12기 정규세션

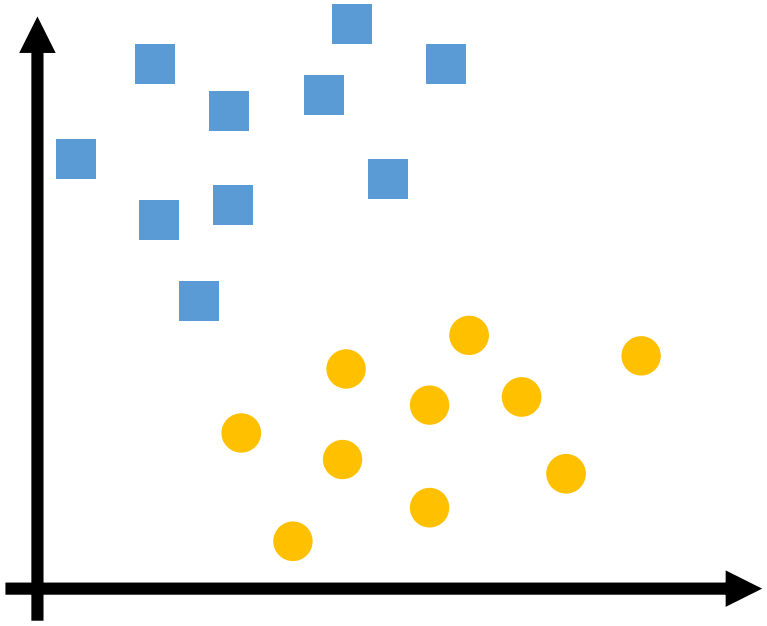
ToBig's 11기 심은선

SVM

Support vector machine

Contents

Classification



■ : 고양이

● : 강아지

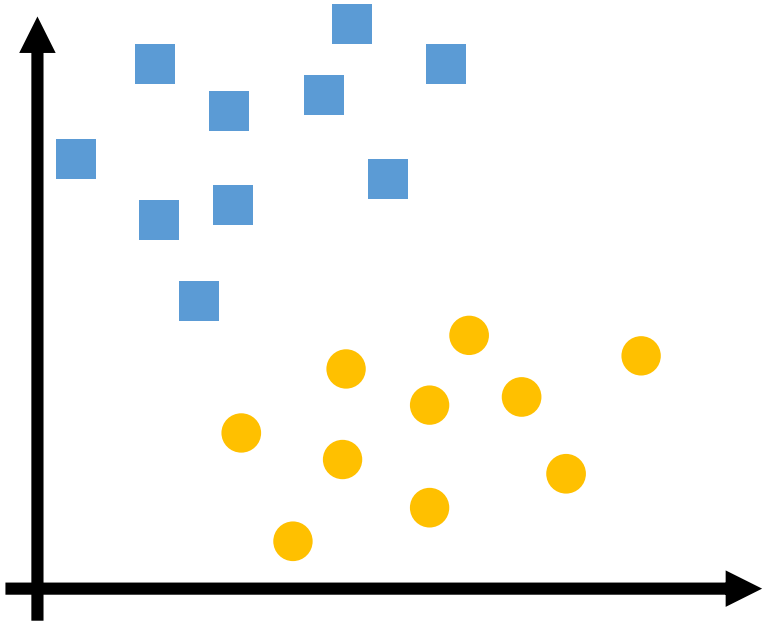
vs.

● : 강아지

■ : 고양이

logistic regression, knn, naïve bayes

Classification



■ : 고양이

● : 강아지

vs.

● : 강아지

■ : 고양이

logistic regression, knn, naïve bayes

+SVM

SVM (support vector machine)

딥러닝 이전에 높은 성능으로 주목받은 모델

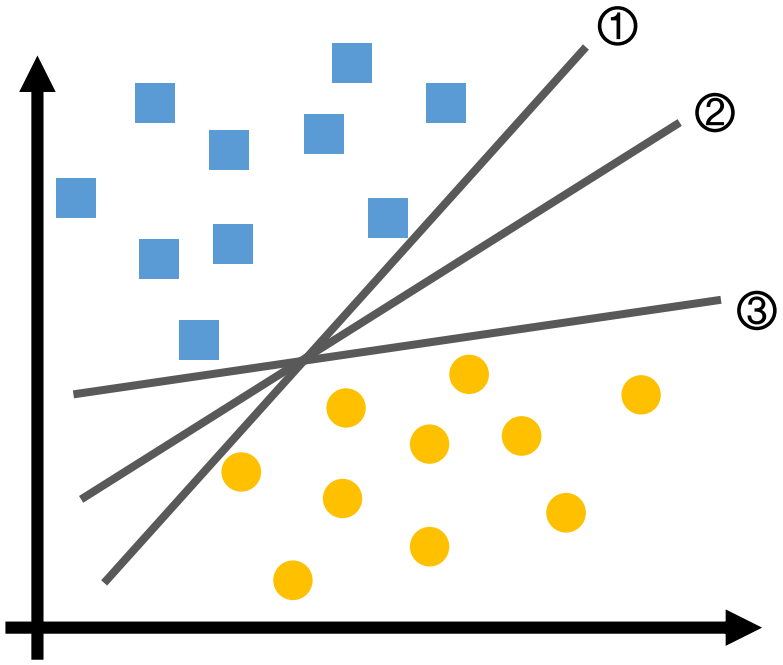
주로 binary classification에 사용되며, 회귀에 사용되는 경우(SVR)도 있다.

