SVM

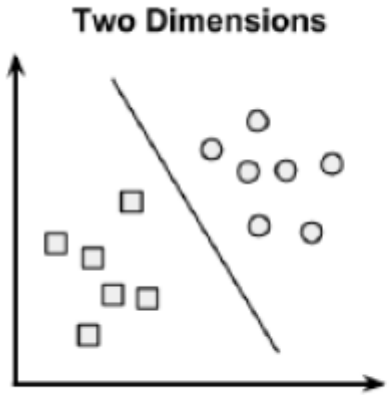
B조 심지원 발표

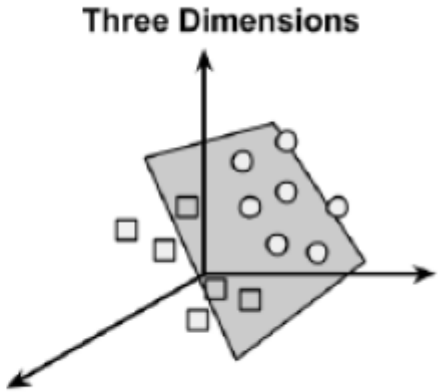
SVM(Support Vector Machine)이란?

- 속성 값에 따라 다차원 공간의 예제를 초평면(hyperplane)을 경계로 분류
- 지도학습기법으로 비-중첩(non-overlapping) 분할을 제공하며 모든 속성(attributes)을 활용하는 전역적(global) 분류 모형
- 최근접 이웃학습과 선형회귀 모델링의 두 측면을 결합한 모델링
- => 주로 분류Classification이나 회귀Regression 분석 모델로 사용한다.

SVM(Support Vector Machine)이란?

- SVM은 유사한 데이터를 분류하기 위해 초평면(Hyperplane)이라는 선형 경계를 사용한다.
- (소프트 마진/커널 트릭을 사용하여 비선형 문제로 확장이 가능하다)



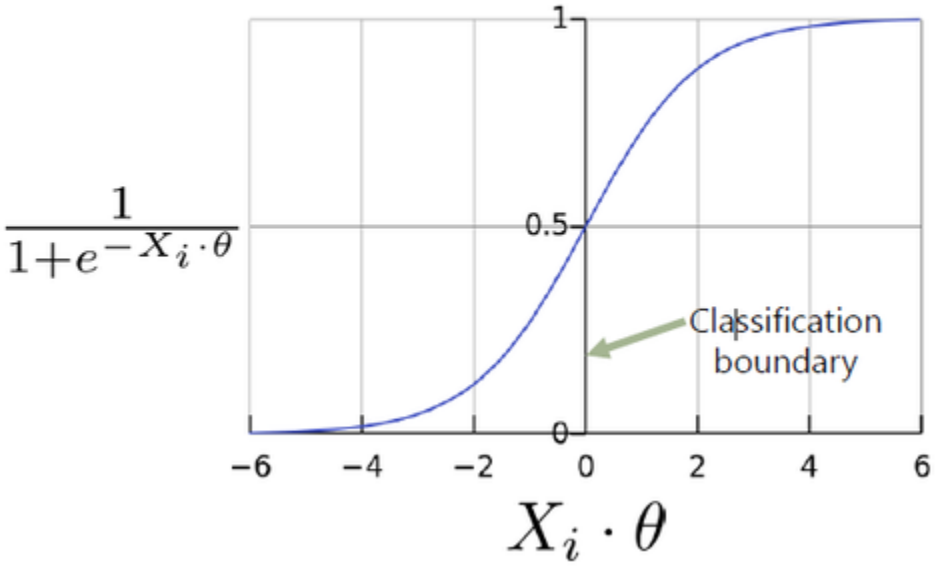


SVM의 이해

- Linear Regression은 정확한 예측을 위한 확률을 기반으로 분류

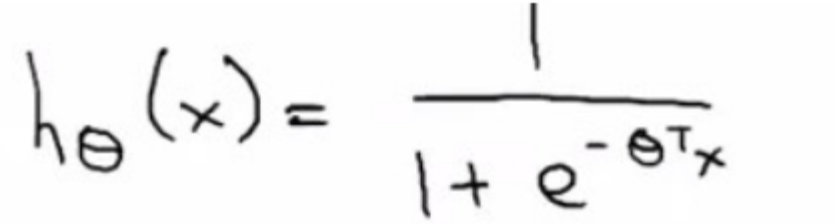
$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

sigmoid function: $\sigma(t) = \frac{1}{1+e^{-t}}$



SVM의 이해

- Linear Regression은 정확한 예측을 위한 확률을 기반으로 분류

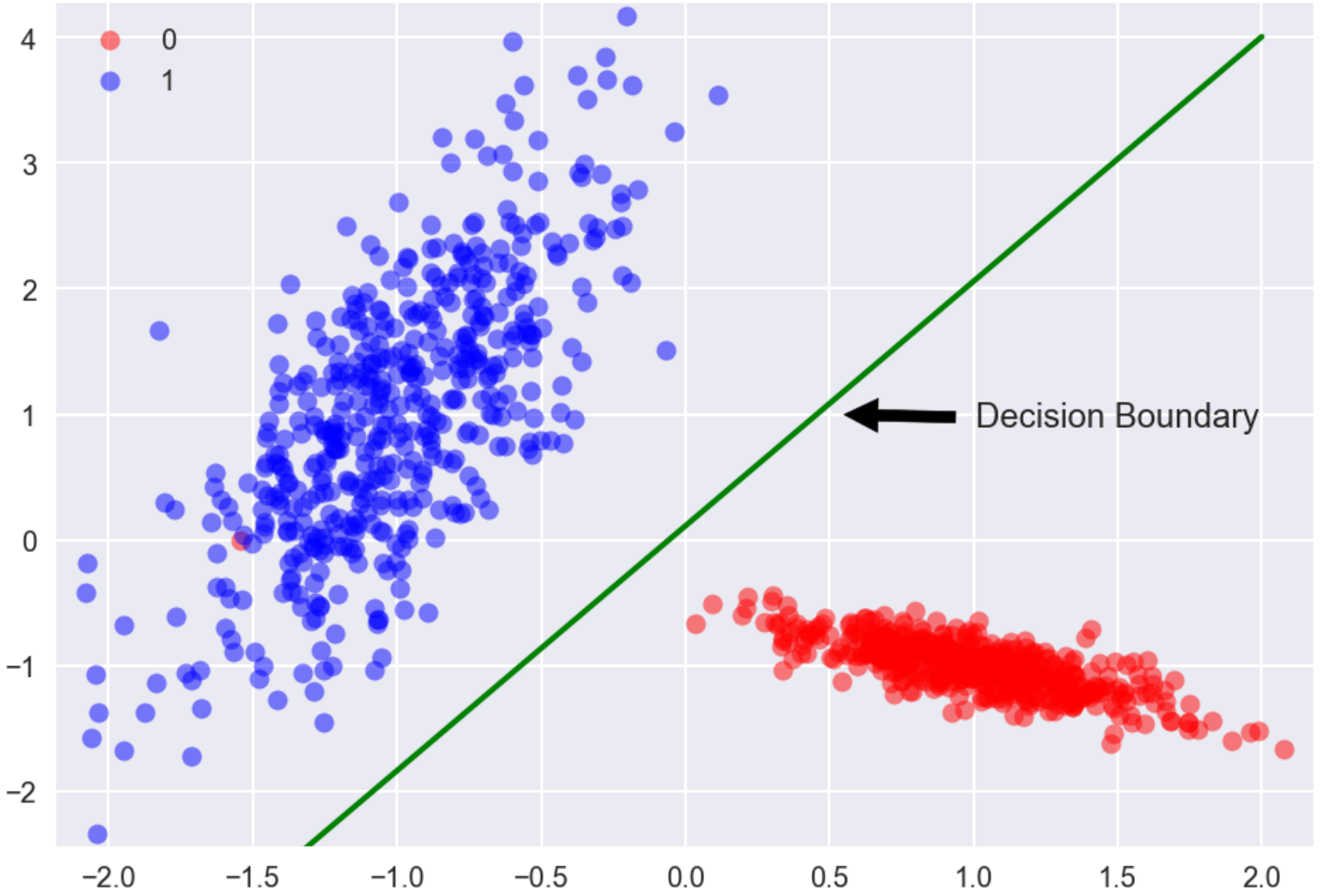


- H(x)는 정답의 확률을 의미. 이 확률을 입력값으로 손실값을 계산하고, 최소화하면서 분류를 진행한다.
- 학습 데이터에서 손실값을 최소화하는 관점으로 접근한다.
  => **학습에 최적화되어 테스트 데이터를 잘 분류하지 못한다.**

