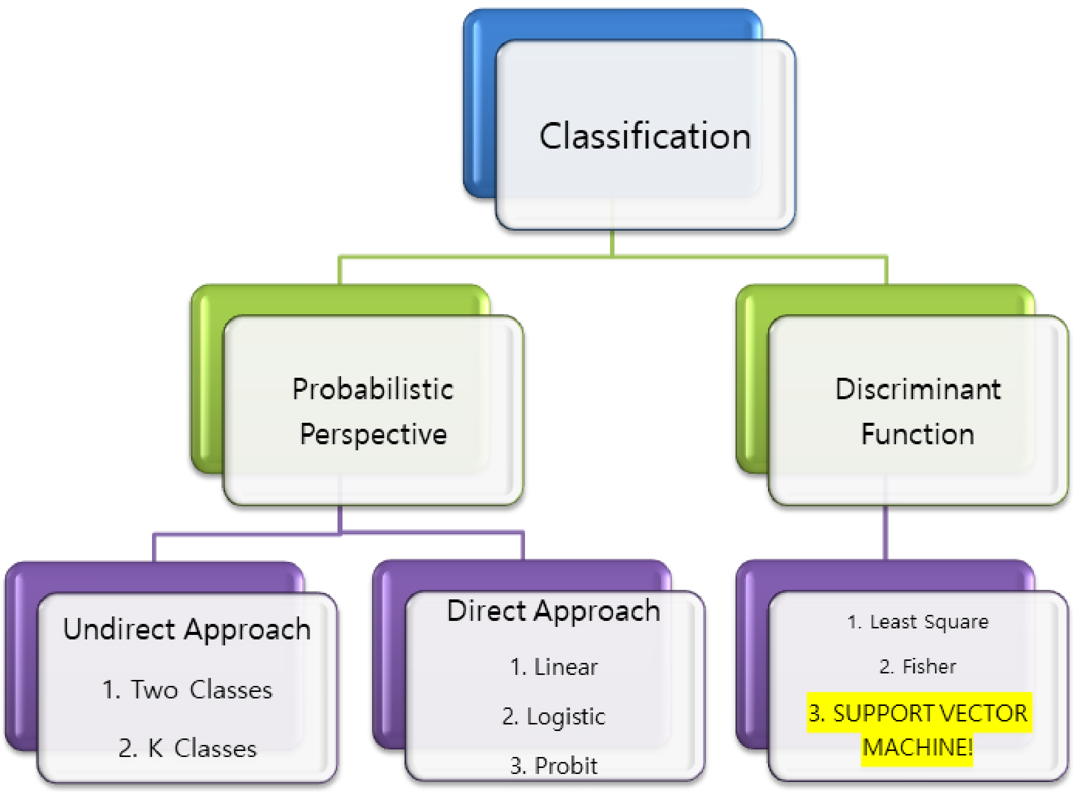
Week 4. Kernel and Support Vector Machine

임선우 ESC YONSEI UNIVERSITY

4/22/2020

1. Where are we?



SVM : No consideration of $p(C_k|X)$

2. Hard SVM without Expansion of Basis

2-1. Good Discriminant Function : Halfway Down!

“Good Linear Discriminant Function”을 알아보자!

"Target" $t(= y)$: $t_n = 1 or - 1$: labeled!