SVM의 분류

선형 여부	분류
선형	linear svm
비선형	non-linear svm

오분류 허용	분류
X	hard magin svm
O	soft margin svm

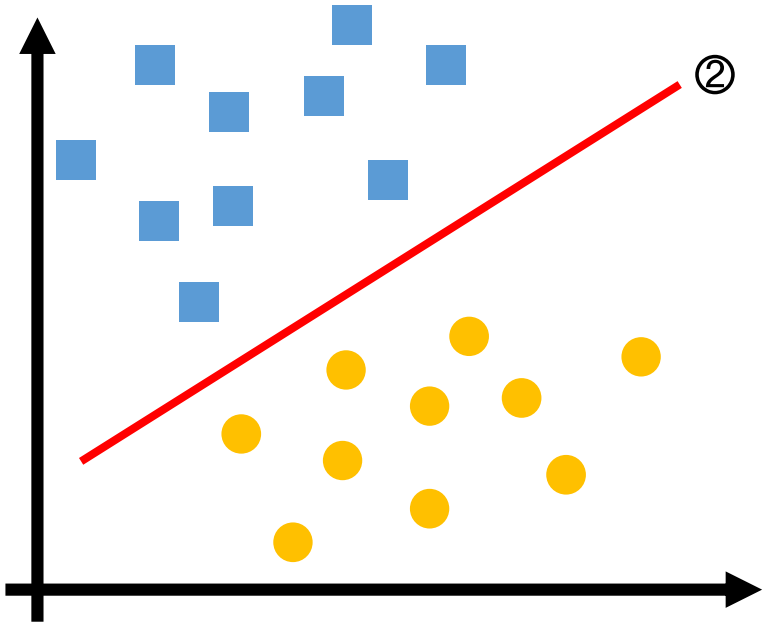
Hard margin SVM



선형 분류기를 찾아보자…

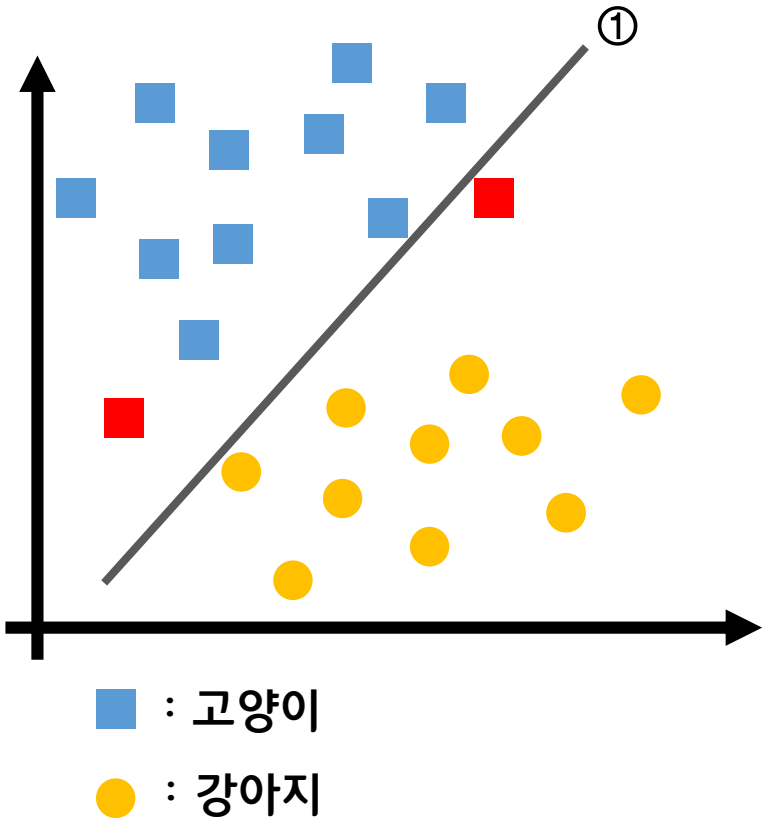
가장 좋은 분류기??

Hard margin SVM



2번 분류기가 가장 좋은 분류기!
왜..?

