SVM(Support Vecotor Machine)

: 주어진 데이터 집합을 바탕으로 새로운 데이터가 두 개의 카테고리 중 어디에 속할지 판단하는 비확률적 이진 선형 분류 모델

패턴인식, 자료 분석을 위한 지도 학습 모델

분류와 회귀 문제에 사용(주로 분류)

커널 트릭(Kernel Trick)을 활용하여 비선형 분류 문제도 사용 가능

학습 방향: 마진(margin)의 최대화

결정 경계(hyperplane)

: 서로 다른 클래스를 분류하는 기준

→ 주변 데이터와 거리가 최대

→ 결정 경계 근처에 새로운 데이터 들어와도 강인한 분류할 수 있다

데이터 임베딩 공간보다 1차원 낮은 부분 공간

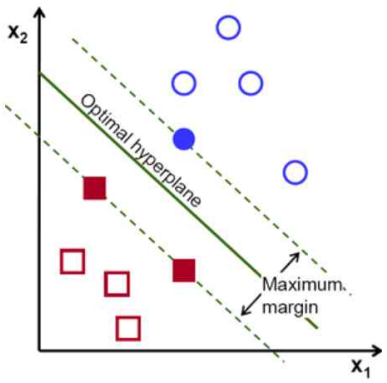
2차원 데이터 공간 → 직선, 3차원 → 평면, 4차원 이상 → 초평면

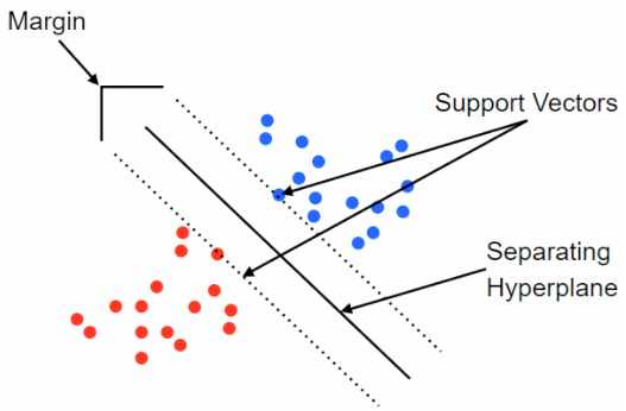
서포트 벡터(support vector)

: hyperplane에 가장 가까이 있는 각 클래스의 데이터

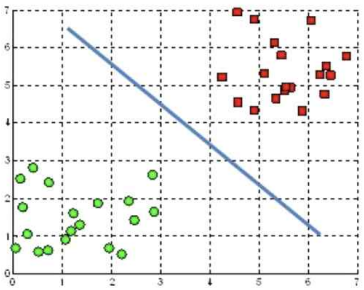
마진(margin)

: 어떤 데이터도 포함하지 않는 영역(서포트 벡터와 직교하는 직선과의 거리)