“Estimator” of t" : $y(x) = w^T x + b$

"Class1" y(x) = $w^T x + b > 0$: 한 쪽 면

"Class2" y(x) = $w^T x + b < 0$: 다른 한쪽 면

"Discriminant Function" : $y(x) = w^t x + b = 0$: 판별함수

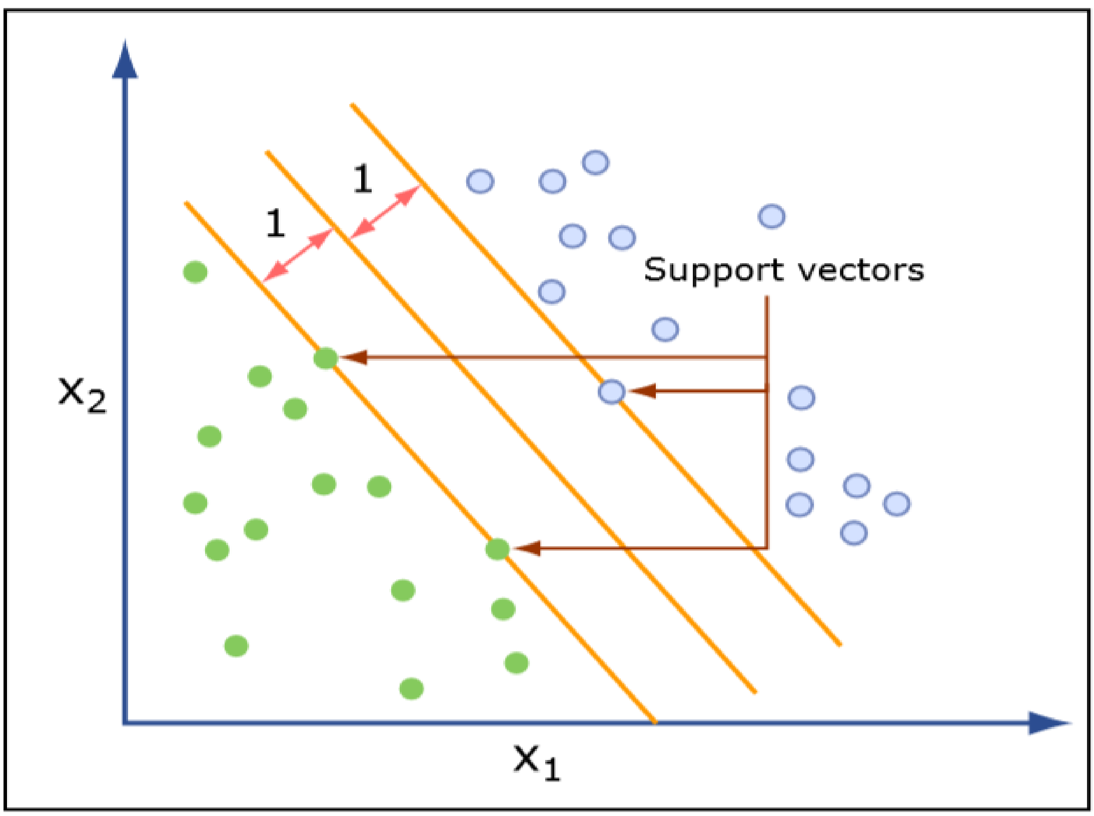
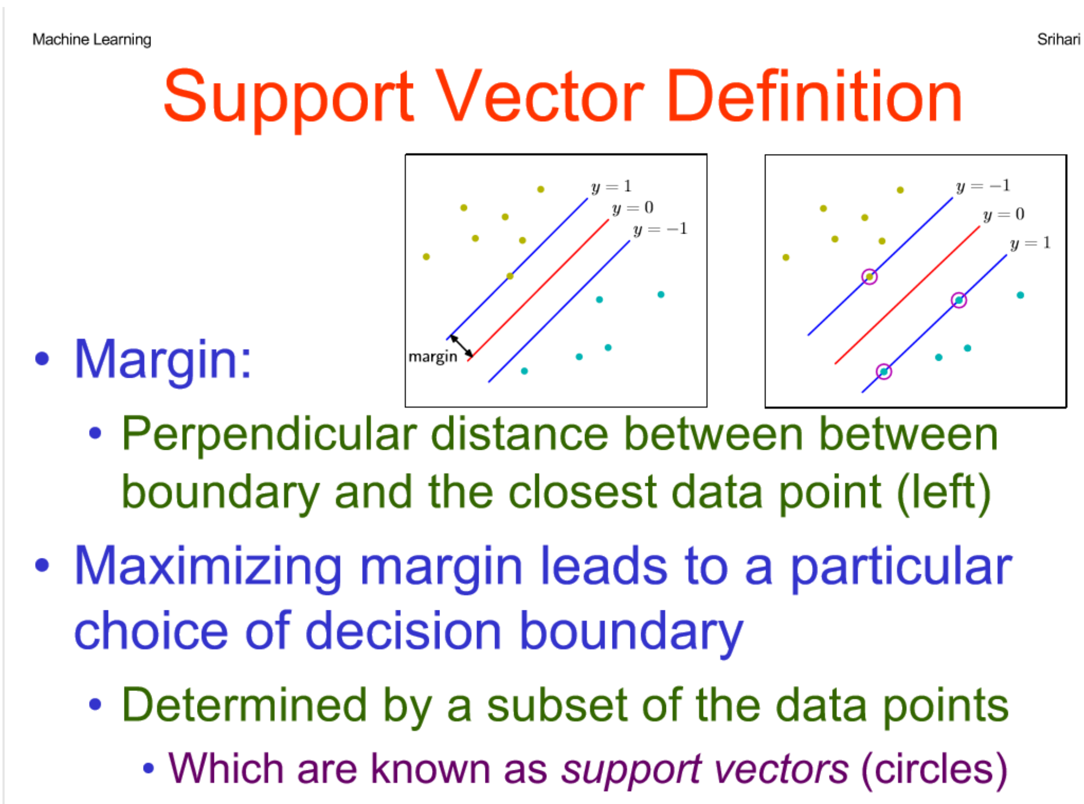


Image by MIT OpenCourseWare.

2-2. Hard Margin

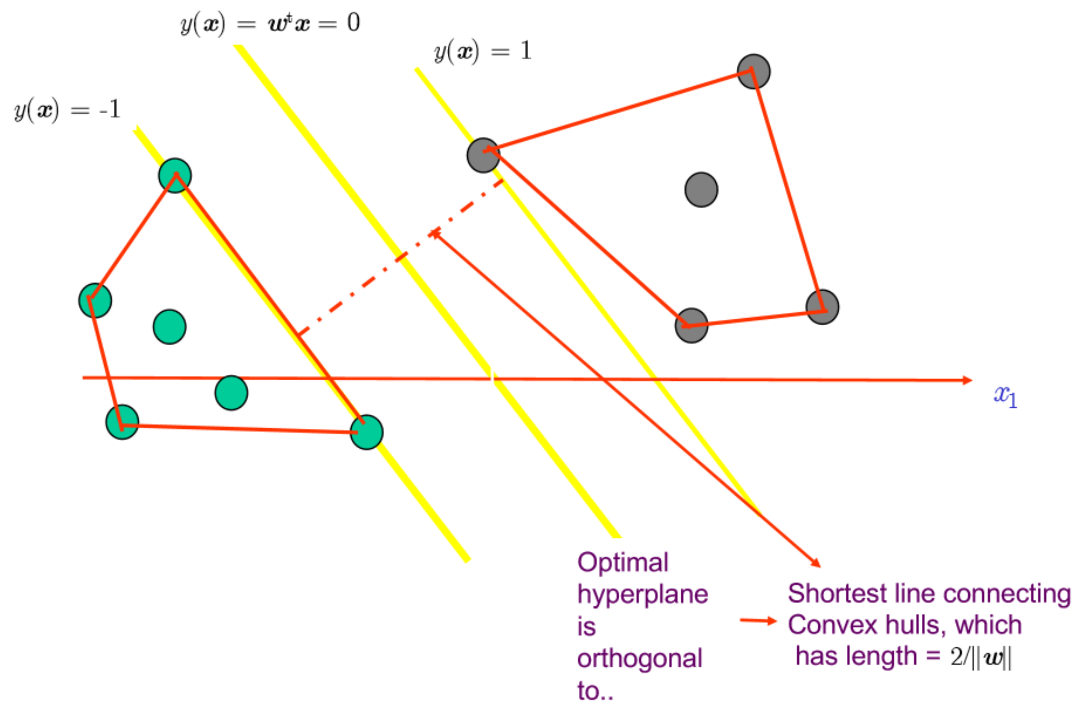


아까 정의할 때는 **0** 을 경계로

- **Class1** y(x) = $w^T x + b > 0$: 한 쪽 면
- **Class2** y(x) = $w^T x + b < 0$: 다른 한쪽 면으로 정의하였다.