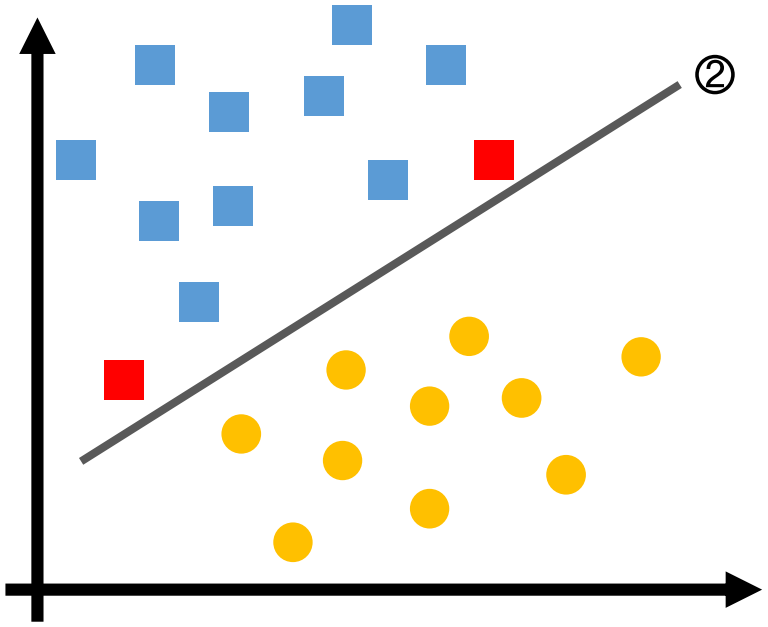
Hard margin SVM



<새로운 데이터가 추가된 경우>

①번 분류기는 데이터 하나를 잘 못 분류함

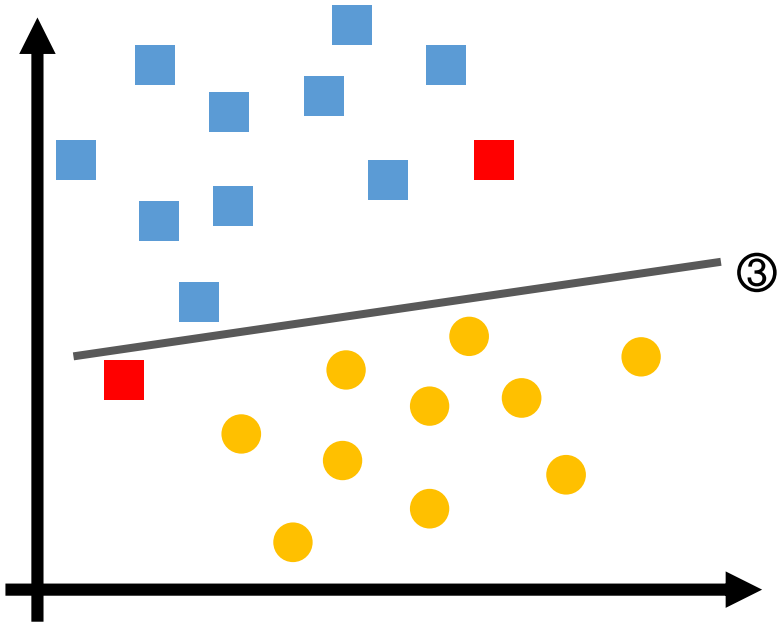
Hard margin SVM



<새로운 데이터가 추가된 경우>

②번 분류기는 모든 데이터를 잘 분류함

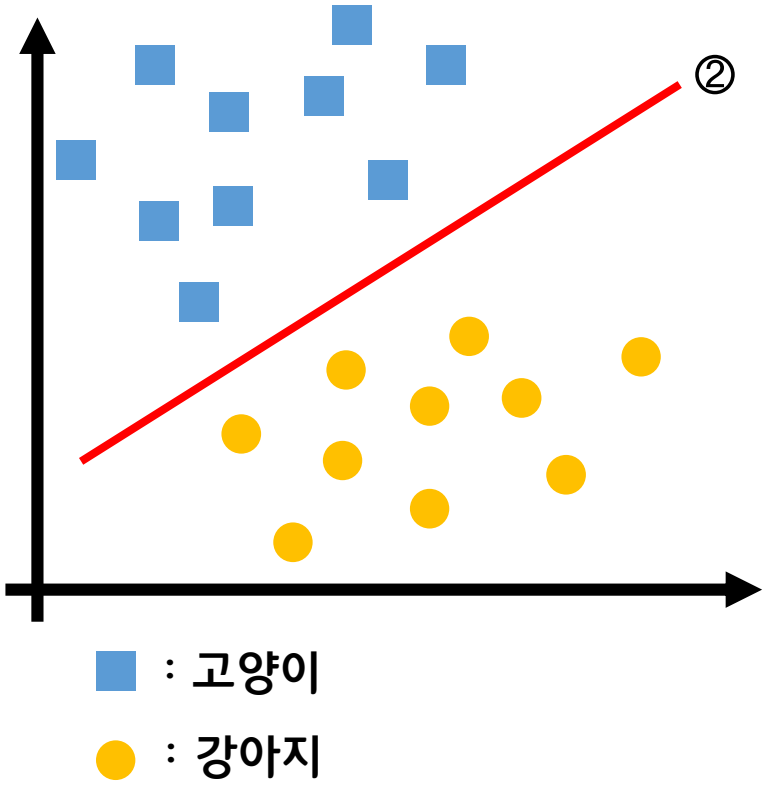
Hard margin SVM



<새로운 데이터가 추가된 경우>

③번 분류기도 데이터 하나를 잘 못 분류함

Hard margin SVM

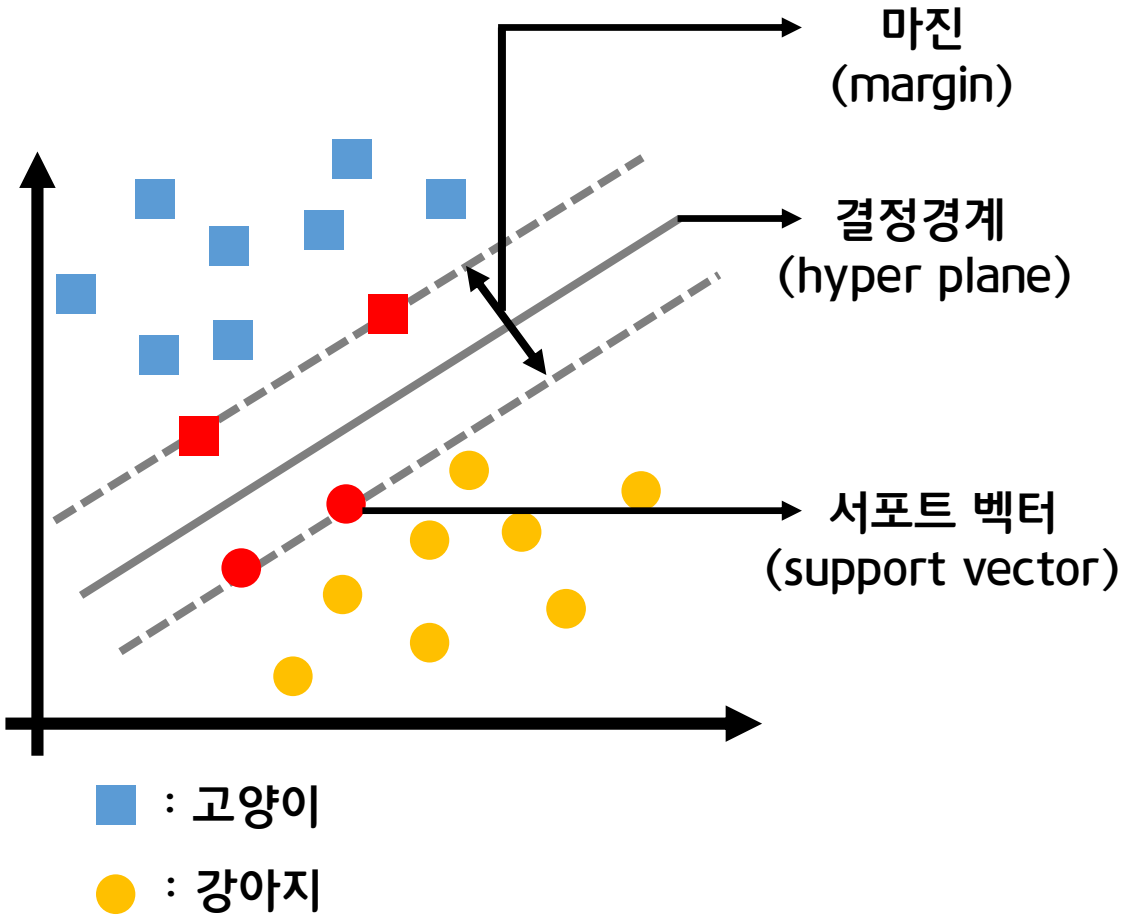


②번이 가장 좋은 분류기인 이유??

분류기와 점(데이터)간의 거리가 최대화되어,
새로운 데이터가 들어와도 일반성을 가지고 잘 분류함

즉, 결정경계(분류기)와 데이터의 거리(여백)이 커서 좋다.
>>여백(마진)을 최대화하는 초평면을 찾자!

Hard margin SVM



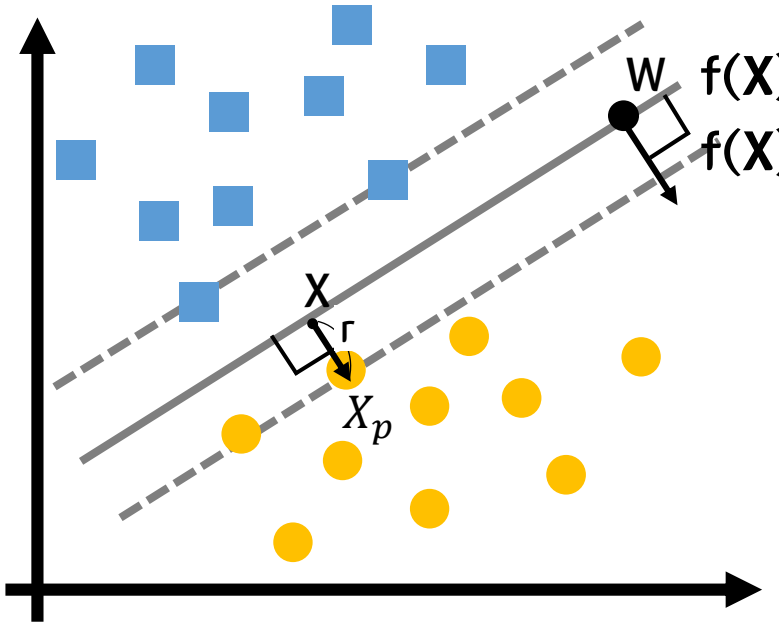
-마진: (결정경계와 서포트 벡터 사이의 거리)x2
-결정경계: 데이터를 가장 잘 분리하는 경계
-서포트 벡터: 결정경계와 거리가 가장 가까운 점

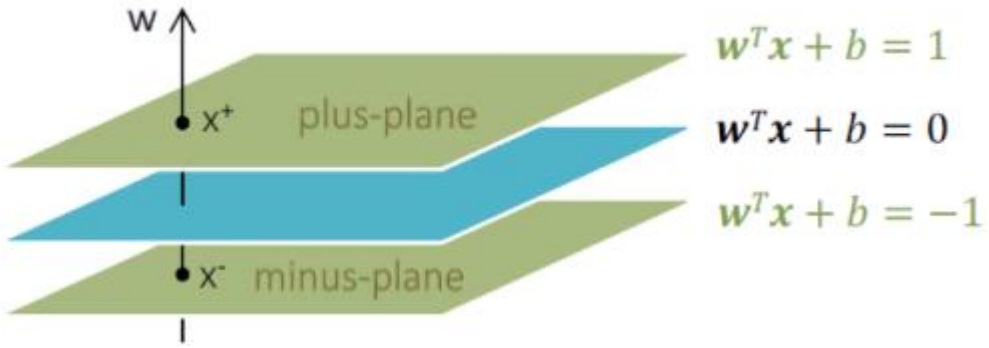
SVM

-margin을 최대화하는 분류기
-support vector로 결정경계가 정해지기 때문에 support vector machine(SVM)이라고 함

목적함수

-margin의 최대화





W는 결정경계와 수직인 법선벡터,

$X_p = \mathrm{X} + r \times \frac{W}{||W||}$ 이고, f(X)=0 이다.