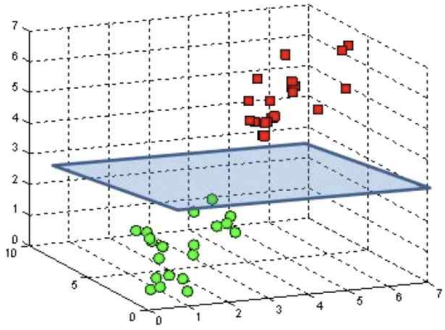




A hyperplane in $\mathbb{R}^2$ is a line



A hyperplane in $\mathbb{R}^3$ is a plane



Linear SVM

1. Hard Margin SVM

결정 경계(초평면): $W^TX+b=0$

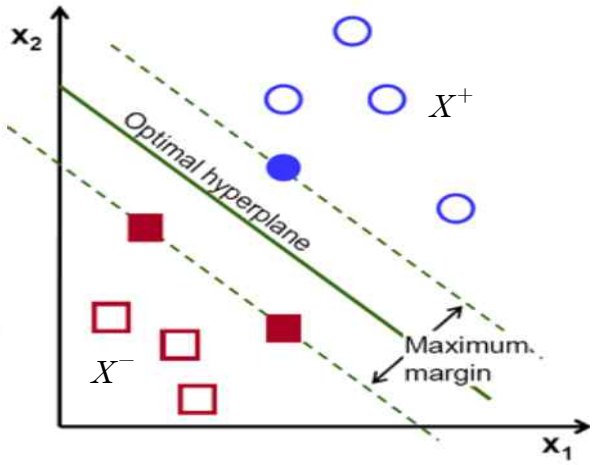
(파랑) 서포트 벡터를 지나는 초평면: $W^TX+b=1$, **데이터:** $W^TX+b\geq 1$

(빨강) 서포트 벡터를 지나는 초평면: $W^TX+b=-1$, **데이터:** $W^TX+b\leq -1$

초평면의 법선 벡터: W^T

★ 서로 다른 클래스 데이터의 관계: $X^+=X^-+\lambda W$

$$W^T+b=1, X^+=X^-+\lambda W$$
$$W^T(X^-+\lambda W)+b=1$$
$$(W^TX^-+b)+\lambda W^TW=1$$
$$-1+\lambda W^TW=1$$
$$\lambda=\frac{2}{W^TW}$$



$$Margin = distance(X^+, X^-)$$
$$= \| X^+-X^- \|_2$$
$$= \| (X^-+\lambda W)-X^- \|_2$$
$$= \| \lambda W \|_2$$
$$= \lambda\sqrt{W^TW}$$
$$= \frac{2}{W^TW}\sqrt{W^TW}$$
$$= \frac{2}{\sqrt{W^TW}} = \frac{2}{\| W \|_2}$$

$$L_2=\sqrt{\sum_i^n x_i^2}=\sqrt{x_1^2+x_2^2+\cdots+x_n^2}$$

목적 함수: 마진의 최대화

$$\max Margin = \max\frac{2}{\| W \|_2}$$
$$\max\frac{2}{\| W \|_2} = \min\frac{\| W \|_2}{2}$$ (역수 : 최대화 문제를 최소화 문제로 변경한다)
$$\min\frac{\| W \|_2}{2} \approx \min\frac{\| w \|_2^2}{2}$$ (계산 상의 편의)

Original Problem: $\min_{w,b}\frac{1}{2}\| w \|_2^2$

$$subjected\ to\ y_i(w^Tx_i+b)\geq 1, i=1,2,...,n$$

라그랑지 승수(Lagrange Multiplier)로 해결하기

1. 제약식(constraints을 목적식(objective function)에 포함하기

Primial Problem: $\max_{\alpha}\min_{w,b} L(w,b,\alpha)=\frac{1}{2}\parallel w\parallel_2^2-\sum_{i=1}^{n}\alpha_i(y_i(w^Tx_i+b)-1)$

$subjected\,to\,\alpha_i\geq 0, i-1,2,...,n$

2. Primial Problem을 Dual Problem으로 변경하기(w는 α, x, y로 변경)

Dual Problem: $\max_{\alpha}\sum_{i=1}^{n}\alpha_i-\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j$

$subjected\,to\sum_{i=1}^{n}\alpha_i y_i=0, \alpha_i\geq 0, i-1,2,...,n$

KKT(Karush-Kuhn-Tucker) conditions

: (w,b,α)가 Lagrandian dual problem의 최적해가 되기 위한 조건

Stationarity: $w=\sum_{i=1}^{n}\alpha_i y_i x_i,\ \sum_{i=1}^{n}\alpha_i y_i=0$

Primal feasibility: $y_i(w^Tx_i+b)\geq 1,\ i=1,2,...,n$

Dual feasibility: $\alpha_i\geq 0, i-1,2,...,n$

★ Complementary sslackness: $\alpha_i(y_i(w^Tx_i+b)-1)=0$

1. $\alpha_i>0, y_i(w^Tx_i+b)-1=0$: x_i가 서포트 벡터인 경우
2. $\alpha_i=0, y_i(w^Tx_i+b)-1\neq 0$: x_i가 서포트 벡터 아니다