SVM의 이해

- **SVM은 Margin을 최대화하는 Decision boundary를 제공한다.**

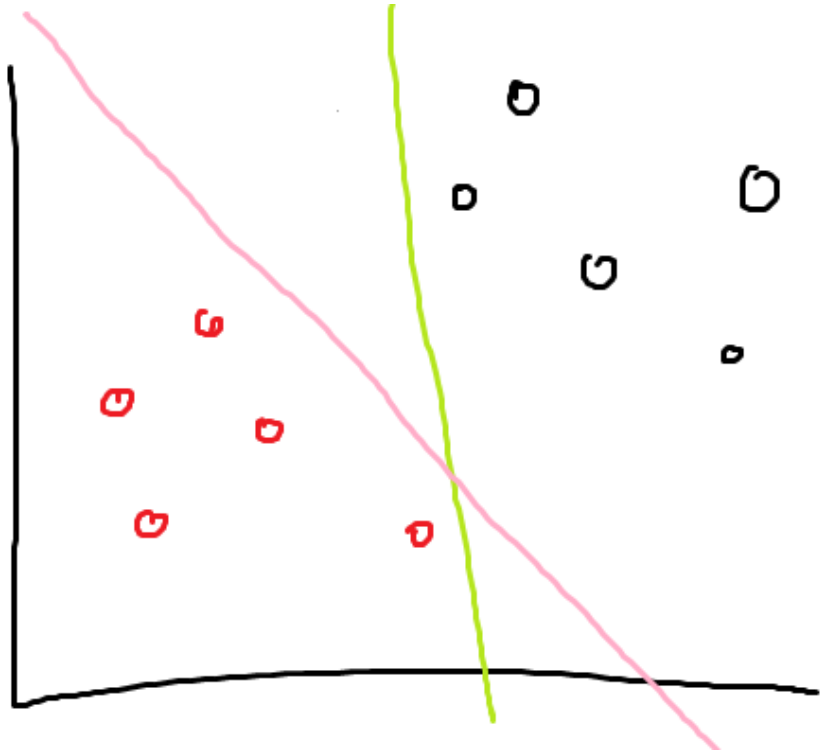
Decision boundary란?



- Decision boundary는 $\boldsymbol{W^T}$***X+b=0**의 수식에 따르는 선형을 의미한다. 그리고 가중치 벡터(W)는 Decision boundary와 직교.
- 왜 직교해야 할까?
  편의상 b=0이라고 할 때,
  W^T *X=0가 Decision boundary라고 할 수 있다.
  2개의 벡터 내적의 결과가 0이 되는 각도는 90도이므로 직교한다고 할 수 있다.

Margin을 최대로 하는 Decision Boundary

- SVM의 Decision boundary는 가중치 벡터(w)에 직교하면서, margin이 최대가 되는 선형을 찾는 것

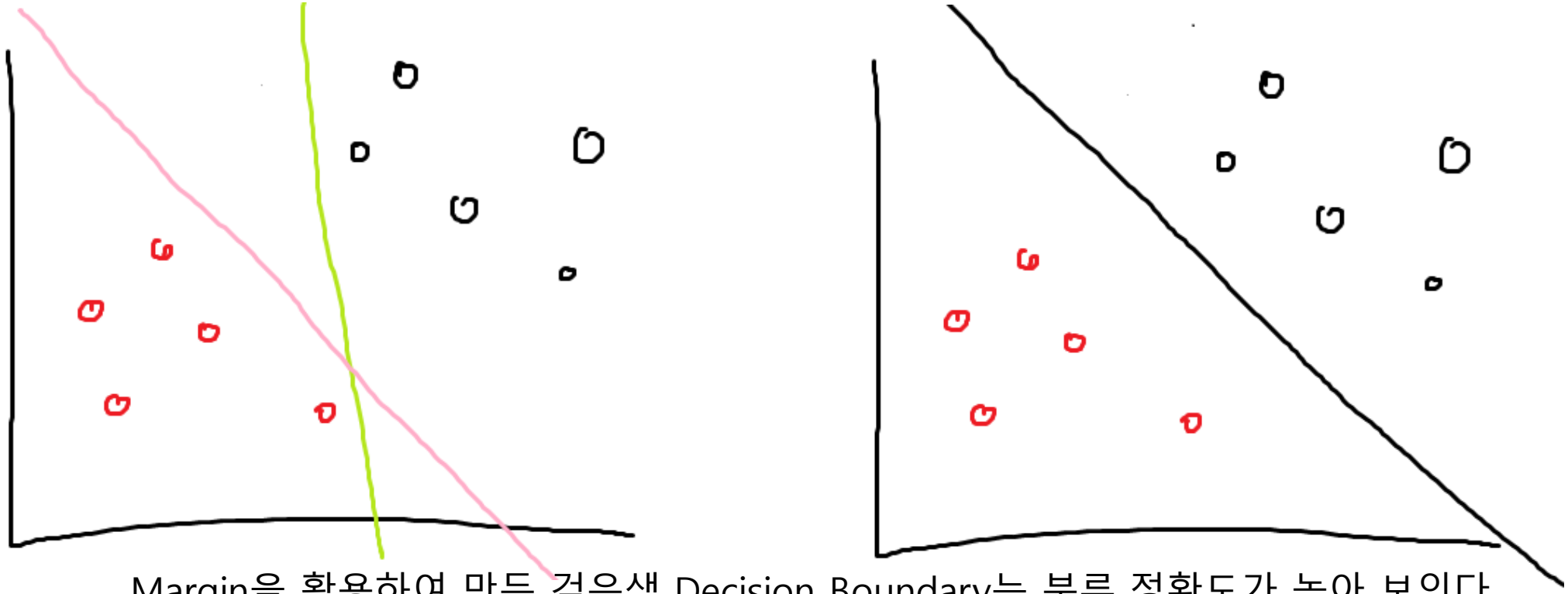


Margin이 없는 경우
데이터를 분류할 때, 연두색 또는 분홍색의 Decision Boundary가 될 수도 있다.

이는 직관적으로 봤을 때도
Decision boundary가 추가되는 데이터를 제대로 분류하지 못할 가능성이 커 보인다.

Margin을 최대로 하는 Decision Boundary

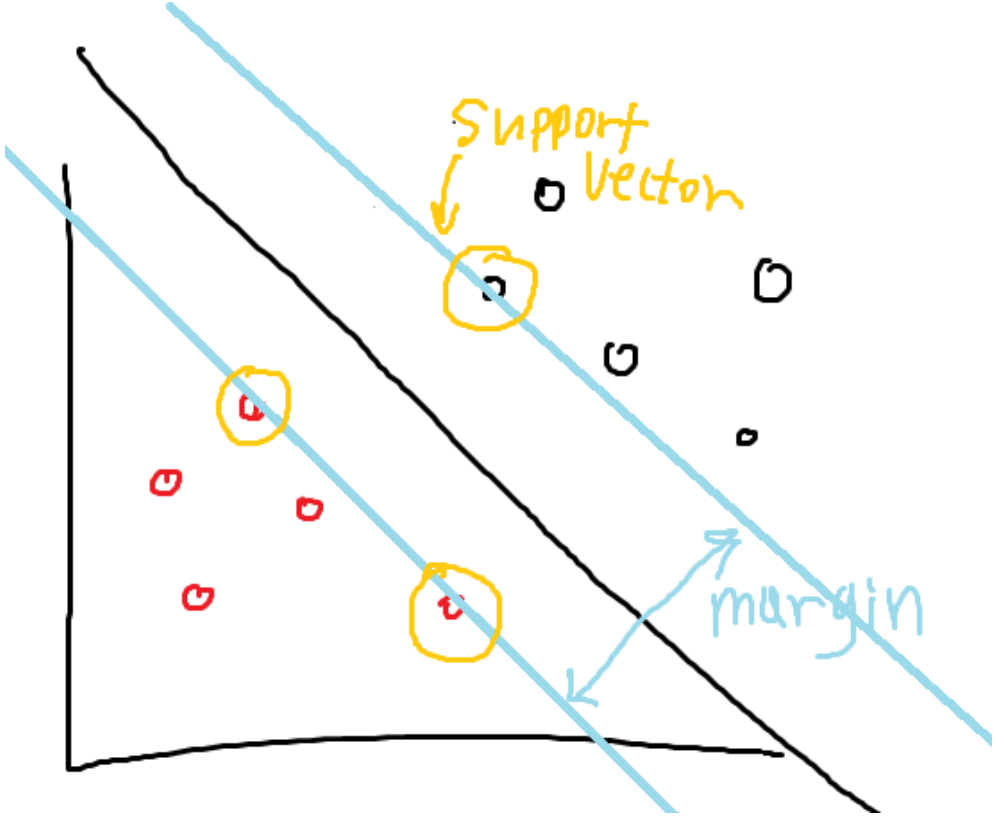
- SVM의 Decision boundary는 가중치 벡터(w)에 직교하면서, margin이 최대가 되는 선형을 찾는 것