그러나 모델을 만들 때는

- $C_1 : w^T x + b \geq 1, C_2 : w^T x + b \leq 1$으로 **표준화**한다. (1 과 -1 은 표준화로 생각하면 된다!)
- **Support Vector** 의 경우 각 Class 에서 **등호**에 해당한다!



이렇게 하면 아까 정의한 **margin** 은 $w^Tx + b = 0$과 $w^Tx + b = 1$ 사이의 거리, $w^Tx + b = -1$ 사이의 거리로 생각하면 된다!

2-3. Maximum Hard Margin Classifier

2-3-1. 문제 세팅

- 지금까지 예시는 **Input Space** 가 2 차원, **Discriminant Function** 이 1 차원이었다!
- 그러나 일반화하면 **Input Space 가 p dimension** 이면 **Discriminant 는** p-1 dimension 의 Hyperplane!**
- 즉, 어떻게 Discriminant Function 을 세우면 **평면과 Support Vector 의 거리를 최대화**할 수 있는지 알아야 한다!
- **spoiler: 평면과 각 클래스의 최전선에 있는 벡터 간의 거리를 최대화하도록 한다!**
- 이제, 점과 평면 사이의 거리(Linear Algebra) 복습!

절대값이 포함된 거리 식이 나오게 된다.

2-3-2. 절대값 없애기

지금 구한 이 거리에는 **절대값**이 분자에 있다! (미분불가능한 문제) 절대값을 없애야 **미분가능한 Optimization Problem** 이 된다!

해결법 : multiply by $t_n \in (-1,1)$

- $t_n = 1$ (adult, conservative,..) : Want $w^T X_n + b \geq 1$
- $t_n = -1$ (children, democratic, ..) : Want $w^T X_n + b \leq 1$
- $t_n(w^T(x_n) + b) \geq 1, \forall n \in (1,2,\ldots,N)$
- $\therefore Margin = min \frac{t_n(w^T(x_n)+b)}{\|w\|}$
- So, Maximum Margin Solution : $argmax_{w,b} min \frac{t_n(w^T(x_n)+b)}{\|w\|}$

2-3-3. Margin 식 간단화

- 이거를 그대로 푸는 것은 너무 어렵다! 생각의 Brilliance 가 필요!
- 지금부터 할 얘기를 용이하게 할 개념 2 개 : $Active, Inactive$
- Active points : $t_n(w^T(x_n) + b) = 1$ = Support Vectors
- Inactive points : $t_n(w^T(x_n) + b) > 1$ = Non- Support Vectors
- 각 Class 에는 적어도 하나의 Active Point 가 있다! 적어도 2 개의 Active Point 가 있다!
- Active Point 들은 $t_n(w^T(x_n) + b)$을 1 로 최소화한다!

2-3-4. 중간결과

- $\therefore Margin = \frac{1}{\|w\|}$
- ∴ Constrained Optimization Problem:
- $argmax_{w,b} \frac{1}{\|w\|}$ such that $t_n(w^T(x_n) + b) \geq 1, \forall n \in (1,2,\ldots,N)$
- $\frac{1}{\|w\|}$ 를 최대화 ≡ $\frac{1}{2} \| w \|^2$를 최소화 (계산의편의)!
- min $L(w,b,\lambda) = \frac{1}{2} \| w \|^2 - \sum_{n=1}^{N} \lambda_n [t_n(w^T X_n + b) - 1], \lambda_n \geq 0, \forall n$

SVM Constrained Optimization

$$\text{Optimize} \quad \arg\min_{\boldsymbol{w},b} \frac{1}{2} || \boldsymbol{w} ||^2 \quad \text{subject to constraints} \quad t_n(\boldsymbol{w}^t \phi(\boldsymbol{x}_n) + b) \geq 1, \quad n = 1, \ldots. N$$

- Can be cast as an unconstrained problem
- by introducing Lagrange undetermined multipliers with one multiplier $a_n \geq 0$ for each constraint
- The Lagrange function we wish to minimize is

2-3-5. 이 Sequential Lagrange 에서 최적화의 필요조건 KKT 조건

1) $\frac{\partial L}{\partial w} = 0: w = \sum_{n=1}^{N} \lambda_n t_n x_n$

2) $\frac{\partial L}{\partial b} = 0: \sum_{n=1}^{N} \lambda_n t_n = 0$

3) $\forall n, \lambda_n \geq 0$

4) $\forall n, \lambda_n = 0 \text{ or } t_n(w^T(x_n) + b) = 1(\text{Support Vectors})$

이 조건을 만족하므로 위 문제는 Dual Optimization Problem of Maximization 으로 바뀜.

2-3-6. 식을 푼다

- Dual OP of maximizing
- $\tilde{L}(\boldsymbol{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_m a_n t_n t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m)$ over N data points

마찬가지로 b의 solution 도 구할 수 있게 된다.

- Solving for b gives $b = \frac{1}{N_S}\left(t_n - \sum_{m \in S} a_m t_m k(\boldsymbol{x}_n, \boldsymbol{x}_m)\right)$
  - Where N_S is the total no of support vectors

결국 이 복잡하게 생긴 각 Solution 들을 넣어 **Prediction** 은 $y(x)$의 부호에 따라!!

- Evaluate sign of $y(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b$
  - This can be expressed in terms of the parameters $\{a_n\}$ and the kernel function by substituting for $\boldsymbol{w}$ using $\boxed{\boldsymbol{w} = \sum_n a_n t_n \boldsymbol{x}_n}$ to give

$$\boxed{y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n t_n k(\boldsymbol{x}, \boldsymbol{x}_n) + b}$$

2-4. Error Function of Hard Margin Classifier

Supervised Machine Learning 에서는 Error 의 개념이 중요!