따라서 서포트벡터의 함숫값을 표현하면,

$$\mathrm{f}(X_p) = \mathrm{W} \cdot X_p + \mathrm{b} = \mathrm{W} \cdot \left(\mathrm{X} + r \times \frac{W}{||W||}\right) + \mathrm{b}$$

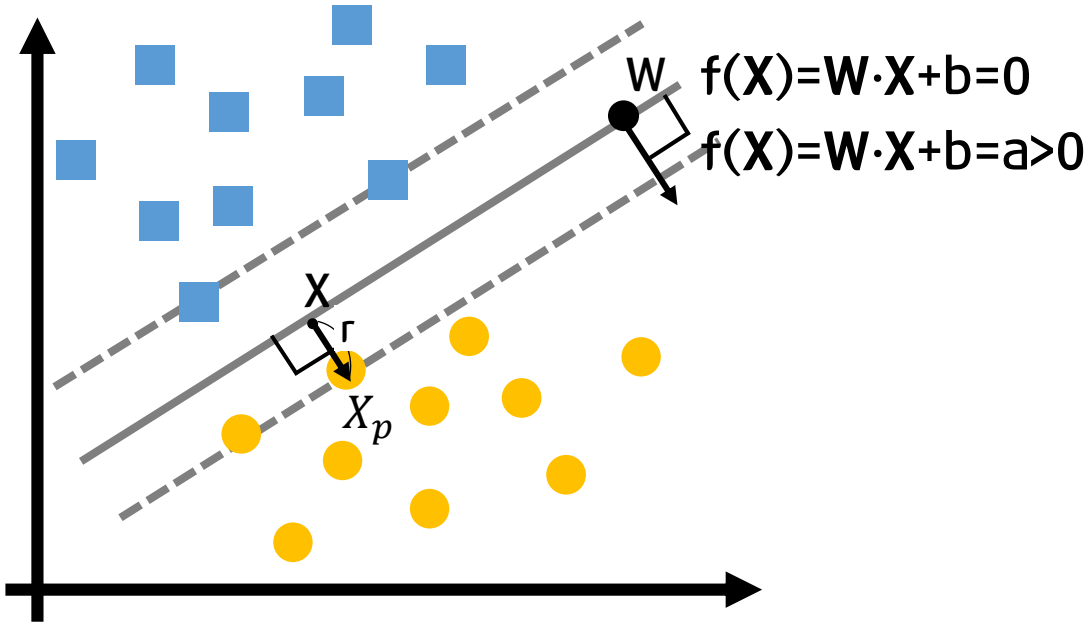
$$= (\mathrm{W} \cdot \mathrm{X} + b) + r \times \frac{W \cdot W}{||W||}$$

$$= 0 + r \times ||W||$$

$$= r||W||$$

목적함수

-margin의 최대화



$f(X_p) = r||W||$ = a이므로, margin = 2r = $\frac{2a}{||W||}$

이때, 실제 값을 y_p라고 하면 $(WX_p+b)y_p \geq a$ 이다.

SVM의 목적함수

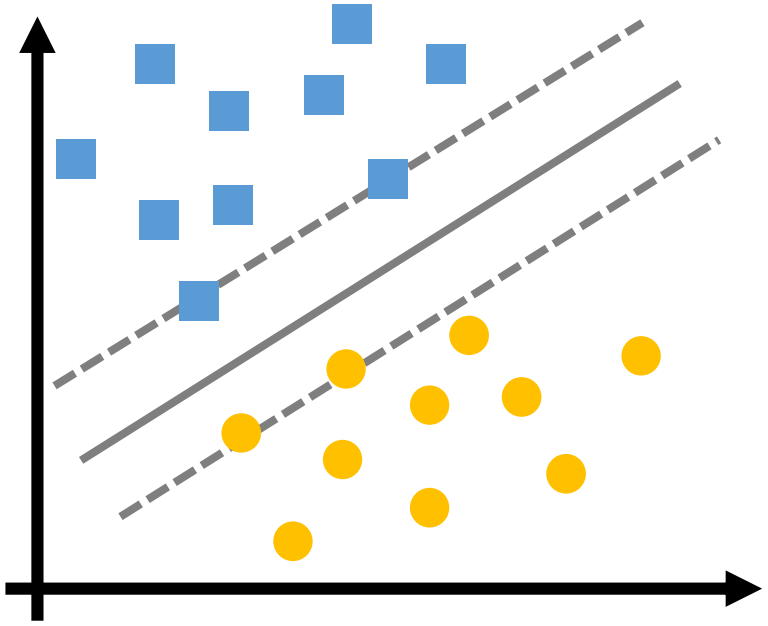
- argmax $\frac{2a}{||W||}$ = argmax $\frac{1}{||W||}$ (a가 임의의 양수이므로 스케일링)

= $\underset{W,b}{\text{argmin}} ||W||$

$s.t.\ (WX_p+b)y_p \geq a$

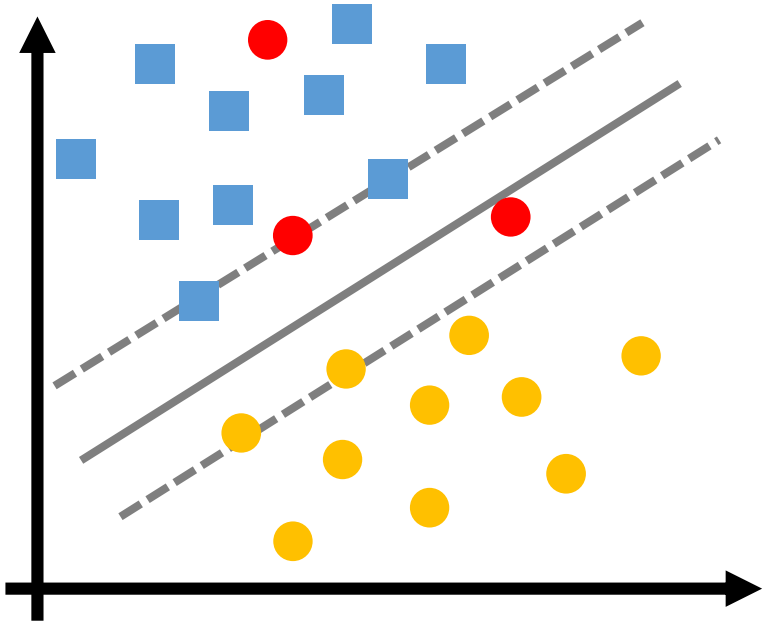
제약식 내에서 목적함수를 최소화함(Quadratic programming)
(라그랑주 승수법, dual problem, KKT condition… 등
너무 어려워서 구체적인 수식 생략)

Soft margin svm



Hard margin svm
: error를 허용하지 않음

Soft margin svm



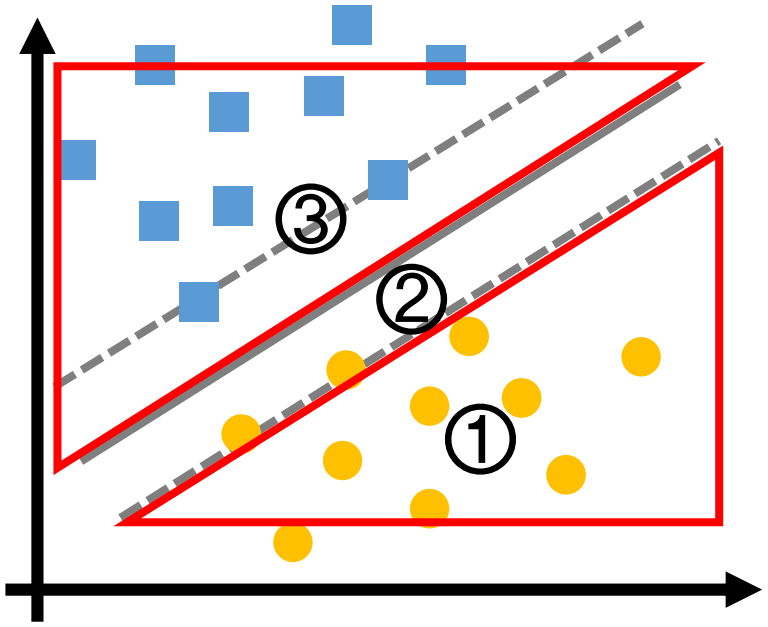
Hard margin svm

: error를 허용하지 않음

하지만 이렇게 hard margin으로 분류할 수 없는 경우에...