hyperlane 구축에 영향X

⇒ SVM이 outlier에 강인한 이유

결정 경계 결정법: 서포트 벡터를 가지고 계산

$$W=\sum_{i=1}^{n}\alpha_i y_i x_i=\sum_{i\in SV}\alpha_i y_i x_i$$

$$b=y_{sv}-\sum_{i=1}^{n}\alpha_i y_i x_i^T x_{sv}$$

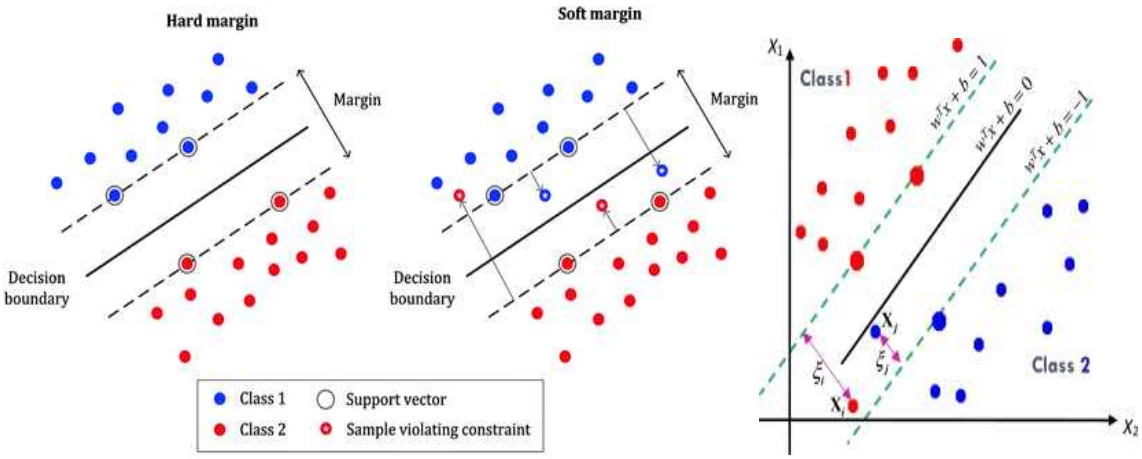
$$y_{new}=sign(W^TX_{new}+b)\qquad \begin{matrix} W^TX_{new}+b>0:1 \\ W^TX_{new}+b<0:-1 \end{matrix}$$

2. Soft Margin SVM

Hard Margin SVM → 선형 분리 가능한 문제

Soft Margin SVM → 선형 분리 불가능한 문제

에러 허용하자(에러=0, 완벽히 나누기 불가능 인정)



목적 함수: 에러(penalty term)을 허용하면서 마진 최대화

Original Problem: $\min_{w,b,\xi} \frac{1}{2} \| w \|_2^2 + C\sum_{i=1}^{n} \xi_i$

$$subjected\,to\, y_i(w^T x_i + b) \geq 1 - \xi_i, \xi \geq 0, i = 1, 2, ..., n$$

Penalty term: $C\sum_{i=1}^{n} \xi_i$

C를 통해 에러 허용 정도 조절한다

1. C가 크면 학습에러를 상대적으로 허용하지 않는다

→ 마진이 작아진다 → Overfitting

2. C가 작으면 학습 에러를 상대적으로 허용한다

→ 마진이 커진다 → Underfitting

라그랑지 승수(Lagrange Multiplier)로 해결하기

1. 제약식(constraints을 목적식(objective function)에 포함하기

Primial Problem:

$$\max_{\alpha} \min_{w,b} L(w, b, \alpha, \xi, \gamma) = \frac{1}{2} \| w \|_2^2 - \sum_{i=1}^{n} \alpha_i (y_i(w^T x_i + b) - 1 + \xi_i) - \sum_{i=1}^{n} \gamma_i \xi_i$$

$$subjected\,to\, \alpha_i \gamma_i \geq 0, i - 1, 2, ..., n$$

2. Primial Problem을 Dual Problem으로 변경하기(w는 α, x, y로 변경)

Dual Problem: $\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$

$subjected\,to \sum_{i=1}^{n} \alpha_i y_i = 0, 0 \le \alpha_i \le C, i-1,2,...,n$