Margin을 활용하여 만든 검은색 Decision Boundary는 분류 정확도가 높아 보인다.

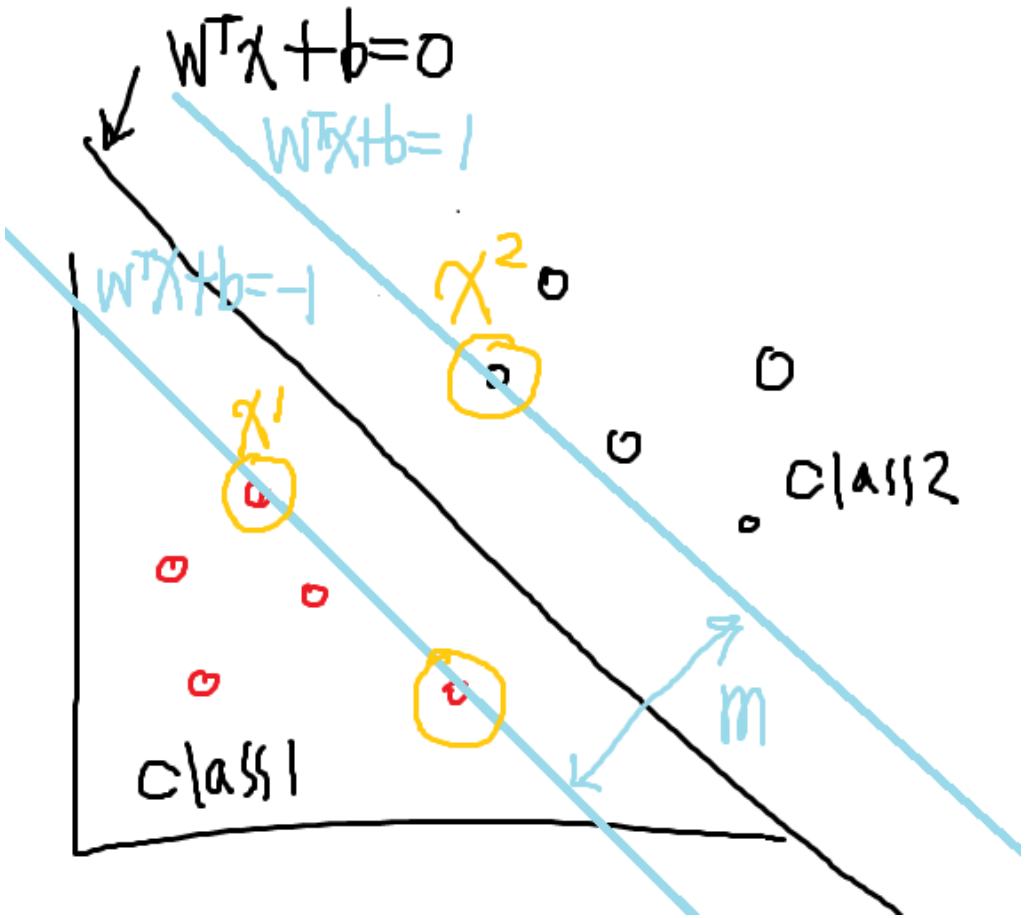
Margin을 최대로 하는 Decision Boundary

- Margin이 커지면 학습 데이터에 최적화되지 않고, 실제 데이터의 분류 정확도를 향상시킨다.



Decision boundary와 평행하고,
가중치 벡터(w)와 직교하며,
Decision boundary와
가장 가까운 좌표와의 거리가
최대가 되는(margin)
3개의 벡터(support vector)를 기준으로
Decision boundary를 결정한다.

Margin 계산법



각 클래스에서 decision boundary와 가장 가까운 포인트를 x1(class1), x2(class2)라고 가정,
편의상 b=0으로 가정,

W^T *X1 = -1

W^T *X2 = 1

M = W^T *X1 - W^T *X2

(W^T *X = 0과 X1, X2 간의 거리로 계산)

= |1| / ||W|| + |-1| / ||W||

= **|2| / ||W||**

Margin 최대화

- Margin의 최대화를 위해 먼저 SVM이 어떻게 모델을 최적화하는지 알아보자.

- SVM의 비용함수를 구하기 위해서는 Logistic regression에 대한 이해가 필요하다.

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

If $y = 1$, we want $h_\theta(x) \approx 1$, $\theta^T x \gg 0$

If $y = 0$, we want $h_\theta(x) \approx 0$, $\theta^T x \ll 0$

Margin 최대화

- Logistic Regression의 비용함수

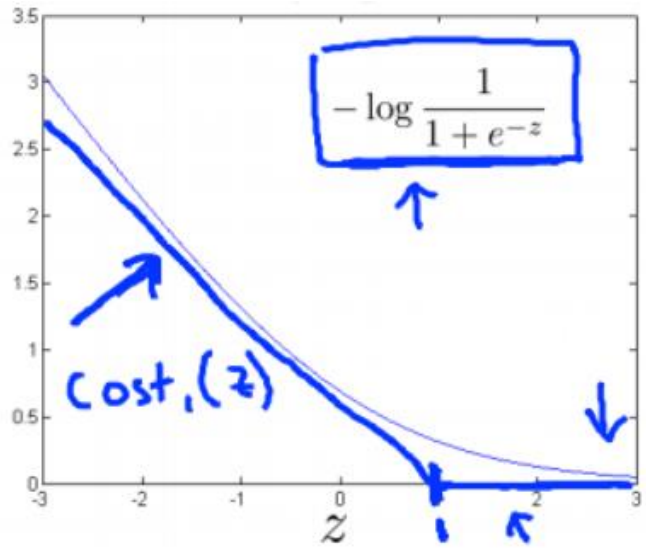
$$-(y \log h_\theta(x) + (1-y) \log(1 - h_\theta(x)))$$

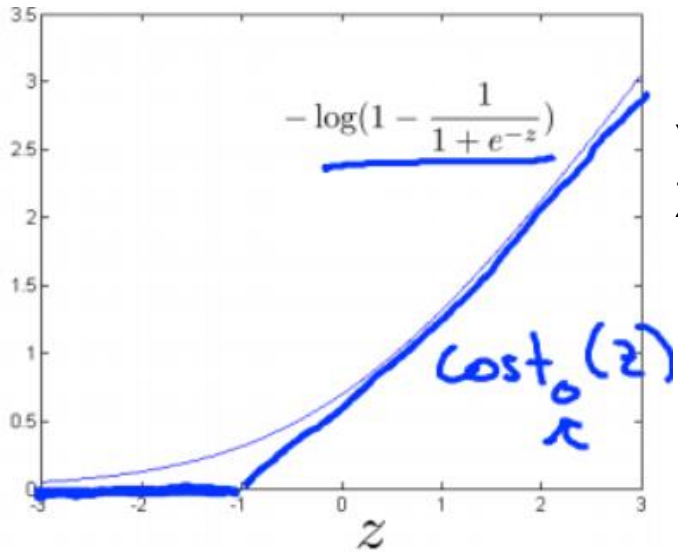
- 위 식에 h(x)를 대입

$$-y \log \frac{1}{1+e^{-\theta^T x}} - (1-y) \log \left(1 - \frac{1}{1+e^{-\theta^T x}}\right)$$

Y=1일때
(Θ^T*X(이하 z) >> 0)

선형적으로 바꾼 비용함수는 나중에 SVM 비용함수 최적화 때 성능향상을 가져다준다.