1) non-linear model
2) error를 허용하되 error를 작게하자
   ->soft margin svm

Soft margin svm



Soft margin svm

-error(오분류)를 허용하지만, 전체 error를 최소화

-오분류 정도에 따라 error의 크기가 달라야함

ex) 실제로 ●인 데이터가 분류기의

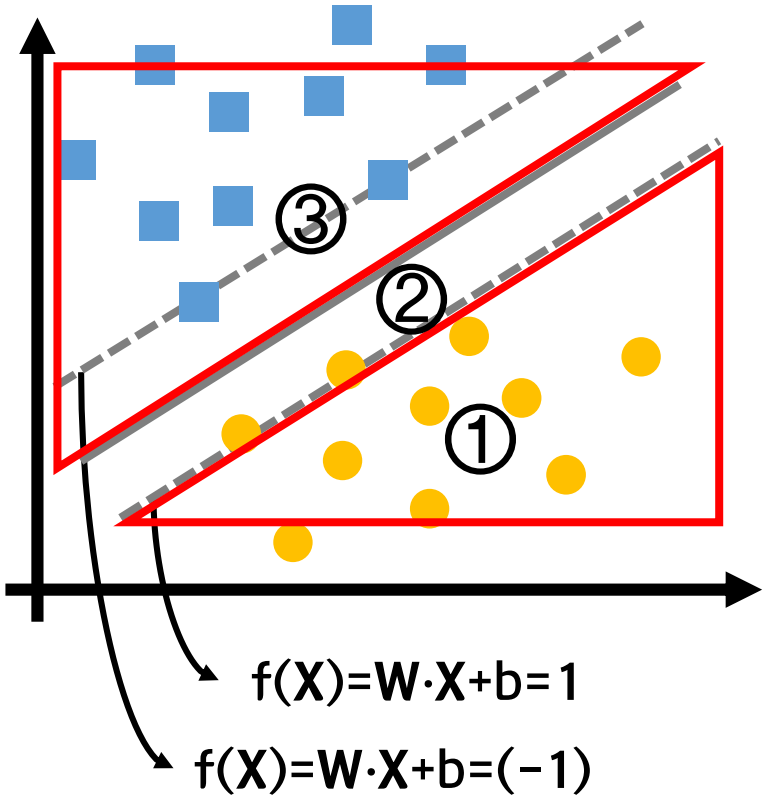
-①번에 있는 경우: 좋은 분류 -> error없음

-②번에 있는 경우: 작은 error

-③번에 있는 경우: 큰 error (잘못 분류)

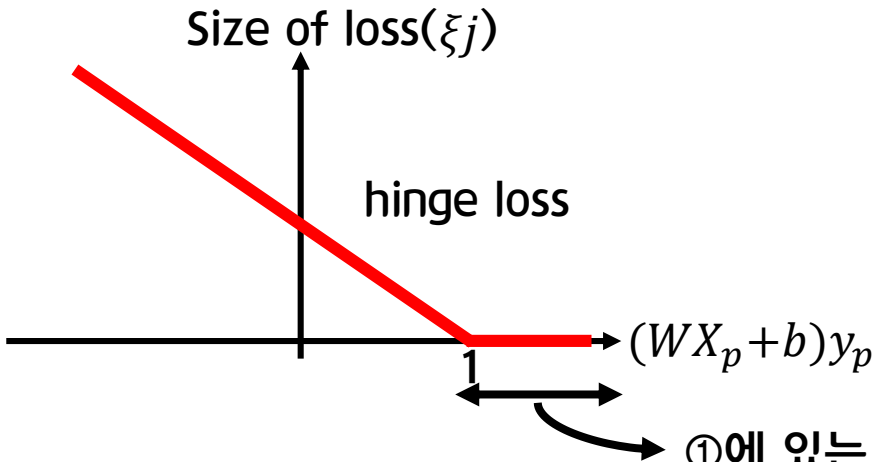
-> slack variable(ξ_j) 사용

Soft margin svm



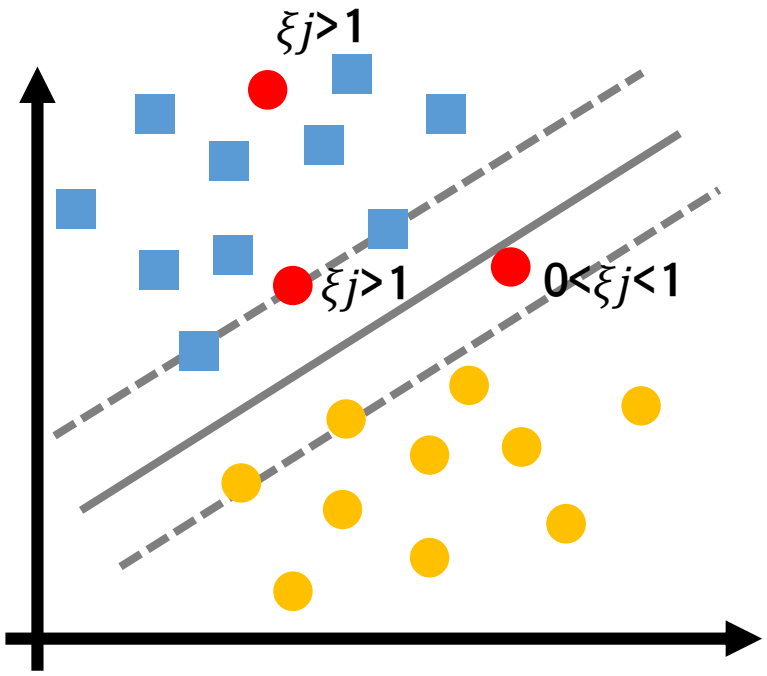
Slack variable(ξj)

● 데이터가 분류기의
-①번에 있는 경우: $(WX_p+b)y_p \geq 1$ -> $\xi j=0$
-②번에 있는 경우: $0 \leq (WX_p+b)y_p < 1$ -> $0 < \xi j \leq 1$
-③번에 있는 경우: $(WX_p+b)y_p < 0$ -> $\xi j > 1$
-> hinge loss



①에 있는 ● 데이터(잘 분류된 데이터)