Machine Learning Srihari

Equivalent Error Function of SVM

- For comparison with alternative models
  - Express the maximum margin classifier in terms of minimization of an error function:

$$\boxed{\sum_{n=1}^{N} E_\infty\left(y(\boldsymbol{x}_n)t_n - 1\right) + \lambda \|\boldsymbol{w}\|^2}$$

  - Where $E_\infty(z)$ is zero if $z \geq 0$ and ∞ otherwise

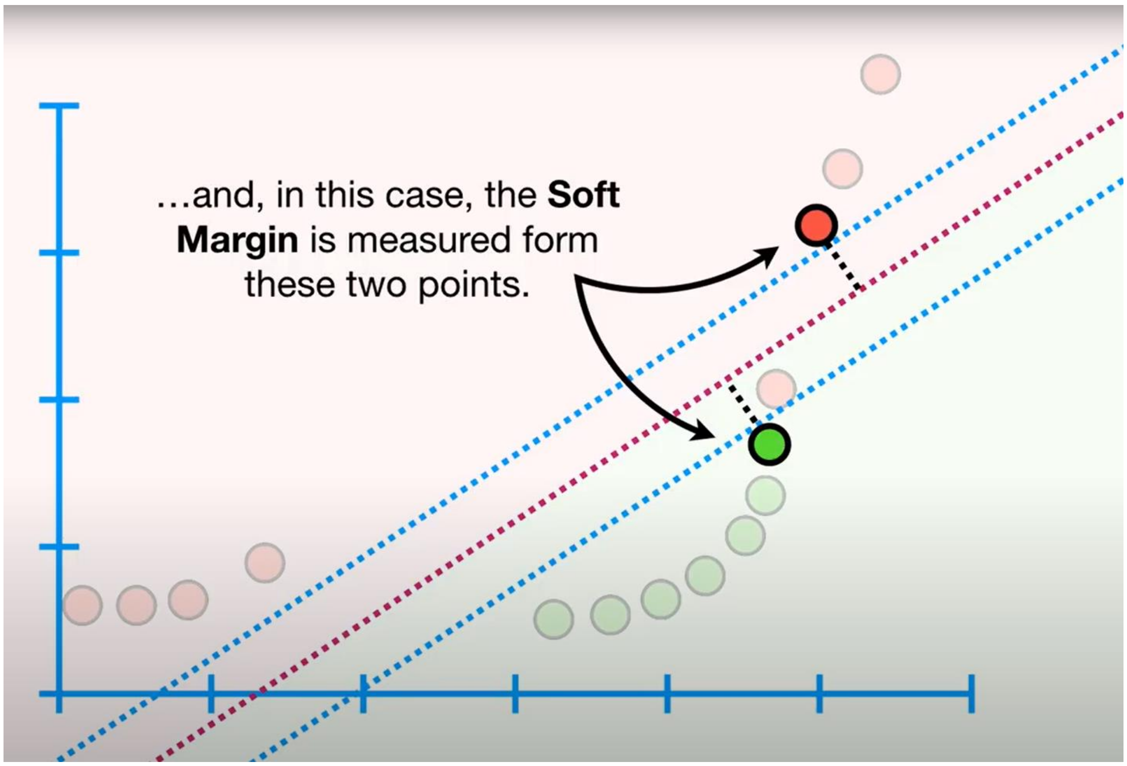
이 Hard Margin SVM 은 Misclassification 이 하나라도 생기면 Error 가 무한!

0% $TRAIN$ error rate 을 추구한다.

나중에 나올 Soft Margin SVM 과 비교해 Small Bias, Big Variance.

3. Soft SVM without Expansion of Basis

Hard Margin 은 Outlier 에 너무 민감하다!



3-1. Jargons

- **Soft Margin** : Distance between the Discriminant Function and the Frontier observation, Allowing Misclassification IN SAMPLE.

- 적절한 Soft Margin 은 **Validation** 을 통해 정한다! (Optimal Problem 이 X)

Hard Margin vs Soft Margin (Bias Variance Tradeoff 시소)

- **Hard Margin** : Bigger Variance, Smaller Bias

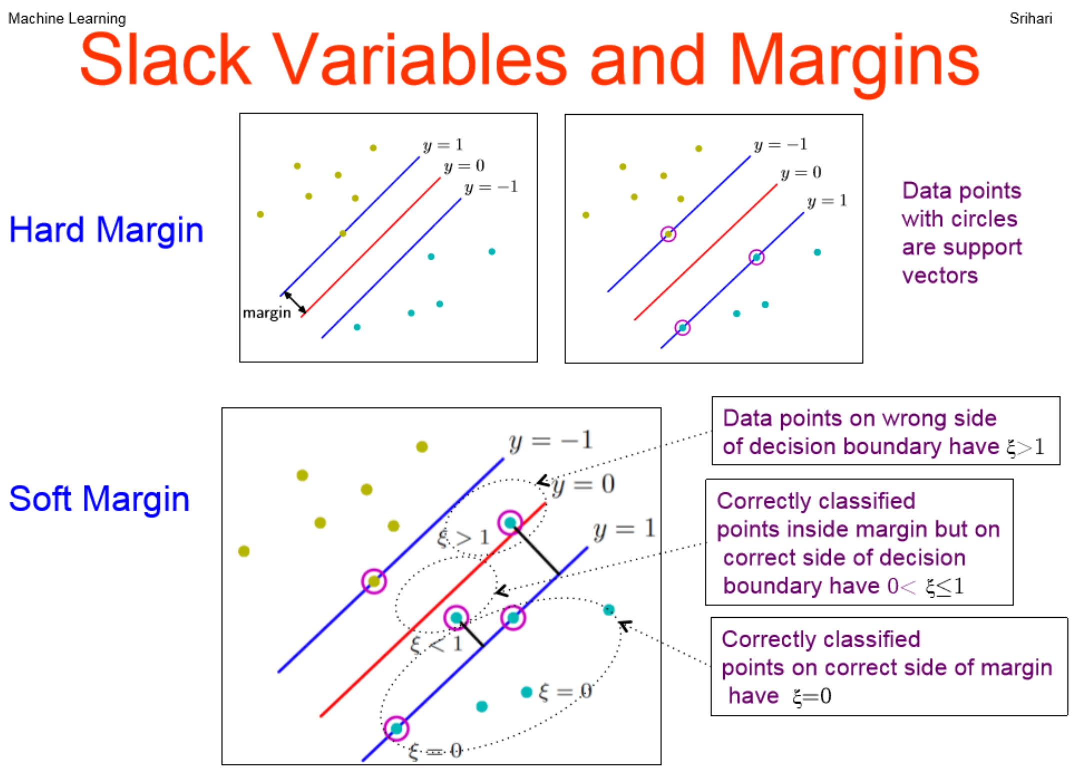
- **Soft Margin** : Bigger Bias, Smaller Variance. (Bias 를 허용하되 Variance 를 줄이겠다)

