런지는 각자 논리적으로 생각해보기로 해요**

Scaling
(피쳐끼리 단위가 다를 수 있어서 표준화 해주는 것)
해줘야 함

$$\frac{X - \text{평균}}{\text{표준편차}}$$

*가우시안 커널은 유클리드 거리가 들어가서 특히…

# RBF 커널 SVM

## 결론

갓 응(앤드류 응 : Andrew Ng) 교수님께서는

피쳐가 데이터 개수에 비해 많으면 커널 없는 SVM 즉, linear kernel을 쓰라고(피쳐 개수 10000개 정도)

피쳐가 적고 데이터 개수는 보통이면 가우시안 커널을 적용한 SVM 즉, RBF 커널 SVM을 쓰라고 하셨고(피쳐갯수 1000개정도, 데이터 개수 10 -10000개 정도)

피쳐가 적고 데이터가 많으면 커널없는 SVM이나 로지스틱 회귀를 쓰라고 하셨으나, 이 3번째 부분은 커널 있는 SVM은 데이터가 많아지면 연산이 많아져 부담이 되기 때문이라고 하셨다 따라서 연산의 부담 좀 많아도 괜찮으면 RBF 커널도 무방할 듯 하다

*지극히 경험에 의한 개인적인 생각이시다

# RBF 커널 SVM

하이퍼 파라미터 두개나 있는데 어떻게 해야 잘될지 감이 안와요 ㅠㅠ

C, gamma

C 커짐

Gamma 커

| | | | | |
|---|---|---|---|---|
| $2^{-5}, 2^{-15}$ | $2^{-3}, 2^{-15}$ | | $2^{11}, 2^{-15}$ | $2^{13}, 2^{-15}$ |
| $2^{-5}$ | | | $2^{11}, 2^{-13}$ | $2^{13}, 2^{-13}$ |
| $2^{-5}, 2^{-11}$ | $2^{-3}, 2^{-11}$ | | $2^{11}, 2^{-11}$ | $2^{13}, 2^{-11}$ |
| | | | | $2^{13}, 2^{-9}$ |
| $2^{-5}, 2^{-7}$ | $2^{-3}, 2^{-7}$ | ... | $2^{11}, 2^{-7}$ | $2^{13}, 2^{-7}$ |
| | | | | |
| $2^{-5}, 2^{-3}$ | $2^{-3}, 2^{-3}$ | | $2^{11}, 2^{-3}$ | $2^{13}, 2^{-3}$ |
| $2^{-5}, 2^{-1}$ | $2^{-3}, 2^{-1}$ | | $2^{11}, 2^{-1}$ | $2^{13}, 2^{-1}$ |
| $2^{-5}, 2^{1}$ | $2^{-3}, 2^{1}$ | | $2^{11}, 2^{1}$ | $2^{13}, 2^{1}$ |
| $2^{-5}, 2^{3}$ | $2^{-3}, 2^{3}$ | | $2^{11}, 2^{3}$ | $2^{13}, 2^{3}$ |

감마랑 C 바꿔보면서
어떤게 좋은지 실험해 보면 되는거죠

이건 그리드써치(Grid Search)라고 해서 이미 라이브러리로 구현되어있어요

*그림1. grid search의 예*

# RBF 커널 SVM

## 정리

1) SVM 알고리즘 중에서 가장 성능이 괜찮고 일반적으로 널리 사용되는 것은 RBF 커널 SVM이다.

2) 좋은 성능을 얻으려면 매개변수인 C와 gamma를 잘 조정해줘야 한다.

3) C는 데이터 샘플들이 다른 클래스에 놓이는 것을 허용하는 정도를 결정하고, gamma는 결정 경계의 곡률을 결정한다.

4) 두 값 모두 커질수록 알고리즘의 복잡도는 증가하고, 작아질수록 복잡도는 낮아진다.

5) 일반적으로 grid search로 경험적으로 최적의 매개변수 값들을 찾아가는데, 이상 설명한 내용을 어느 정도 숙지하고 있다면 훨씬 더 빠르게 좋은 성능을 내는 매개변수 값들을 찾아낼 수 있을 것이다.