Soft margin svm



Soft margin SVM

-기존 목적함수에 error항을 추가

-$\underset{W,b}{\text{argmin}} \left|\left|W\right|\right| + C\sum\xi j$

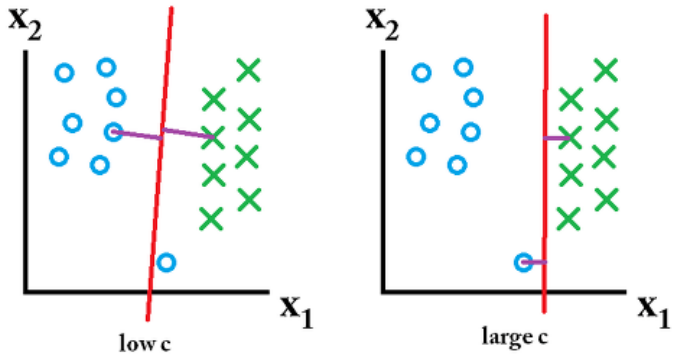


그림6. 매개변수 C의 영향

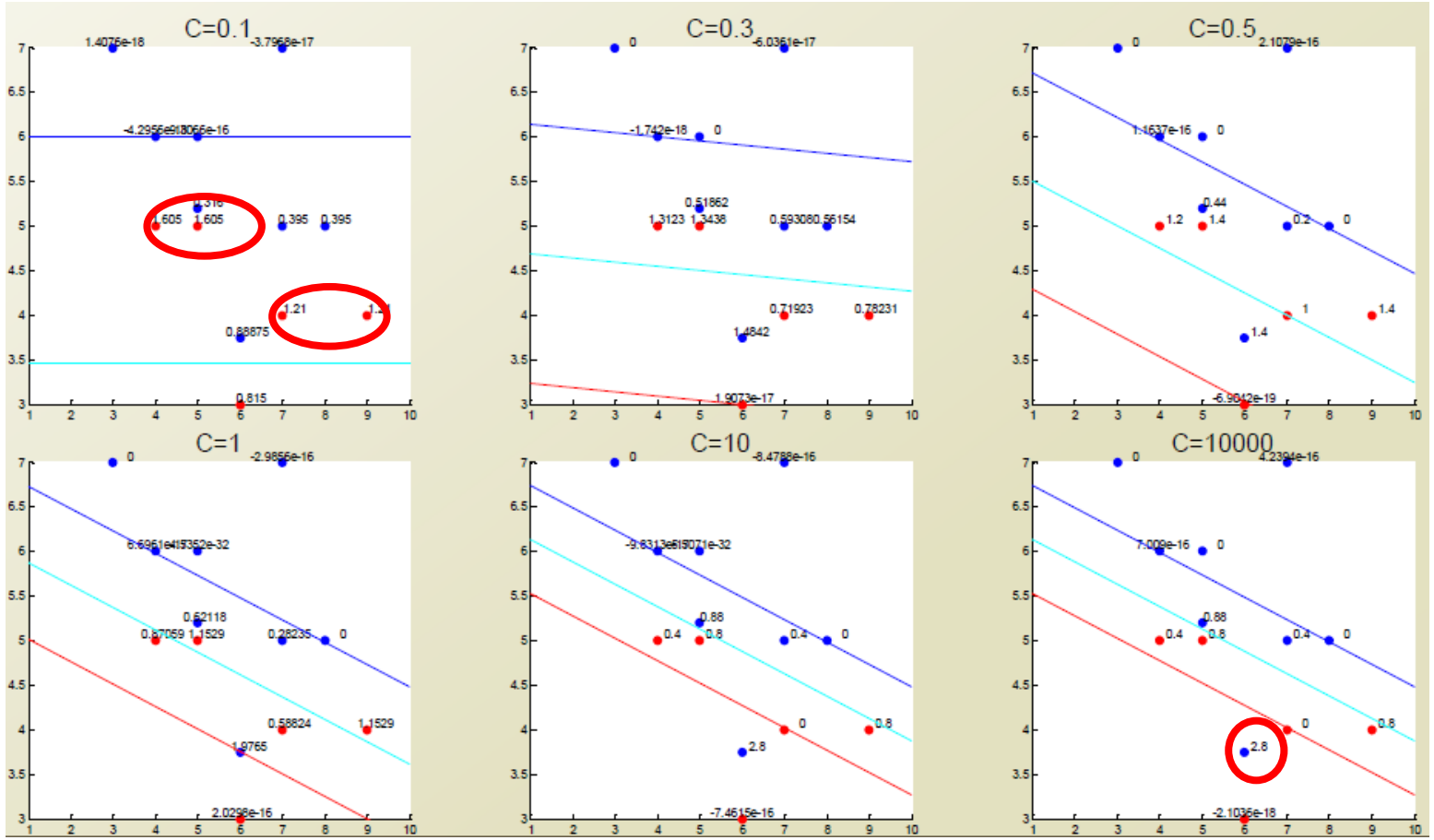
하이퍼파라미터 C

:오분류에 패널티를 얼마나 줄건지 결정
(다른 클래스에 놓이는 걸 허용하는 정도)

-작은C: 패널티가 작아 error를 상대적으로 많이 허용함
-큰 C: 패널티를 크게 주니까 error를 상대적으로 적게 허용함

Unit 03 | Soft margin SVM

Soft margin svm

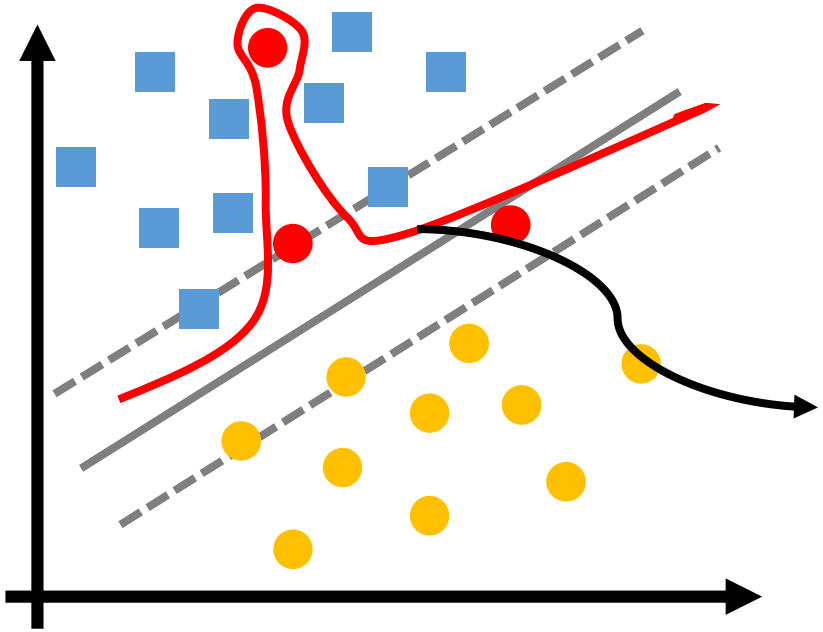


하이퍼파라미터 C

C가 작으면 분류 error를 많이 허용
-c=0.1이면 제대로 분류하지 못함
C가 크면 분류 error를 적게 허용
-c=10000이면 error가 하나만 있음

C의 유무에 따라
-C있음: soft margin svm
-C없음: hard margin svm

Non-linear SVM



Hard margin svm
: error를 허용하지 않음

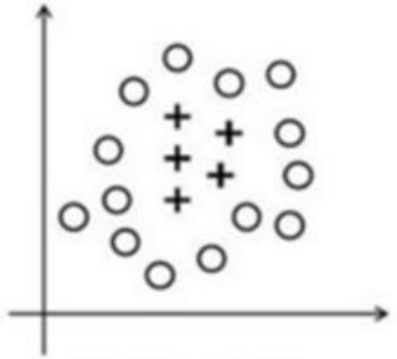
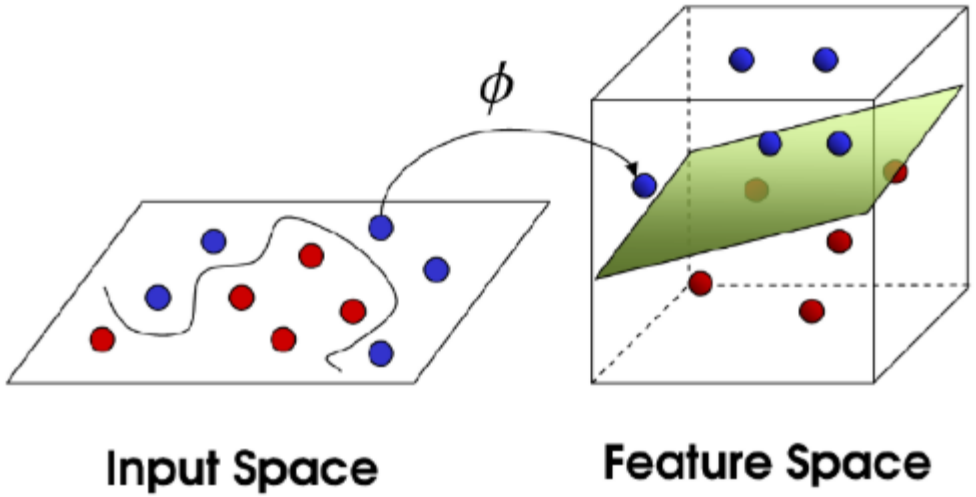