- **Soft Margin** 은 **Ridge, Lasso** 와 분야는 다르나 참 비슷한 게 많다! (아이디어 뿐 아니라 나중에 Error 의 Penalty Term 도!)

3-2. Error Function

3-2-1. Our new Friend “Xi”

ξ_n (Slack Variable) : 옳은 Discriminant Function 에서 거리가 멀면 멀수록 LINEARLY 커지는 친구



자율적으로 생각해보기 : 만약 ξ_n이 올바른 Discriminant Function 과의 거리의 제곱에 비례하도록 설정한다면

- 4 가지 경우의 이해!

1) $\xi_n = 0$
2) $0 < \xi_n < 1, \xi =| t_n - y(x_n) |$
3) ξ_n =1, $\xi =| t_n - y(x_n) |$
4) $\xi_n > 1, \xi =| t_n - y(x_n)$

3-2-2. Error Function

- We therefore minimize $\boxed{C\sum_{n=1}^{N}\xi_n + \frac{1}{2}\|\boldsymbol{w}\|^2}$
  - Parameter $C>0$ controls trade-off between slack variable penalty and the margin
- Subject to constraints

  $t_n y(\boldsymbol{x}_n) \geq 1\text{-}\xi_n \;\; n=1,..,N$

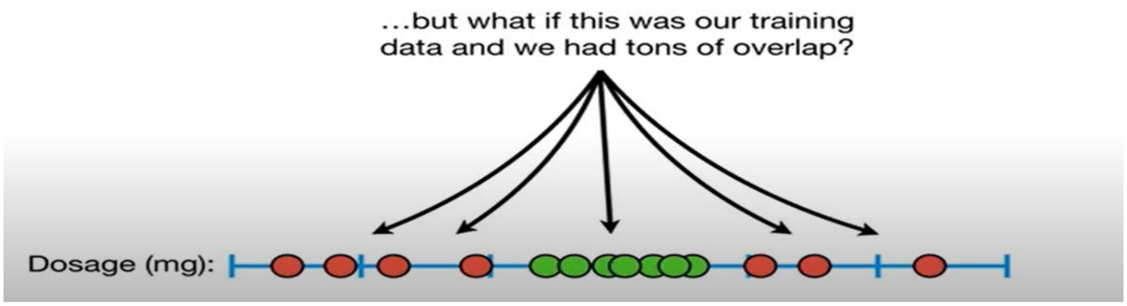
3-2-3. C 의 의미 (역시 C 도 Validation)

Large C : $\sum_{n=1}^{N}\xi_n$이 작아야! = 오분류로 인한 Penalty 에 신경 많이 씀 = **하드마진과 가깝다**

Small C : $\sum_{n=1}^{N}\xi_n$이 커도 되고 큰 마진을 신경쓴다! = 오분류로 인한 Penalty 에 신경 덜 씀 = **하드마진과 멀다**

Bias Variance Tradeoff Again!

4. Problems and Kernel Introduced

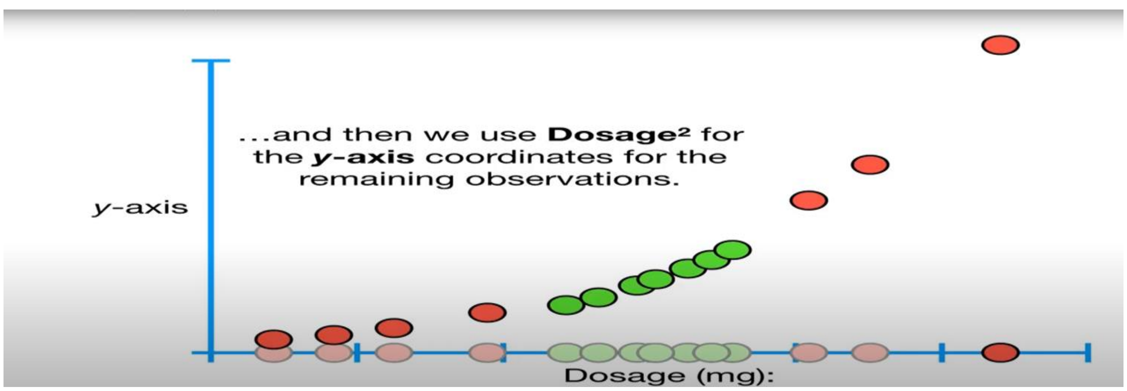


4-1. Problems. 어떡해?

후보 1) 절반정도의 misclassification 을 발생시킨다

후보 2) 이건 Support Vector Machine 이 아니야라고 포기한다

4-2. 해결방안 : SVM 은 이것도 선형으로 분리할 수 있어 (...) ?!?!