주의 사항: $0 \le \alpha_i \le C, from\, \alpha_i \ge 0, \gamma_i \ge 0, C-\alpha_i-\gamma_i = 0$

Hard Margin SVM	Soft Margin SVM
$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$ $subjected\,to \sum_{i=1}^{n} \alpha_i y_i = 0,$ $\alpha_i \ge 0, i-1,2,...,n$	$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$ $subjected\,to \sum_{i=1}^{n} \alpha_i y_i = 0,$ $0 \le \alpha_i \le C, i-1,2,...,n$
선형 분류의 두 케이스 모두 quadratic programming을 풀어 α를 구한다 $\alpha \rightarrow w, b \rightarrow$ 모델	

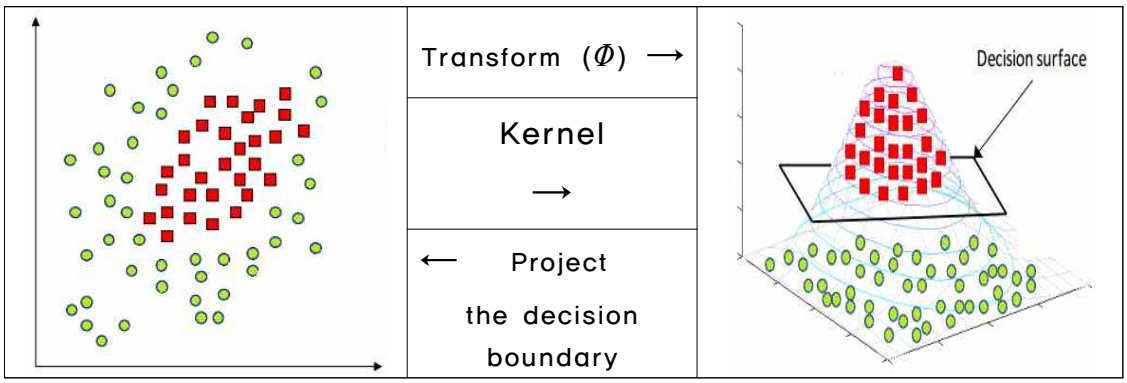
Nonlinear SVM

1. Kernel SVM

데이터를 선형으로 분류하기 위해 차원을 높이는 방법

Feature Map(Φ)을 통해 차원을 높인다: X 대신 $\Phi(X)$

커널: Feature Map의 내적



Original space가 아닌 Feature space에서 학습: $X \rightarrow \Phi(X)$

Original space: Nonlinear decision boundary

→ Feature space: Linear decision boundary

고차원 Feature space에서 분류가 더 쉬울 수 있다

고차원 Feature space를 효율적으로 계산할 수 있다

$\Phi : X \rightarrow Z = \Phi(X)$
$ex) 2D \rightarrow 5D$
$\Phi : (x_1, x_2) \rightarrow (x_1, x_2, x_1^2, x_2^2, x_1 x_2)$

비선형 SVM의 목적 함수

$$\text{Dual Problem: } \max_{\alpha} \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \Phi(x_i)^T \Phi(x_j)$$
$$subjected\,to \sum_{i=1}^{n} \alpha_i y_i = 0, 0 \le \alpha_i \le C, i - 1, 2, ..., n$$

커널 $K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$일 때

$$\text{Dual Problem: } \max_{\alpha} \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$
$$subjected\,to \sum_{i=1}^{n} \alpha_i y_i = 0, 0 \le \alpha_i \le C, i - 1, 2, ..., n$$