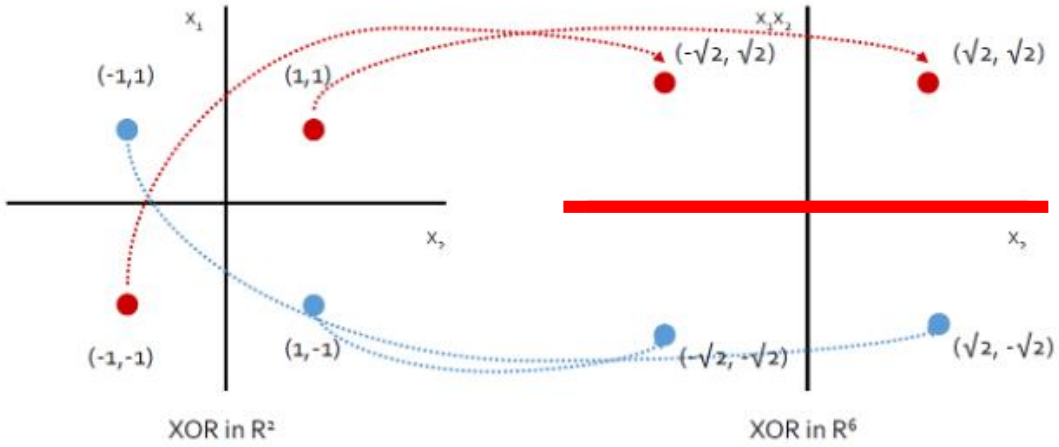
그림7. 선형 SVM으로는 제대로 분류할 수 없는 상황

하지만 이렇게 hard margin으로 분류할 수 없는 경우에...
1) non-linear model –kernel svm
2) Soft margin svm

Kernel SVM



-Input 차원을 고차원으로 mapping한 뒤 선형분류
-고차원에서는 선형분류이나, 원래 차원에서는
비선형으로 보임



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

위와 같은 mapping함수를 사용해
선형 분류가 불가능한 2차원에서,
선형 분류가 가능한 6차원이 된다!

Kernel SVM

목적함수를 최적화하는 수식에 내적이 있다

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

고차원으로 mapping시 내적 부분의 차원이 증가!

>> 큰 차원을 내적해야해서 연산량이 매우매우 많아진다.

>> Kernel사용으로 **연산량의 문제를 해결**

(kernel함수는 mapping과 내적의 순서를 바꿀 수 있음)

Polynomial Kernel Function of degree 2

- $K(<x_1, x_2>, <z_1, z_2>) = <x_1^2, \sqrt{2}x_1x_2, x_2^2> \cdot <z_1^2, \sqrt{2}z_1z_2, z_2^2>$ → mapping후 내적: 연산이 너무 많음
- $= x_1^2 z_1^2 + 2x_1x_2z_1z_2 + x_2^2 z_2^2 = (x_1z_1 + x_2z_2)^2 = (\mathbf{x} \cdot \mathbf{z})^2$ → 내적후 mapping: 연산이 간단함

Kernel SVM

Kernel사용 이유??

-고차원으로 mapping하되, 연산을 간단히 하기위해!

kernel의 종류

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

-가장 많이 사용하는 kernel
-Radial Basis Function kernel이라고도 하고, 줄여서 RBF kernel이라고 함

RBF kernel(=Gaussian kernel)

-무한대의 차원으로 mapping하는 kernel

$$K(x_1, x_2) = exp\left\{-(x_1 - x_2)^2\right\}$$
$$= exp(-x_1)exp(-x_2)exp(2x_1x_2) \xrightarrow{\text{테일러 급수}} exp(2x_1x_2) = \sum_{k=0}^{\infty} \frac{2^k x_1^k x_2^k}{k!}$$

-RBF kernel의 하이퍼파라미터 γ

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

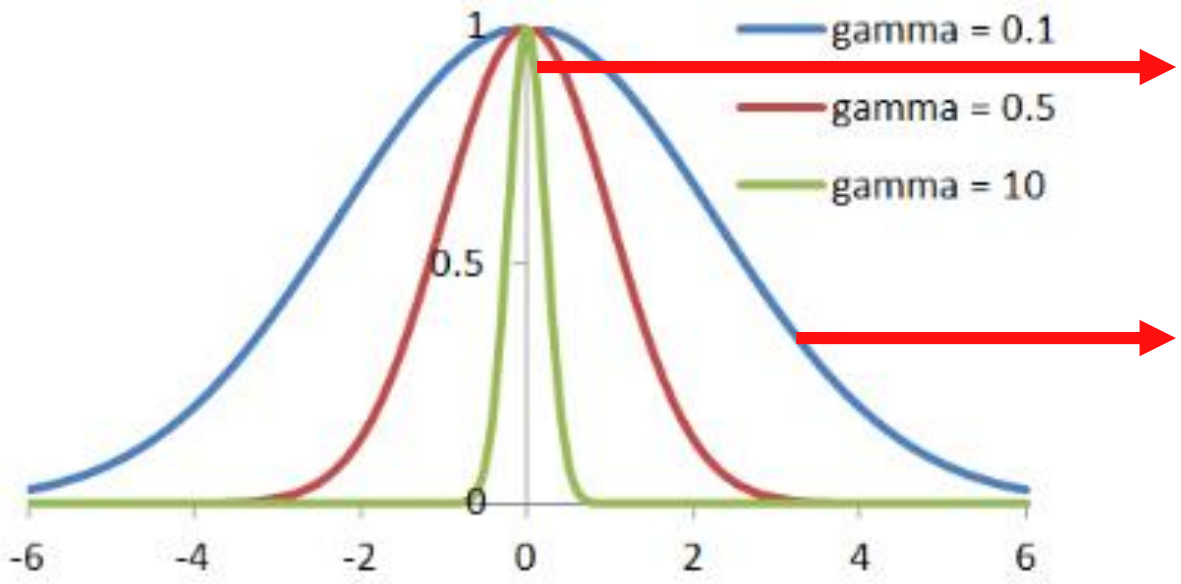
$\gamma \left(= \frac{1}{2\sigma^2}\right)$: 분산(표준편차)와 반비례 관계

감마가 클수록 표준편차는 작고, 감마가 작을수록 표준편차는 크다

RBF kernel(=Gaussian kernel)

$$exp\left\{-\frac{\|x_1 - x_2\|_2^2}{2\sigma^2}\right\}$$

-RBF kernel의 하이퍼파라미터 γ

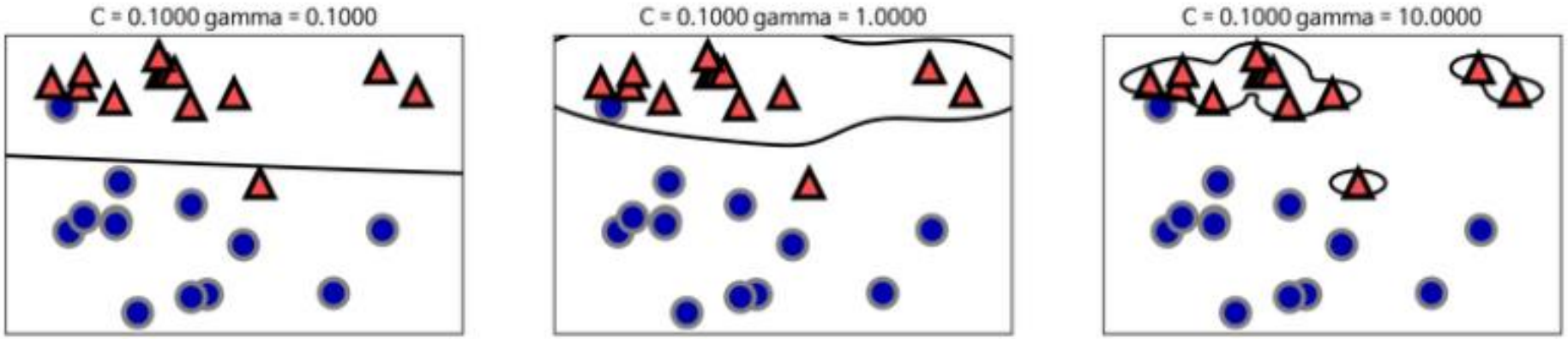


<표준편차가 작은 경우>
작은 표준편차=큰 감마
>>폭이 좁은 분포
>>아주 인접한 값만 같은 라벨로 분류할 수 있음

<표준편차가 큰 경우>
큰 표준편차=작은 감마
>>폭이 넓은 분포
>>어느정도 인접하면 같은 라벨로 분류함

RBF kernel(=Gaussian kernel)