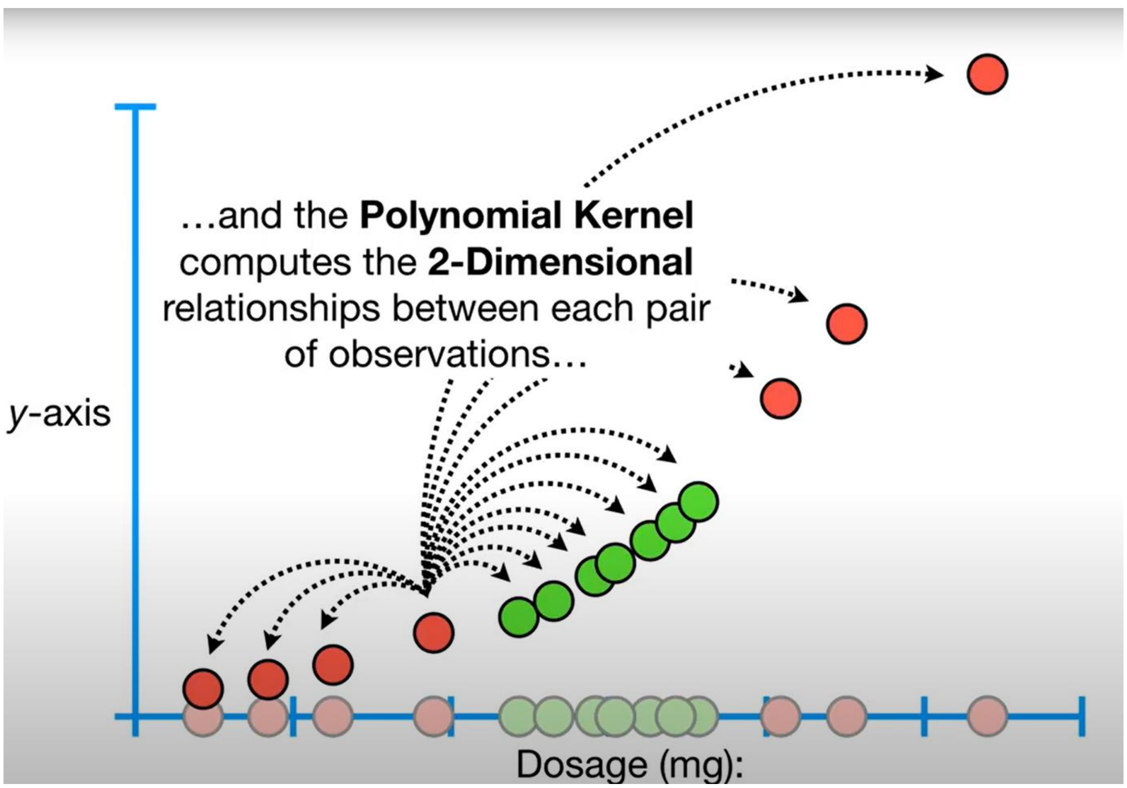
- **해결방안 : Input Space 에서 차원을 확장한 Feature Space 에서 Linearly Separable Hyperplane !!**
- **주의점** : 우리가 Client 한테 보고할 때는 고차원(Augmented Space)에서 나눈 경계를 저차원에 환원시켜 보고한다. 이 그림에서는 Curvature 경계가 된다.

- **고차원 Feature Space 를 만들고 점 간의 유사도를 측정**하는 **Kernel Function** 이 필요한 것
- 그래서 앞에서 쓴 모든 X 를 일반화하여 $\phi(X)$로 다 바꿔! 아까는 Maximum Margin Problem 의 원활한 이해를 위해 $\phi(X) = X$인 특수 케이스를 다뤘던 것이다!

4-3. Kernel 은 두가지 일을 하는 친구다.

1st) Input Space 에서 고차원 Feature Space 로 확장!

2nd) 모든 점들 간의 유사도 조합을 계산 : matrix 형태를 띈다!



Q) : Why x^2? How about x^{e3}? How to decide 'The Best' Dimension?

A) Cross Validation…

5) What is Kernel Mathematically? (Similarity?)

Definition) $\boldsymbol{k(x_i, x_j) = \phi(x_i)^T \phi(x_j)}$ **,** $\boldsymbol{x_i, x_j \in R^n}$

Definition) $\boldsymbol{\phi(X)}$ **is a fixed nonlinear mapping (basis Function) from input space** $\boldsymbol{R^n}$ **to Inner product space** $\boldsymbol{F}$ **in** $\boldsymbol{R^m}$

- 왜 계속 내적 이야기? **내적이 Similarity?**

-**Yes!** 두 벡터 간 각도를 0 도에서 180 도까지 생각해보자! (**Cosine Similarity** :
ESC 면접문제)

- Disclaimer : 여기서의 Kernel Function 은 시계열에서 배운 Kernel Smoothing 에서의 Kernel Density 와 다르므로 구별!

6) Kernel Trick : For Computational Ease

만약 초고차원 확장이 필요한 굴곡진 데이터면 어떨까? 원래대로면 computationally intensive

그래서 컴퓨터는 실제로 고차원 변환 후 내적을 하지 않고 그와 똑같은 계산법을 통해 저차원에서 내적을 한다!

예시 하나를 들어보자!

- $\psi(\vec{x}_i) = \phi([x_{i1}, x_{i2}]) = (x_{i1}^2, \sqrt{2}x_{i1}x_{i2}, x_{i2}^2)$

Kernel Trick 을 쓰지 않고서는 **11 번의 계산**이 필요한데 반면 Kernel Trick 을 쓰면 **3 번의 계산으로 OK**

Mercer's thm) $K(x_i, x_j)$ 가 PSD matrix iff $K(x_i, x_j)$ can be expressed as $\phi(x_i)^T\phi(x_j)$

해석 : 위 조건을 만족하면 $\phi(X)$계산을 거치지 않고 고차원에서의 Similarity 를 계산가능

7) Examples of Kernel

7-1) Polynomial Kernel

$k(x_i, x_j) = (x_i^T x_j + C)^d$, C 와 d 는 Validation 으로..