Y=0일때
Z << 0

Margin 최대화

- Logistic Regression의 비용함수

$$-(y \log h_\theta(x) + (1-y) \log(1 - h_\theta(x)))$$

- 위 식에 h(x)를 대입

$$-y \log \frac{1}{1+e^{-\theta^T x}} - (1-y) \log \left(1 - \frac{1}{1+e^{-\theta^T x}}\right)$$

정답의 유형에 따라 Cost가 증가하는 방향이 다르다.

Logistic Regression의 Log 기반의 cost함수(cross-entropy)와 다르게 SVM은 Hinge loss 함수를 사용한다.

SVM uses "hinge" loss $\max(0, 1 - y_i f(\mathbf{x}_i))$

Margin 최대화

$$\text{cost}_0(z) = \max(0, k(1+z))$$

$$\text{cost}_1(z) = \max(0, k(1-z))$$

$$\min_\theta \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\left(-\log h_\theta(x^{(i)})\right) + (1-y^{(i)})\left((-\log(1-h_\theta(x^{(i)}))\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Margin 최대화

$$\min_{\theta} \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\left(-\log h_\theta(x^{(i)})\right) + (1-y^{(i)})\left((-\log(1-h_\theta(x^{(i)}))\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

$$\min_{\theta} C\left[\sum_{i=1}^{m} y^{(i)}\underbrace{\left(-\log h_\theta(x^{(i)})\right)}_{\text{cost}_1(\theta^T x^{(i)})} + (1-y^{(i)})\underbrace{\left((-\log(1-h_\theta(x^{(i)}))\right)}_{\text{cost}_0(\theta^T x^{(i)})}\right] + \frac{\cancel{\lambda}}{\cancel{2m}}\sum_{j=1}^{n}\theta_j^2 \quad \left(C = \frac{1}{\lambda}\right)$$

Margin 최대화

$$\min_\theta \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\left(-\log h_\theta(x^{(i)})\right) + (1-y^{(i)})\left((-\log(1-h_\theta(x^{(i)}))\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

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

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

Margin 최대화

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

Margin 최대화

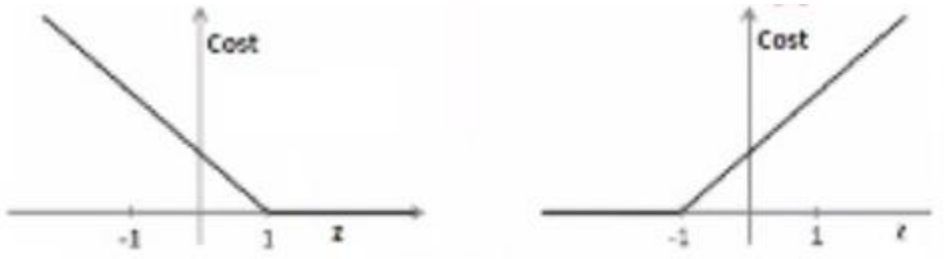
$$\min_{\theta} C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$

A B

- C가 엄청 크다고 가정
- CA+B가 최소화하기 위해서는 A=0

If y=1, we want $\Theta^T x \geq 1$ (not just ≥0)

If y=0, we want $\Theta^T x \leq -1$ (not just <0)



Margin 최대화

- CA=0이 성립되어야 함

$$\sum_{i=1}^{m} y^{(i)} \text{cost}_1(\Theta^T x) + (1 - y^{(i)}) \text{cost}_0(\Theta^T x) = 0$$

- 비용함수를 간략화할 수 있다. (C가 엄청 크다는 전제하에)

$$J(\theta) = C \cdot 0 + \frac{1}{2} \sum_{j=1}^{n} \Theta_j^2$$

$$= \frac{1}{2} \sum_{j=1}^{n} \Theta_j^2$$

참고) C가 미치는 영향

- C는 regularization 파라미터 λ와 관련되어 모델학습시 overfitting을 조정한다.

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

- C가 크면, cost($\theta^T x^{(i)}$)가 작아야 전체 cost가 최소화된다.
- C가 크다 = λ가 작다

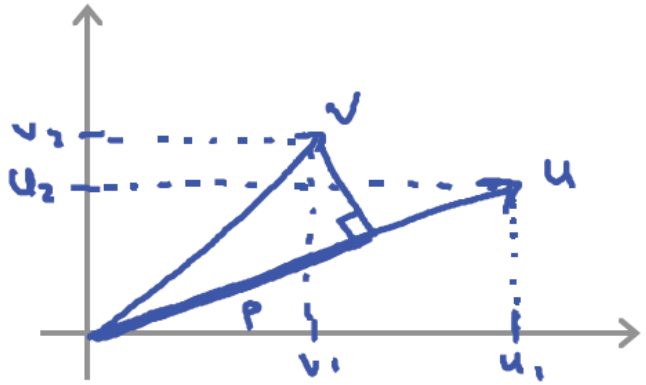
-> 가중치 Θ가 미치는 영향이 커진다.

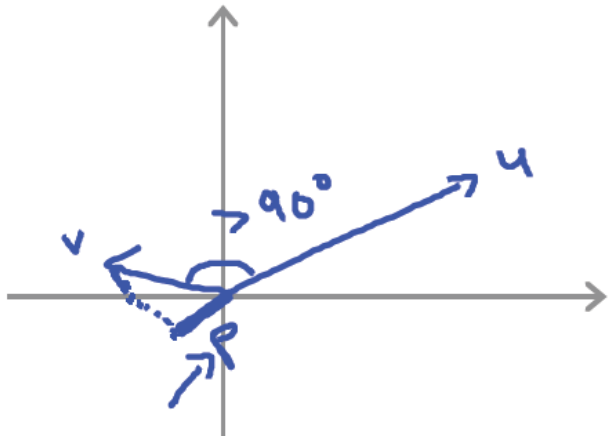
= margin이 좁아지는 경향이 있다.

Regularization에 영향이 적어,
모든 학습 데이터에 오분류가 없도록 분류하게 된다.

Margin 최대화

- 벡터의 내적 이해





- u=[u1, u2], v=[v1, v2]
- the inner products between the verctors u and v = u^T*v
- p = the length of the projection of the vector v onto u
- **u^T *v = p||u||**

Margin 최대화

$$\min_{\theta} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

$$\text{s.t.} \quad \theta^T x^{(i)} \geq 1 \qquad \text{if } y^{(i)} = 1$$

$$\theta^T x^{(i)} \leq -1 \qquad \text{if } y^{(i)} = 0$$

- 간략한 설명을 위해 Θ0 =0, n=2
- ½(Θ1^2 + Θ2^2) = ½($\sqrt{\Theta 1^2 + \Theta 2^2}$)^2 = ½||Θ||^2

Margin 최대화

- 벡터 내적 수식인 $\mathbf{u^T * v = p||u||}$을 이용해서,
- $\mathbf{\Theta^T * x(i) = p(i)||\Theta||}$ 로 나타낼 수 있다.

Margin 최대화

- 비용함수를 최소화하기 위해 Θ값을 구해야 한다.