# 실습

# 실습

데이터 불러오기

```
iris = datasets.load_iris()
X = iris['data']
y = (iris['target'] == 2).astype(np.str)

svm_clf = Pipeline([
    ('scaler', StandardScaler()),
    ('liear_svc', LinearSVC(C=1, loss='hinge', random_state=42))
])

svm_clf.fit(X, y)
```

X와 y 나누기

Pipeline만들어서
모델을 설정해 놓는다

모델에 fit하는 함수

# 실습

```
iris
```

```
5  (high!)₩n    ============== ==== ==== ======= ===== ====================₩n₩n    :Missing Attribute Values: None₩n
n: 33.3% for each of 3 classes.₩n    :Creator: R.A. Fisher₩n    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
988₩n₩nThe famous Iris database, first used by Sir R.A. Fisher. The dataset is taken₩nfrom Fisher₩'s paper. Note that i
R, but not as in the UCI₩nMachine Learning Repository, which has two wrong data points.₩n₩nThis is perhaps the best kno
und in the₩npattern recognition literature.  Fisher₩'s paper is a classic in the field and₩nis referenced frequently to
a & Hart, for example.)  The₩ndata set contains 3 classes of 50 instances each, where each class refers to a₩ntype of i
s is linearly separable from the other 2; the₩nlatter are NOT linearly separable from each other.₩n₩n.. topic:: Referen
R.A. "The use of multiple measurements in taxonomic problems"₩n    Annual Eugenics, 7, Part II, 179-188 (1936); also i
₩n    Mathematical Statistics" (John Wiley, NY, 1950).₩n  - Duda, R.O., & Hart, P.E. (1973) Pattern Classification an
   (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.₩n  - Dasarathy, B.V. (1980) "Nosing Around the N
ystem₩n    Structure and Classification Rule for Recognition in Partially Exposed₩n    Environments".  IEEE Transacti
sis and Machine₩n    Intelligence, Vol. PAMI-2, No. 1, 67-71.₩n  - Gates, G.W. (1972) "The Reduced Nearest Neighbor R
ions₩n    on Information Theory, May 1972, 431-433.₩n  - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AU
ceptual clustering system finds 3 classes in the data.₩n  - Many, many more ...',
'feature_names': ['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)'],
```

**유명한 iris 데이터인데 꽃잎 넓이나 길이 같은걸로 품종 구분하는 아주 단순한 데이터다**

# 실습

5-fold 함수로 구현

```
def fivefold(x,y,model):
    model_scores = np.zeros(5)
    scores = np.zeros(5)

    cv = KFold(5, shuffle=True, random_state=0)
    for i, (idx_train, idx_test) in enumerate(cv.split(x)):
        #데이터 하나씩 트레인과 테스트에 투과
        x_train = x[idx_train]
        x_test = x[idx_test]
        y_train = y[idx_train]
        y_test = y[idx_test]

        #모델 fit
        result = model.fit(x_train,y_train)

        #prediction
        model_pred = result.predict(x_train)
        pred = result.predict(x_test)
        model_rmse = accuracy_score(y_train, model_pred)
        rmse = accuracy_score(y_test, pred)

        #test 5번 하나하나 프린트
        model_scores[i] = model_rmse
        scores[i] = rmse
        print("학습 accuacy = {:.8f}, 검증 accuacy = {:.8f}".format(model_rmse, rmse))
    print('평균 train accuacy : ' + str(model_scores.mean()) )
    print('평균 test accuacy : ' + str(scores.mean()) )
```

모델 정확도랑 테스트 정확도 넣을 변수 공간 만들기

5개 테스트 데이터 인덱스로 나누기

트레인과 테스트 인덱싱을 5번 다르게 해야한다
(위에 for문이 5번 돌아감)

모델 fit

모델에 train데이터와 test데이터 예측(predict)해서 정확도 구함

5번의 정확도와
최종 평균 print하는 코드

# 실습

```
scaler = StandardScaler()
svm_clf1 = LinearSVC(C=1, loss="hinge", random_state=42)
svm_clf2 = LinearSVC(C=100, loss="hinge", random_state=42)
svm_clf3 = LinearSVC(C=0.1, loss="hinge", random_state=42)

scaled_svm_clf1 = Pipeline([
        ("scaler", scaler),
        ("linear_svc", svm_clf1),
    ])
scaled_svm_clf2 = Pipeline([
        ("scaler", scaler),
        ("linear_svc", svm_clf2),
    ])
scaled_svm_clf3 = Pipeline([
        ("scaler", scaler),
        ("linear_svc", svm_clf3),
    ])


scaled_svm_clf1.fit(X, y)
scaled_svm_clf2.fit(X, y)
scaled_svm_clf3.fit(X, y)
```