- d 는 차원 확장의 차수를 말한다!
- C 의 의미를 위해 두 예시를 든다

ex1) $(x_i^T x_j + \frac{1}{2})^2$

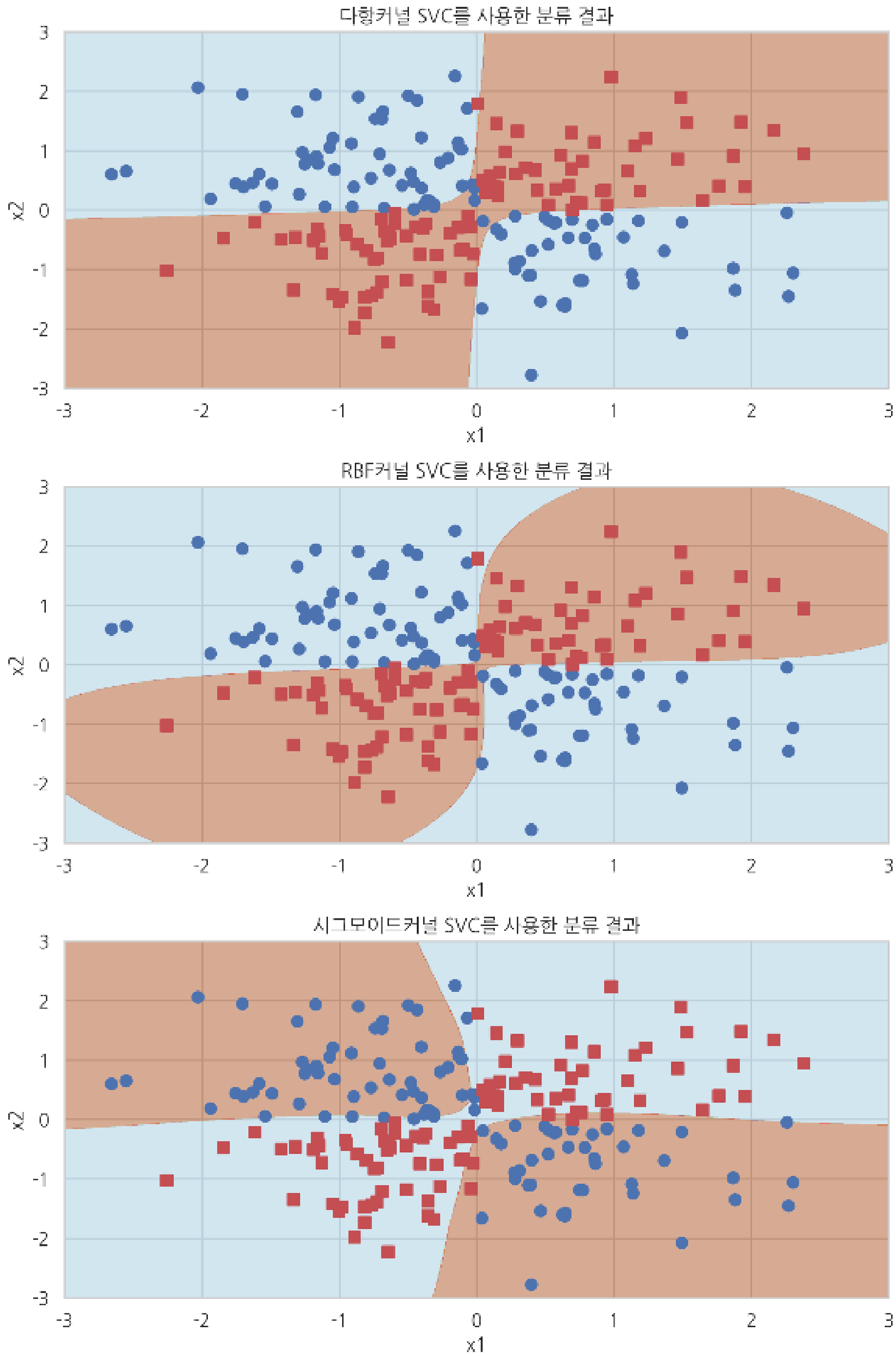
$$=< x_i, x_i^2, \frac{1}{2} >< x_j, x_j^2, \frac{1}{2} >$$

흔히 말하는 Z 축(위로 오른 축) 의1/2는 모든 점들에서 다 똑같다! 정보가 없는 축! ∴ 삭제!

ex2) $(x_i^T x_j + 1)^2$

$$=< \sqrt{2}x_i, x_i^2, 1 >< \sqrt{2}x_j, x_j^2, 1 >$$

- 즉, C 가 큰 경우는 클래스들 간 데이터가 촘촘해서 축을 신축성 있게 늘릴 필요가 있을 때! (고무줄에 비유)


다항커널 SVC를 사용한 분류 결과
x1
x2
RBF커널 SVC를 사용한 분류 결과
x1
x2
시그모이드커널 SVC를 사용한 분류 결과
x1
x2