Kernel Mapping

ex) $X = (x_1, x_2),\ Y = (y_1, y_2)$
$\Phi(X) = (x_1^2, x_2^2, \sqrt{2}\, x_1 x_2), \Phi(Y) = (y_1^2, y_2^2, \sqrt{2}\, y_1 y_2)$
$< \Phi(X), \Phi(Y) \ge x_1^2 y_1^2 + x_2^2 y_2^2 + 2 x_1 x_2 y_1 y_2$

But 실제로는 연산량이 매우 많다

해결 방법: 명시적으로 $\Phi(X), \Phi(Y)$를 각각 계산하지 않고

커널을 사용하여 암묵적으로 $< \Phi(X), \Phi(Y) >$를 계산하여 연산 효율을 높힌다

$$\begin{aligned}(X, Y)^2 &= < (x_1, x_2), (y_1, y_2) >^2 \\ &= < x_1 y_1 + x_2 y_2 >^2 \\ &= x_1^2 y_1^2 + x_2^2 y_2^2 + 2 x_1 x_2 y_1 y_2 \\ &= < \Phi(X) \Phi(Y) >\end{aligned}$$

Kernel Function: $(X, Y)^2 = < (x_1, x_2), (y_1, y_2) >^2 = K(X, Y)$

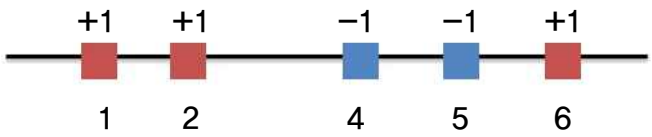
Kernel Function의 종류

1. Linear Kernel: $K(x_1, x_2) = < x_1, x_2 >$
2. Polynomial Kernel: $K(x_1, x_2) = (a < x_1, x_2 > + b)^d$
3. Low Degree Polynomial Kernel: $K(x, y) = (xy + 1)^2$
4. Sigmoid Kernel: $K(x_1, x_2) = \tanh(a < x_1, x_2 > + b)$
5. Gaussian Kernel(RBF, Radial Basis Function Kernel)

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

비선형 SVM 예시

i	x_i	y_i
1	1	+1
2	2	+1
3	4	−1
4	5	−1
5	6	+1



1. 선형 분류가 불가능하므로 커널 사용한다

Low degree polynomial kernel: $K(x,y)=(xy+1)^2$

Tuning parameter C = 100

2. α 계산

$$\max_{\alpha}\sum_{i=1}^{5}\alpha_i-\frac{1}{2}\sum_{i=1}^{5}\alpha_i\alpha_j y_i y_j(x_i x_j+1)^2, subjected\ to\ \sum_{i=1}^{5}\alpha_i y_i=0, 0\le\alpha_i\le 100$$

complementary slackness에 의해

1. $\alpha_i\neq 0\rightarrow\alpha_i>0, x_i\rightarrow SV$
2. $\alpha_i=0, x_i\neq SV$

α_1	α_2	α_3	α_4	α_5
0	2.5	0	7.333	4.833
−	SV	−	SV	SV

3. α 계산 → b 계산 → 모델 학습 완료

SVM model: $f(x)=\sum_{i\in SV}\alpha_i y_i K<\Phi(x_i),\Phi(x_{new})>+b$

$$SV=\{\alpha_2,\alpha_4,\alpha_5\}$$

$$f(x)=2.5(+1)(2x+1)^2+7.333(-1)(5x+1)^2+4.833(+1)(6x+1)^2+b$$
$$=0.667x^2-5.333x+b$$