Linear svm 파이프라인 3개 만들어 놓음
C가 0.1, 1, 100일 때 구분

# 실습

```
#c=0.1
fivefold(X,y,scaled_svm_clf3)
```

```
학습 R2 = 0.93333333, 검증 R2 = 0.93333333
학습 R2 = 0.95833333, 검증 R2 = 0.86666667
학습 R2 = 0.94166667, 검증 R2 = 0.96666667
학습 R2 = 0.95000000, 검증 R2 = 1.00000000
학습 R2 = 0.95000000, 검증 R2 = 0.93333333
평균 train rmse : 0.9466666666666667
평균 test rmse : 0.9400000000000001
```

```
#c= 1
fivefold(X,y,scaled_svm_clf1)
```

```
학습 R2 = 0.96666667, 검증 R2 = 1.00000000
학습 R2 = 0.99166667, 검증 R2 = 0.90000000
학습 R2 = 0.97500000, 검증 R2 = 1.00000000
학습 R2 = 0.96666667, 검증 R2 = 1.00000000
학습 R2 = 0.98333333, 검증 R2 = 0.90000000
평균 train rmse : 0.9766666666666668
평균 test rmse : 0.96
```

```
#c=100
fivefold(X,y,scaled_svm_clf2)
```

```
학습 R2 = 0.97500000, 검증 R2 = 1.00000000
학습 R2 = 1.00000000, 검증 R2 = 0.93333333
학습 R2 = 0.97500000, 검증 R2 = 1.00000000
학습 R2 = 0.98333333, 검증 R2 = 0.96666667
학습 R2 = 0.99166667, 검증 R2 = 0.90000000
평균 train rmse : 0.9850000000000001
평균 test rmse : 0.9600000000000002
```

**아까 만든 함수에 넣음**

# 실습

```
from sklearn.svm import SVC

rbf_kernel_svm_clf = Pipeline([
        ("scaler", StandardScaler()),
        ("svm_clf", SVC(kernel="rbf", gamma=5, C=0.001))
    ])

rbf_kernel_svm_clf.fit(X, y)
```

```
Pipeline(memory=None,
     steps=[('scaler', StandardScaler(copy=True, with_mean=True, wi
e, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=5, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True
  tol=0.001, verbose=False))])
```

```
fivefold(X,y,rbf_kernel_svm_clf)
```

```
학습 R2 = 0.63333333, 검증 R2 = 0.80000000
학습 R2 = 0.70833333, 검증 R2 = 0.50000000
학습 R2 = 0.66666667, 검증 R2 = 0.66666667
학습 R2 = 0.66666667, 검증 R2 = 0.66666667
학습 R2 = 0.65833333, 검증 R2 = 0.70000000
평균 train rmse : 0.6666666666666666
평균 test rmse : 0.6666666666666666
```

```
X.shape
```

```
(150, 4)
```

**똑같은 코드를 RBF 커널로 적용**

# Reference

1. 앤드류 응(Andrew Ng) 교수님의 Machine Learning (명)강의

https://www.coursera.org/learn/machine-learning#syllabus

#강추, #SVM의 살아있는 직관

2. Ratsgo's Blog

https://ratsgo.github.io/machine%20learning/2017/05/23/SVM/

https://ratsgo.github.io/machine%20learning/2017/05/29/SVM2/

https://ratsgo.github.io/machine%20learning/2017/05/30/SVM3/

#SVM 부수적인 자료로 훌륭함

3. 충북대학교 이건명 교수님 강의, 카이스트 오혜연 교수님 강의

http://www.kocw.net/home/search/kemView.do?kemId=1170523

www.kmooc.kr/courses/course-v1:KAISTk+KCS470+2017_K0203/about

#SVM 함수 정확한 작동원리가 '굳이' 궁금하다면 보는 것을 추천드립니다


들어주셔서 감사합니다.