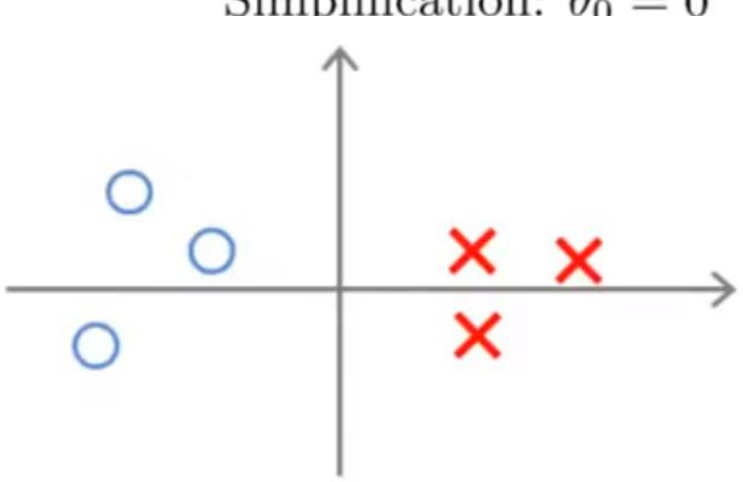
$\min_\theta \frac{1}{2}\sum_{j=1}^{n}\theta_j^2$ = ½||Θ||^2 (C는 엄청 큰 값을 가진다)

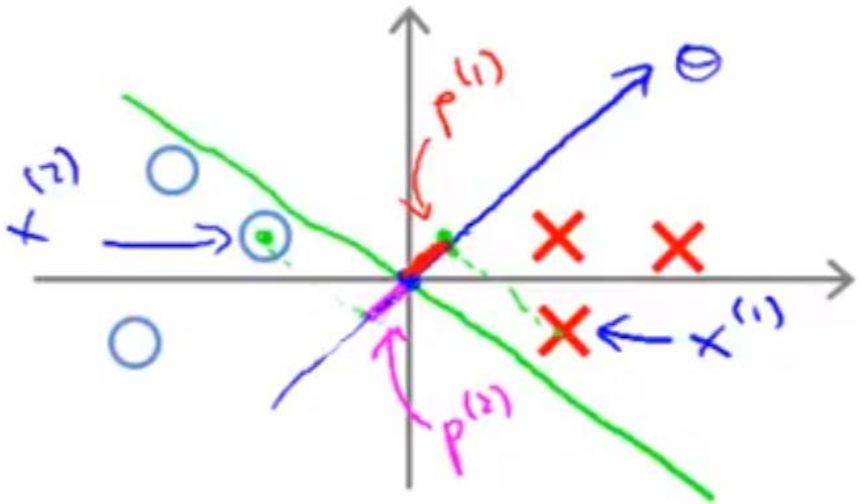
s.t. $p^{(i)} \cdot \|\theta\| \geq 1$ if $y^{(i)} = 1$ (조건에 대입)

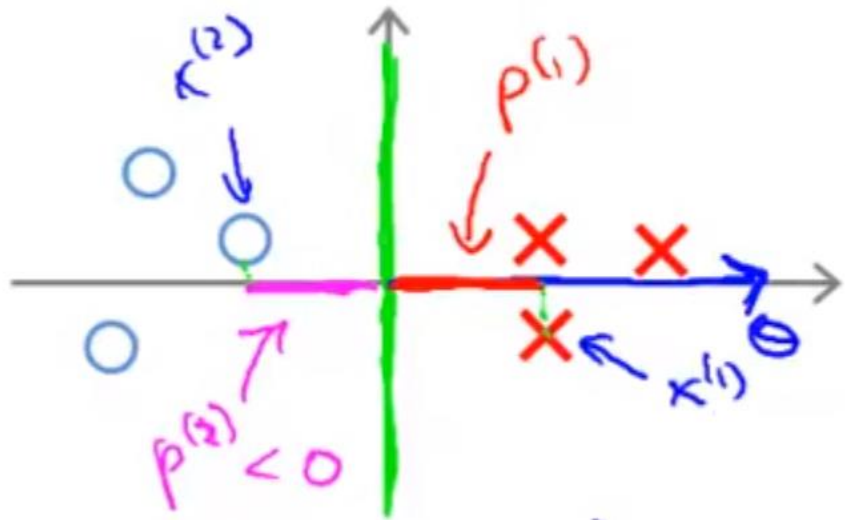
$p^{(i)} \cdot \|\theta\| \leq -1$ if $y^{(i)} = 1$

where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ.

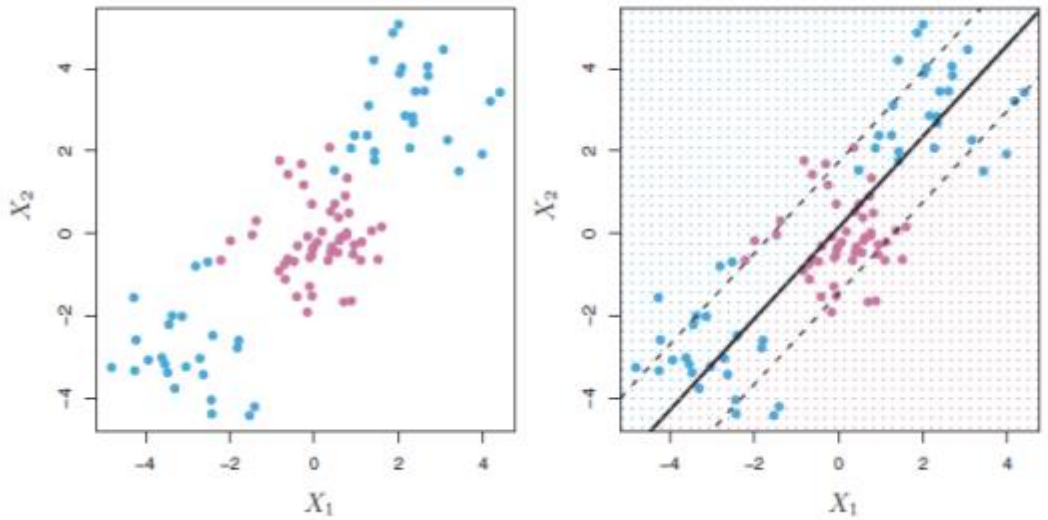
Simplification: $\theta_0 = 0$





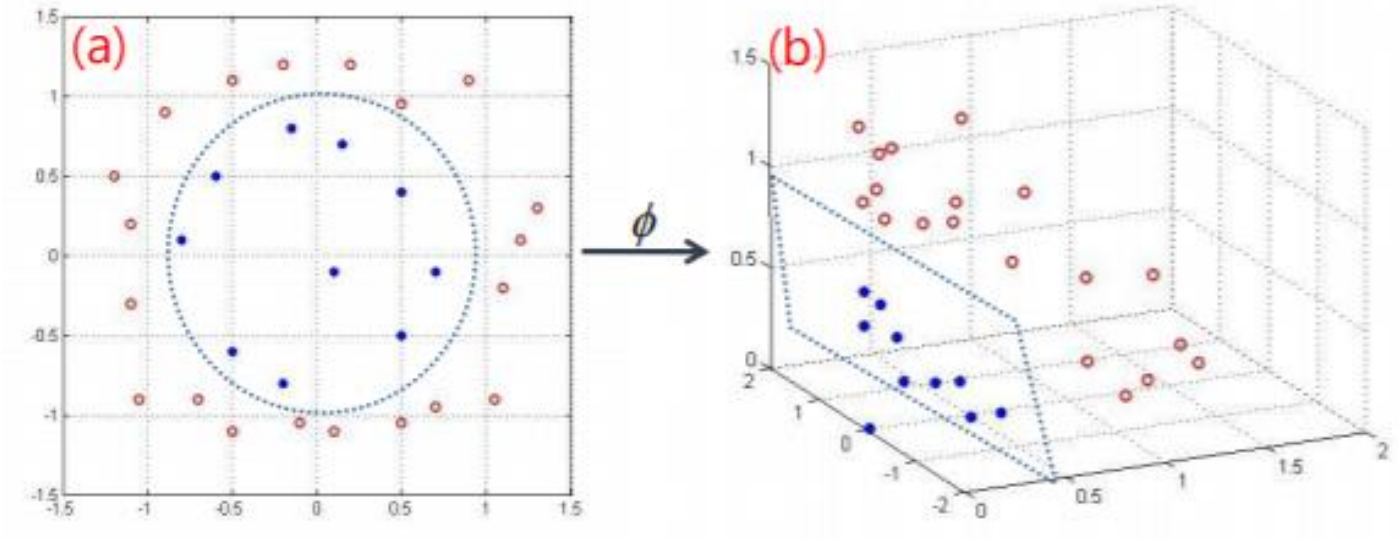


비선형 분류 문제



$$X_1, X_1^2, X_2, X_2^2, \ldots, X_p, X_p^2.$$

다항식이나 비선형 함수를
표현하는 변수를 사용한다면?