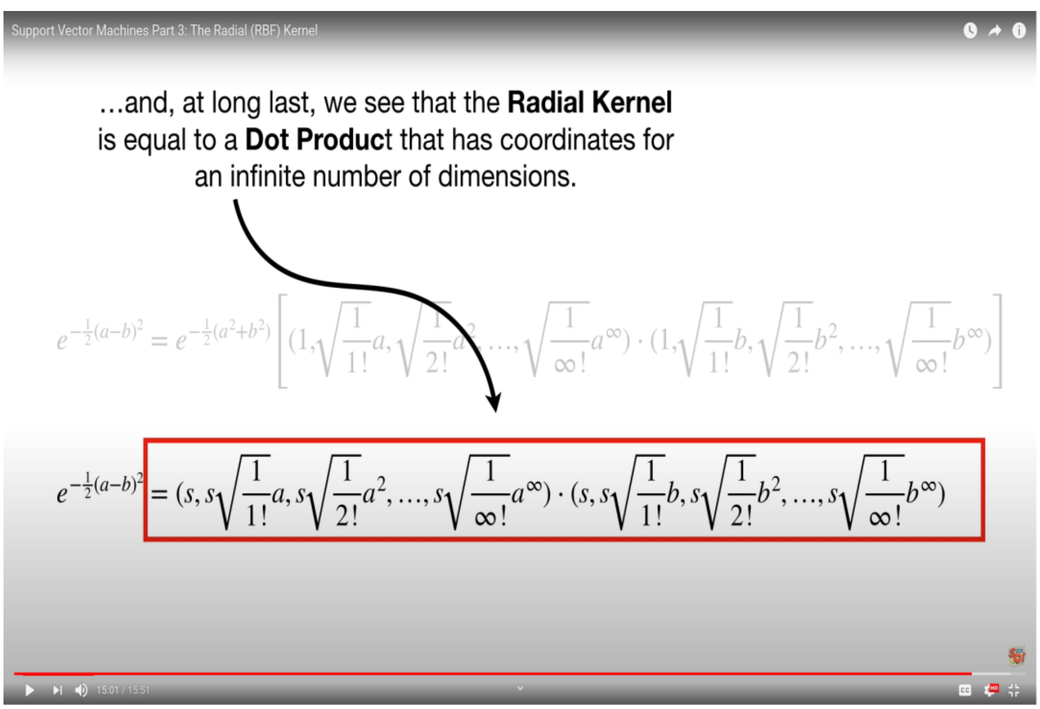
7-2) Radial Kernel

$k(x_i, x_j) = e^{r(x_{ik}-x_{jm})^2}$, r 은 Validation 으로..

Big r : Big Variance

Small r: Big Bias

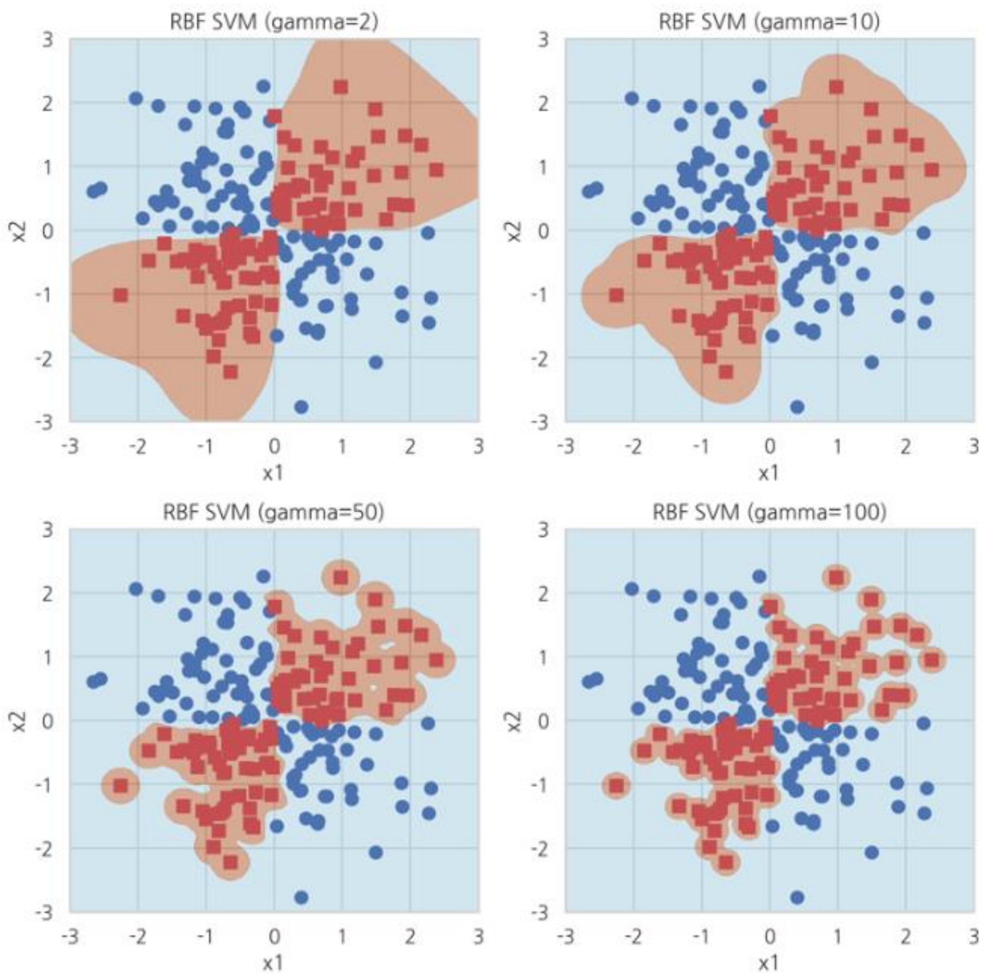
- 중요한 점은 이 커널은 무한차원 확장이 가능하다는 성질!
- 무한차원 확장의 장점? 아무리 굴곡진 Discriminant Function 이 필요한 데이터라도 무한차원에서는 곧은 Hyperplane 을 그을 수 있다 (광활하다고 직관적 이해)



커널 파라미터의 영향

In [10]:

```
plt.figure(figsize=(8, 8))
plt.subplot(221)
plot_xor(X_xor, y_xor, SVC(kernel="rbf", gamma=2).fit(X_xor, y_xor), "RBF SVM (gamma=2)")
plt.subplot(222)
plot_xor(X_xor, y_xor, SVC(kernel="rbf", gamma=10).fit(X_xor, y_xor), "RBF SVM (gamma=10)")
plt.subplot(223)
plot_xor(X_xor, y_xor, SVC(kernel="rbf", gamma=50).fit(X_xor, y_xor), "RBF SVM (gamma=50)")
plt.subplot(224)
plot_xor(X_xor, y_xor, SVC(kernel="rbf", gamma=100).fit(X_xor, y_xor), "RBF SVM (gamma=100)")
plt.tight_layout()
plt.show()
```



and Gaussian, Sigmoid Kernel, etc.

8. Why Sparse Kernel Method?

이를 위해 Kernel Method 의 상위범주인 **Memory Based Model** 을 알아보자!

Linear Model vs Memory Based Model

- **Linear Model** 은 train data-$y(x, w)$모형-test data prediction!
- Train 에 쓰인 데이터는 Test 에 쓰이지 않는다.
- 그러나 **Memory Based Model** 은 Train data 가 Test data 의 Prediction 에 직접 쓰인다!
- **Support Vector Machine 도 memory based model! 그러나 Sparse!**
- Why? **최전선 Support Vector 들만이 Test Data Prediction 에 쓰이므로 Sparse!**