$$f(2)=0.667(2^2)-5.333(2)+b=1, b\approx 9$$

$$\therefore\ f(x)=0.667x^2-5.333x+9$$

$\Rightarrow f(x)=0$: 결정 경계 $(hyperplane)$

$f(x_{new})>0\rightarrow class: 1$

$f(x_{new})<0\rightarrow class: -1$

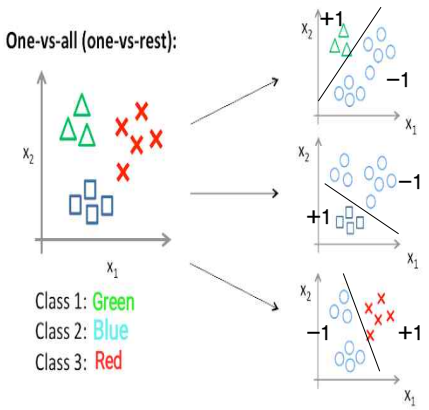
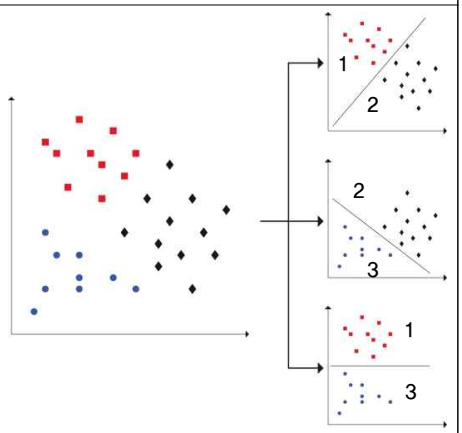
커널 선정 방법

정해진 기준 없으므로 실험적으로 결정한다

사용하는 커널에 따라 feature space의 특징이 달라지므로
데이터의 특성에 맞는 커널을 결정하는 것이 중요하다

일반적으로 RBF kernel, Sigmoid kernel,
Low Degree Polynomial kernel(4차 미만) 등이 사용된다

SVM 다계층 분류(Multi-Classification)

	하나-나머지 방법 (OvR. One-vs-Rest)	하나-하나 방법 (OvO. One-vs-One)
	이항 분류 값(hypothesis function)이 가장 큰 값을 그룹으로 할당한다	주어진 특성 자료에 대해 가장 많이 할당된 그룹으로 할당한다(voting)
		
분류기 개수	클래스 수 n개	${}_nC_2$

SVM api

1. 선형 SVM

sklearn.svm.SVC(kernel='linear', C=1.0, class_wight=None, random_state=None)

kernel: 선형 SVM = 'linear'
{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'}, default = 'rbf'
C: 에러 허용 정도(float, default = 1.0)
class_weight: dict or 'balanced', default = None

2. 비선형 SVM

skelarn.svm.SVC(kernel='rbf', C=1.0, gamma=scale, class_weight=

None, ramdom_state=None)

kernel: {'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'},
default = 'rbf'

C: 에러 허용 정도(float, default = 1.0)

gamma: 'rbf', 'poly', 'sigmoid' kernel의 설정 파라미터
{'scale', 'auto'} or float, default = 'scale'
'scale' = 1 / (n_features * X.var())
'auto' = 1 / n_features
gamma가 작을수록 두 데이터의 거리를 실제보다 멀게 변환(overfit)
클수록 두 데이터의 거리르 실제보다 가깝게 변환(underfit)

class_weight: dict or 'balanced', default = None

Accuracy, Precision, Recall, F1 평가 지표
sklearn.metrics.classification_report(y_true, y_pred)

데이터 압축
sklearn.decomposition.PCA(n_component=None, whiten=False, svd_solver='auto')

n_component: number of components to keep
n_component == min(n_sample, n_feature)
int, float or 'mle', default = None

whiten: feature가 상관관계 최소화 (bool, default = None)

svd_solver:
{'auto', 'full', 'arpack', 'randomized'}, default = 'auto'

GrudSearchCV
: 파라미터 변경하면서 가장 좋은 성능의 분류기 찾는 방법
sklearn.model_selection.GridSearchCV(estimator, param_grid, cv=None)

estrimator: 평가 대상(학습시킨 모델)

param_grid: 다양하게 실험할 파라미터
{dict or list of dictionaries}

cv: cross-validation strategy
{int, cross-validation generator or an iterable}, default = None