-RBF kernel의 하이퍼파라미터 γ



감마가 커질수록(=표준편차가 작을수록) 아주 인접한 데이터만 같은 라벨로 분류한다

따라서 감마가 클수록 결정경계가 구불구불하고, 작을수록 부드러운 결정경계가 만들어진다

>>감마는 결정경계의 곡률을 결정

Unit 05 | Summary

Hyperparameter

1. C –soft margin SVM
   - error term에 패널티를 얼마나 줄건지 결정
     (샘플이 다른 클래스에 놓이는 걸 허용하는 정도)
   - C가 클수록 모델의 오분류에 패널티를 크게 줌

$$\underset{W,b}{\operatorname{argmin}} \left|\left|W\right|\right| + C\sum \xi j$$

2. γ –RBF kernel SVM
   - 얼마나 인접한 값까지 같은 라벨로 분류하는지 결정
   - 감마가 클수록(=표준편차가 작을수록) 아주 인접한 값만 같은 라벨로 분류

$$k_{rbf}(x_1, x_2) = exp(-\gamma||x_1 - x_2||^2)$$

Unit 05 | Summary

Hyperparameter

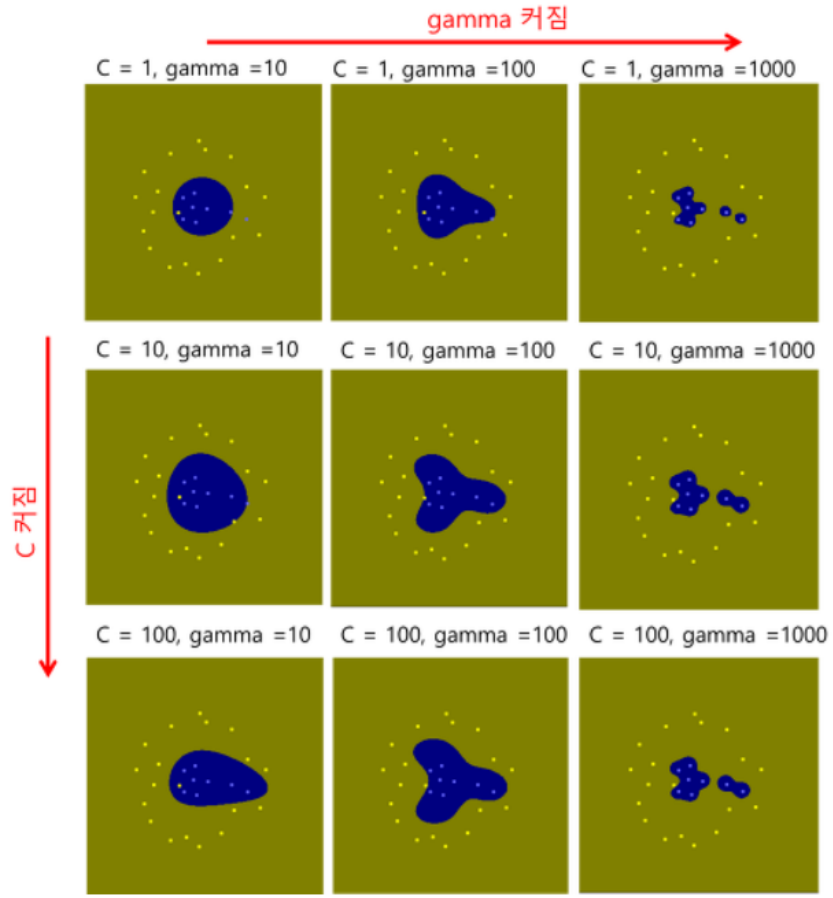


그림9. C와 gamma의 영향

C: 데이터가 다른 클래스에 놓이는 걸 허용하는 정도
γ: 결정경계의 곡률을 결정(kernel 모델의 파라미터)

C	γ
-큰 C: 큰 패널티 -> error 허용x >> low bias, high variance >> overfitting 위험 -작은 C: 작은 패널티 -> error 다수 >> high bias, low variance >> underfitting 위험	-큰 γ: 구불구불한 결정경계 >> low bias, high variance >> overfitting 위험 -작은 γ: 부드러운 결정경계 >> high bias, low variance >> underfitting 위험

Unit 05 | Summary

Hyperparameter

sklearn.svm.SVC

class sklearn.svm. SVC (*C=1.0, kernel='rbf', degree=3, gamma='auto_deprecated', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None*) [source]

C-Support Vector Classification.

The implementation is based on libsvm. The fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples. For large datasets consider using `sklearn.linear_model.LinearSVC` or `sklearn.linear_model.SGDClassifier` instead, possibly after a `sklearn.kernel_approximation.Nystroem` transformer.

The multiclass support is handled according to a one-vs-one scheme.

For details on the precise mathematical formulation of the provided kernel functions and how `gamma`, `coef0` and `degree` affect each other, see the corresponding section in the narrative documentation: Kernel functions.

Read more in the User Guide.

Parameters:

C : *float, optional (default=1.0)*
Penalty parameter C of the error term.

kernel : *string, optional (default='rbf')*
Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`.

degree : *int, optional (default=3)*
Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

gamma : *float, optional (default='auto')*
Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

Kernel의 종류에서 linear은 kernel을 사용하지 않는 linear SVM을 말함

앤드류 응 교수님께서
피쳐 개수가 데이터 개수에 비해 많으면 linear kernel을 사용하고 (피쳐개수 10000개정도)
피쳐가 적고 데이터 개수는 보통이면 RBF커널 SVM을 쓰라고 하셨다(피쳐 1000개, 데이터 10~10000개 정도)
피쳐가 적고 데이터가 많으면 linear SVM이나 로지스틱 회귀를 쓰라고 하셨으나, 많은 연산량이 부담이 안되면 RBF kernel SVM도 괜찮다
(교수님의 개인적인 생각입니다)

Assignment

'Santander Customer Transaction Prediction' 데이터 분석

- EDA, feature engineering (변수선택, scaling 등)
  (생략해도 되고, Kaggle kernel 참고해도 됨)
- PCA를 적용해서 차원을 축소
  (차원축소의 성능이 안좋으면 원래 데이터 사용가능)
- SVM 학습, 하이퍼파라미터 튜닝
- Kaggle에 제출해서 test accuracy 캡쳐

(https://www.kaggle.com/c/santander-customer-transaction-prediction/kernels)

Reference

투빅스 10기 박규리님 svm강의

(http://www.datamarket.kr/xe/index.php?mid=board_jPWY12&page=2&document_srl=51056)

투빅스 8기 김은서님 svm강의

(http://www.datamarket.kr/xe/index.php?mid=board_jPWY12&page=3&document_srl=45884)

하이퍼파마리터 자세한 설명

(https://bskyvision.com/163)

(https://tensorflow.blog/%ED%8C%8C%EC%9D%B4%EC%8D%AC-%EB%A8%B8%EC%8B%A0%EB%9F%AC%EB%8B%9D/2-3-7-%EC%BB%A4%EB%84%90-%EC%84%9C%ED%8F%AC%ED%8A%B8-%EB%B2%A1%ED%84%B0-%EB%A8%B8%EC%8B%A0/#4)

커널 설명

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

들어주셔서 감사합니다.