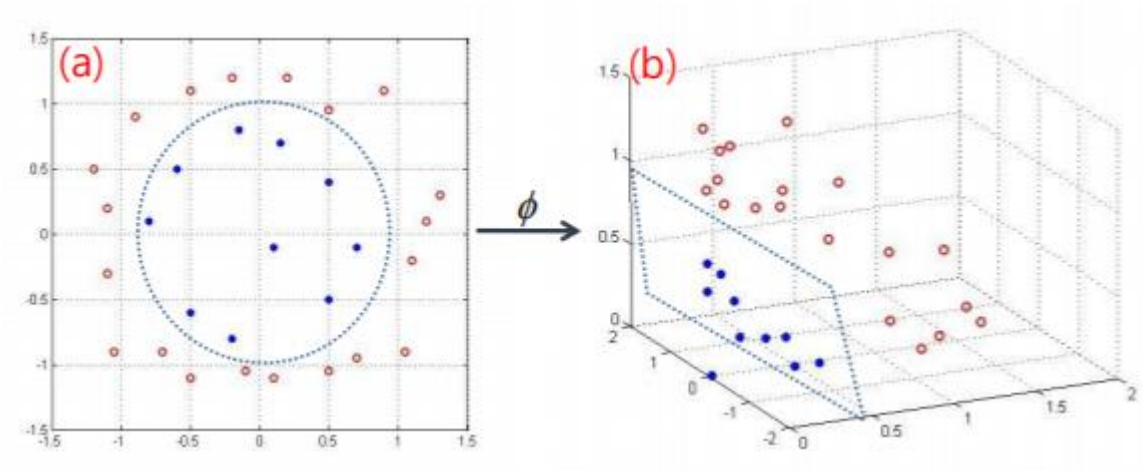
비선형 분류 문제

- 고차원으로 선형화하면
  간단한 선형 분류기를 사용한 분류 가능!
- 성능상의 문제도 없지만 차원이 아주 큰 고차원 변환 시 너무 많은 연산비용이 소모된다.
  → **커널 함수** 사용하여 해결

커널함수

- SVM에서 연산은 개개의 Φ(x)가 아니라 두 벡터의 내적 Φ(x) · Φ(y)를 사용
- 고차원 매핑 Φ(x) 대신에 Φ(x) · Φ(y)를 하나의 함수 **k(x,y)**로 정의하여 사용한다.

커널함수



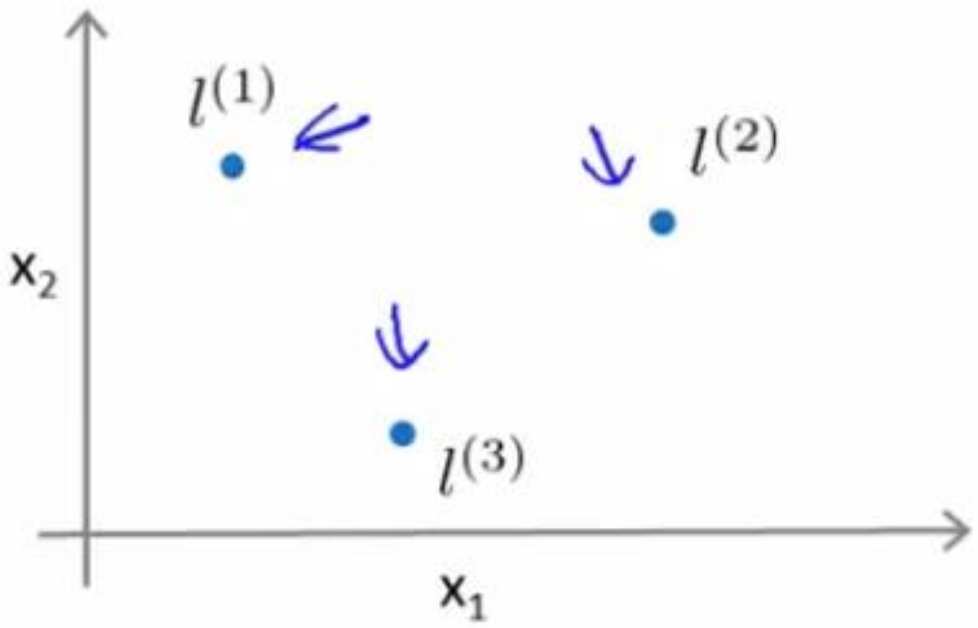
커널함수

대표적인 커널 함수

다항식 커널	$k(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} \cdot \boldsymbol{y} + c)^d$
시그모이드 커널	$k(\boldsymbol{x}, \boldsymbol{y}) = \tanh(\theta_1 \boldsymbol{x} \cdot \boldsymbol{y} + \theta_2)$
가우시안 커널	$k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left\{-\frac{\| \boldsymbol{x} - \boldsymbol{y} \|^2}{2\sigma^2}\right\}$

- 문제에 따라 최적의 커널 함수를 직접 학습, 테스트 해서 찾아야 한다.

커널함수

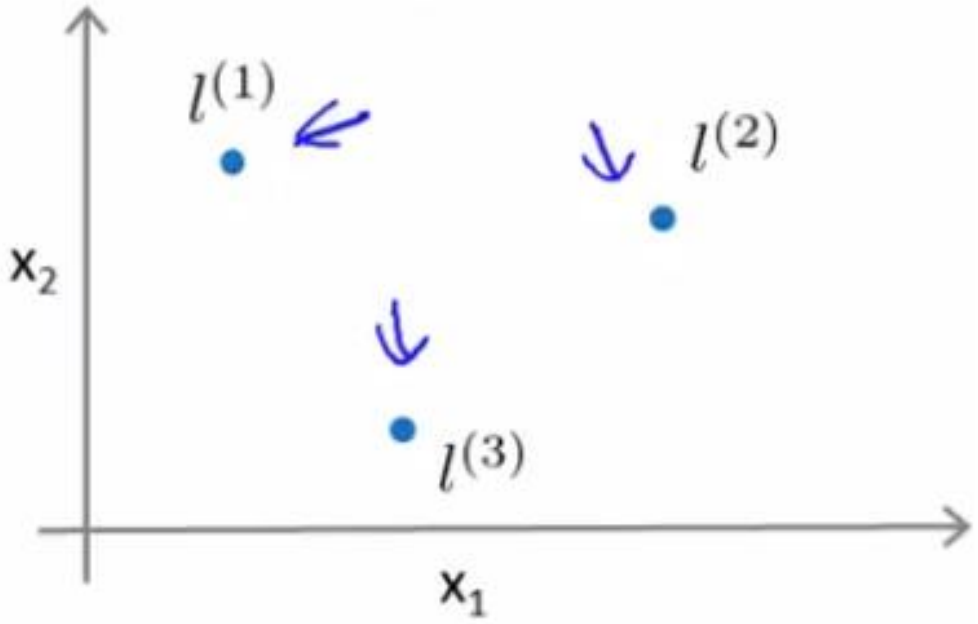


$$f_i = similarity(x, l^{(i)})$$

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

$$= \exp(-\frac{\sum_{j=1}^{n}(x_j - l_j^{(i)})^2}{2\sigma^2})$$

커널함수

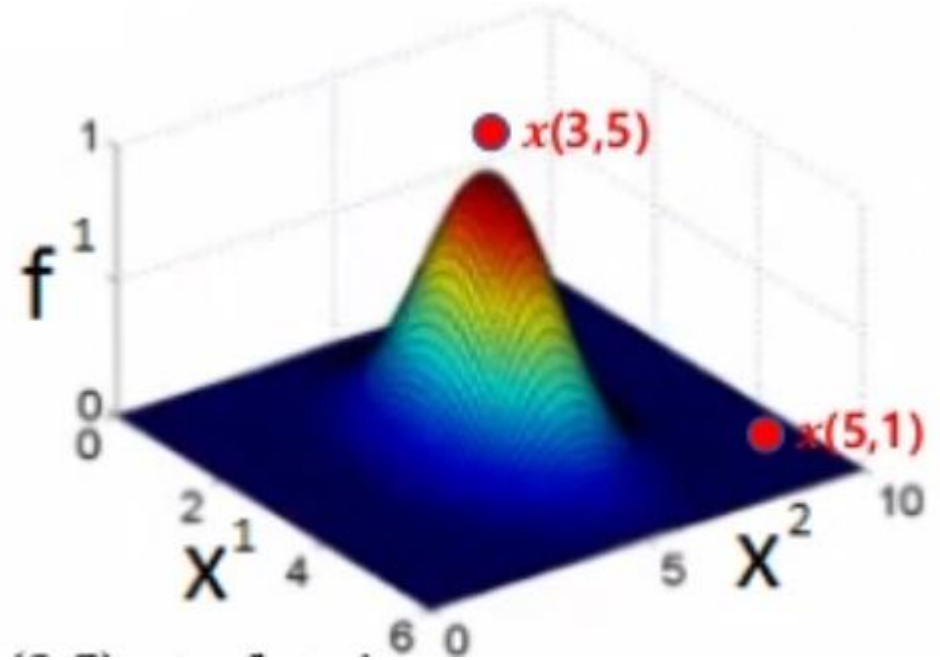


if $x \approx l^{(i)}$, then $f_i = \exp(-\frac{\approx 0^2}{2\sigma^2}) \approx 1$

if x is far from $l^{(i)}$, then $f_i = \exp(-\frac{(large\ number)^2}{2\sigma^2}) \approx 0$

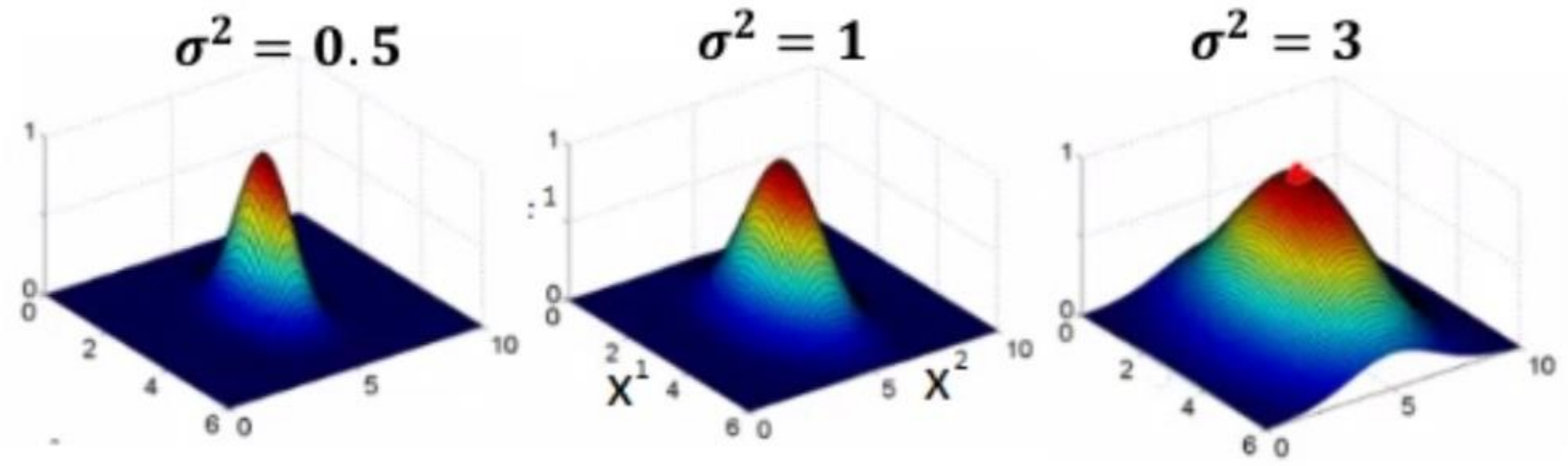
커널함수

- 랜드마크 l1 = (3,5)
- X가 l1에 가까워지면 f1 ≈ 1,
- X가 l1에 멀어지면 f1 ≈ 0



커널함수

- σ^2 크기에 따른 경사의 변화



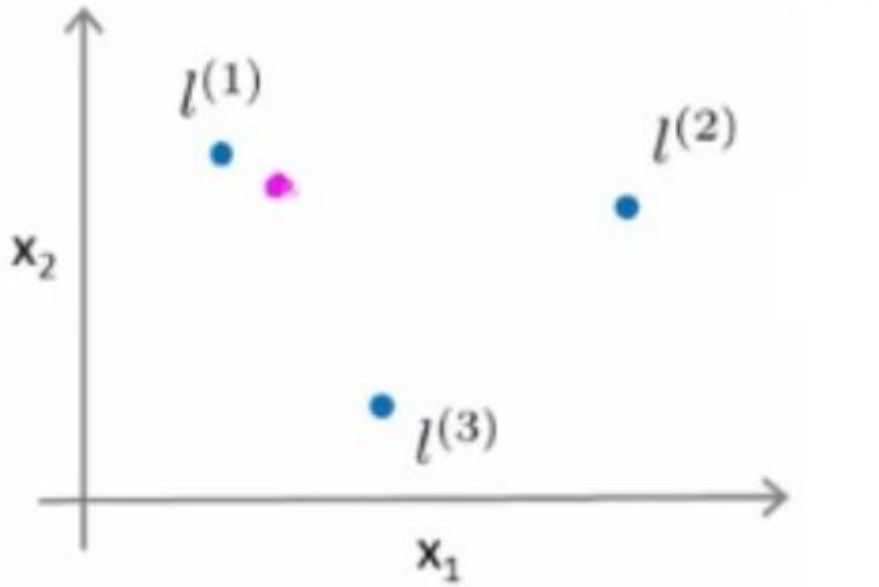
커널함수

가정:
학습샘플 x가 아래 조건을 만족하면 정답 1을 예측한다.

$$Return\ 1\ when\ \ \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$$

커널함수(예)

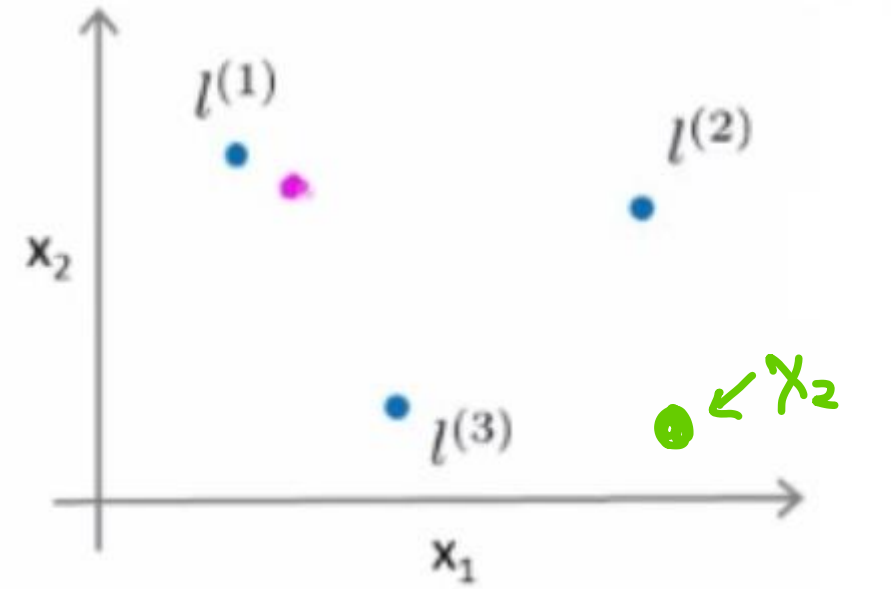
예) Θ0 = -0.5
Θ1 = 1
Θ2 = 1
Θ3 = 0



- $\theta_0+\theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$
- $= -0.5 + 1 * 1 + 1 * 0 + 0 * 0 = 0.5 \geq 0$ ➔ 1을 예측

커널함수(예)

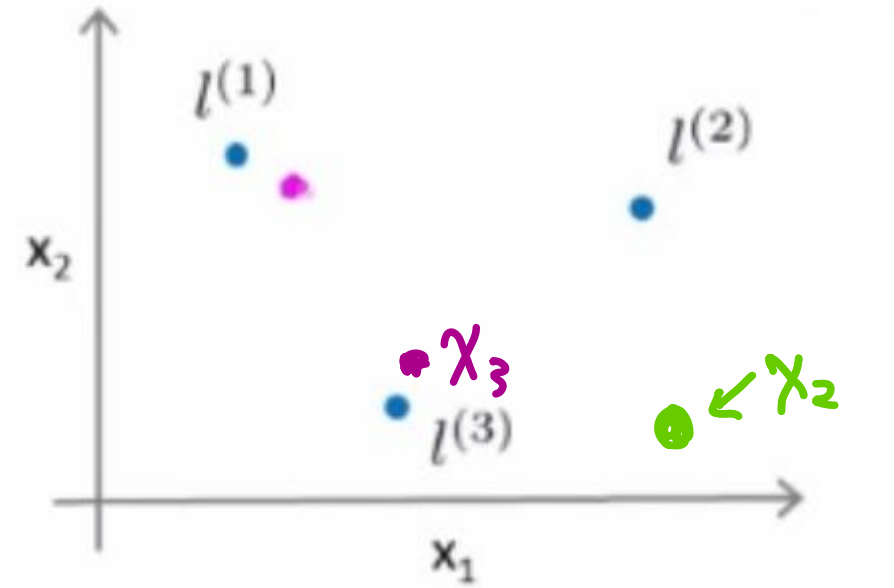
예) Θ0 = -0.5
Θ1 = 1
Θ2 = 1
Θ3 = 0



- $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 \geq 0$
- $= -0.5 + 1 * 0 + 1 * 0 + 0 * 0 = -0.5 < 0$ → 0을 예측

커널함수(예)

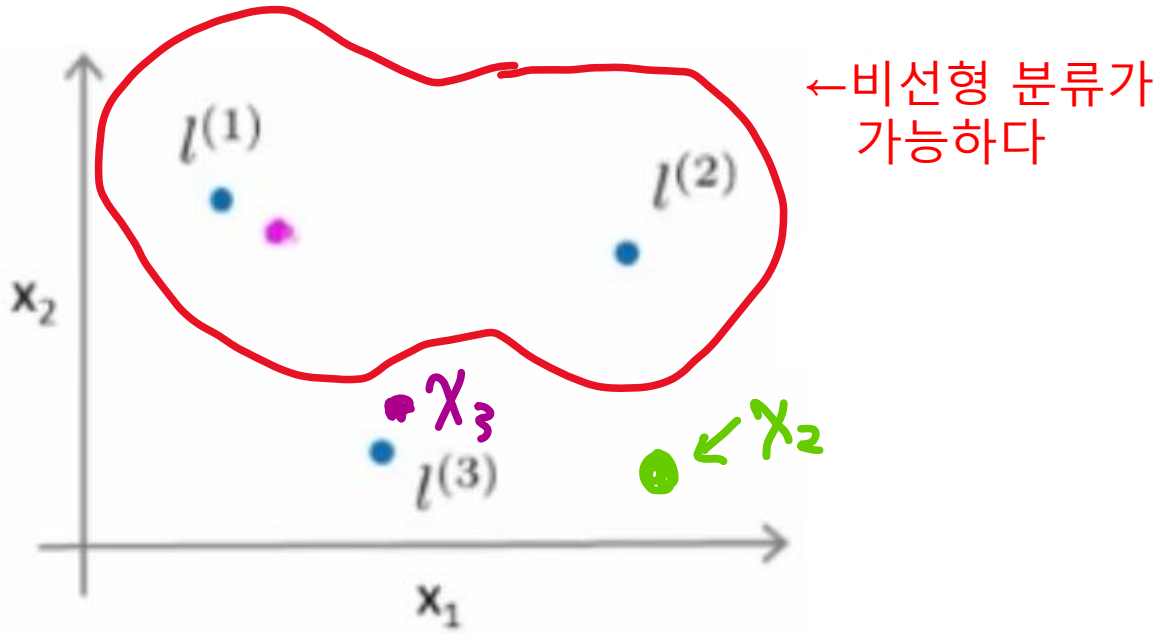
예) Θ0 = -0.5
Θ1 = 1
Θ2 = 1
Θ3 = 0



- $\theta_0+\theta_1 f_1+\theta_2 f_2+\theta_3 f_3 \geq 0$
- $=-0.5+1*0+1*0+0*1=-0.5<0$ ➔ 0을 예측

커널함수(예)

예) Θ0 = -0.5
Θ1 = 1
Θ2 = 1
Θ3 = 0



←비선형 분류가 가능하다

파라미터 값(θ_1, θ_2, θ_3)이 서로 다르게 정의되어 있으므로, 이 값에 따라 예측이 다르게 됨.

- $\theta_1, \theta_2 = 1$ → f_1, f_2 값이 분류에 영향을 미침
- $\theta_3 = 0$ → f_3 값이 영향을 주지 못함.

랜드마크 선택하기

1. 학습 데이터를 읽어온다. (100건)
2. 학습 데이터 별로 landmark를 생성한다. (학습데이터와 동일한 위치)
   - 학습데이터 1건 별로 1개의 landmark가 생성됨
   - 최종 100개의 landmark 생성
3. 새로운 샘플이 주어지면, 모든 landmark와의 거리(f)를 계산한다.
   - $f_0 \sim f_{99}$, 총 100개의 f 결과
   - $f_0 = 1$ (θ_0는 bias 값이므로, 값을 그대로 유지)
   - 자세히 계산과정을 보면
     - $f_1^i = K(x^i, l^1)$
     - $f_2^i = K(x^i, l^2)$
     - ....
     - $f_{99}^i = K(x^i, l^2)$
   - 위 과정을 반복하면, X 자신과 동일한 landmark와 비교하는 구간이 있다. → 이 경우 Gaussian Kernel에서는 1로 평가 (동일한 위치)

X값이 vector(배열)로 구성된 경우

- 이렇게 계산된 f 값을 [m(99)+1 x 1] 차원의 vector로 저장한다.
- f^i → f 벡터의 i 번째 데이터를 의미

랜드마크 선택하기

1. 이전에 정의했던 수식을 확인해보자
   - $Predict\ y = 1$
   - $when\ \ \theta^T f \geq 0$
   - $And\ f = [m + 1 * 1]$
2. 그럼 θ는 어떻게 계산할 수 있을까?
   - SVM Optimization 알고리즘 이용

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

   - 위 최적화 결과를 최소화하기 위해 f 벡터를 이용한다.
   - 그리고 이 최적화 알고리즘을 계산하면, θ를 찾을 수 있게 된다.
3. 계산 성능 향상을 위한 Tip
   - 위 예시에서 m = n으로 가정 (학습 데이터와 f 벡터의 수가 같기 때문)

$$\sum_{j=1}^{n} \theta_j^2 = \theta^T \theta \quad \Rightarrow \quad \theta^T M \theta$$

   - 좌측방식 구현보다는 우측 방식으로 구현하는 것이 계산성능 향상
   - 데이터가 많을 경우 수많은 for loop를 하지 않고, 매트릭 계산

Bias, Variance trade off

SVM에서 bias와 variance는 C의 값과 σ^2에 의해 조절이 가능

C($\frac{1}{\lambda}$) 가 클 때: Lower bias/high variance

작을 때: Higher bias/low variance

σ^2 가 클 때: 완만한 형태, Higher bias /low variance

작을 때: 가파른 형태, Lower bias/high variance

SVM의 장단점

- **장점**
- 학습 데이터에 over fitting을 방지
- Kernal method 활용을 통해 비선형 데이터 분류도 가능하다.
- **단점**
- 학습 데이터의 margin이 적을 때 문제가 발생할 가능성이 있다.
- Support vector(margin) 근처의 데이터만 고려를 하며, 고차원 데이터에서 효율적이다.

SVM 라이브러리

SVM 라이브러리

LIBSVM: 서포트 벡터 머신만을 위한 라이브러리로 파이썬, R, 외 다양한 언어를 지원

파이썬

scikit-learn: 머신 러닝 기법들을 사용하는데 도움을 주는 라이브러리
PyML: 머신 러닝 기법들을 사용하는데 도움을 주는 라이브러리

R

klaR: SVM 알고리즘을 활용한 함수 패키지
Kernlab: 초심자가 쉽게 사용할 수 있는 패키지