
START
머신러닝과 딥러닝
Machine Learning deep Learning

Chapter 8 비선형분류모형 III

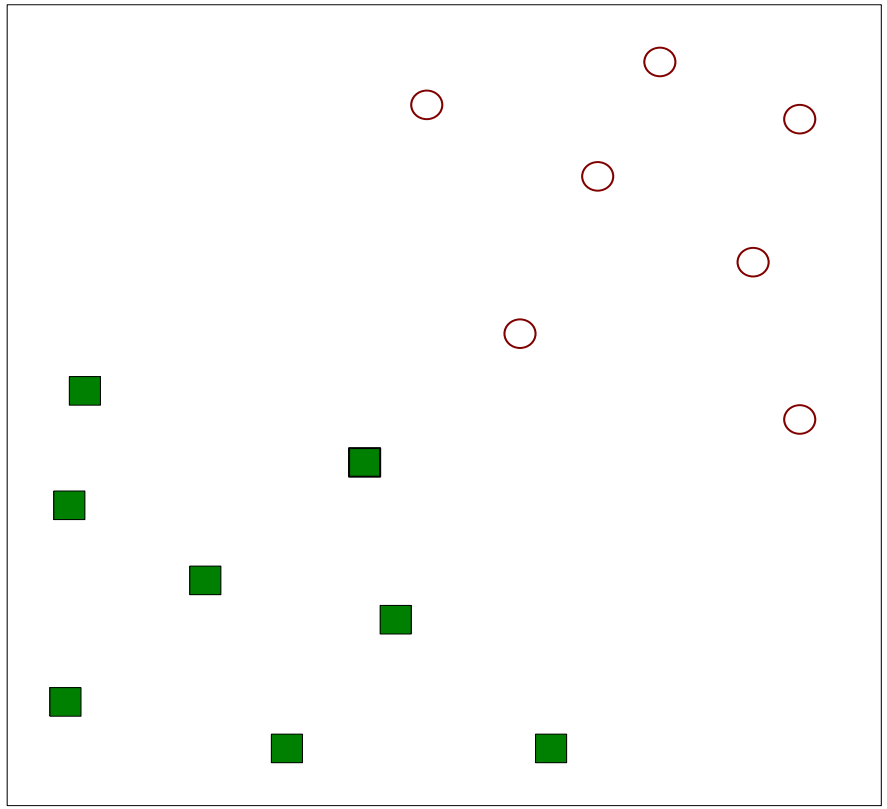
Machine Learning & Deep Learning

손영두

e-mail: youngdoo@dongguk.edu

Support Vector Machine

- Find a linear hyperplane (decision boundary) that will separate the data

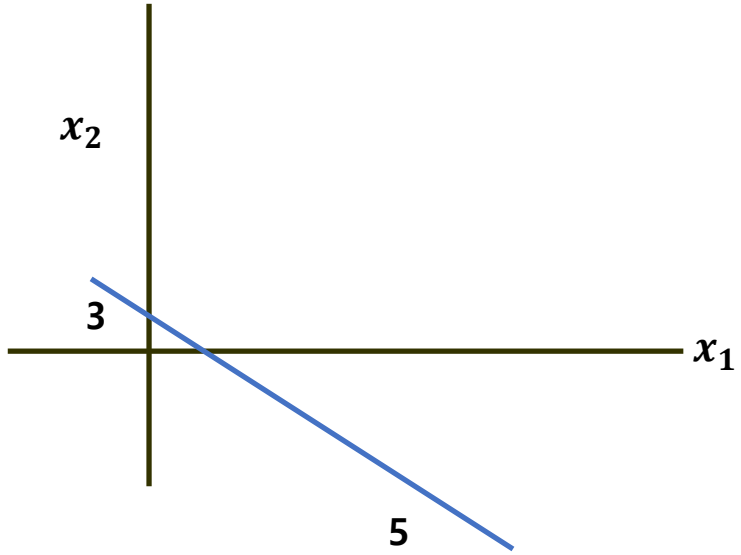


Hyperplane (초평면)

- Hyperplane E^n: the set of points $X = \{x \mid cx = z\}$ with $c \neq 0$
- Hyperplane은 임의의 공간을 반으로 양분
- z=00이면, hyperplane이 원점을 지나게됨

$$wx = w1x_1 + w_2x_2 + \cdots + w_nx_n = z$$

$$wx = 3x_1 + 5x_2 = 15$$

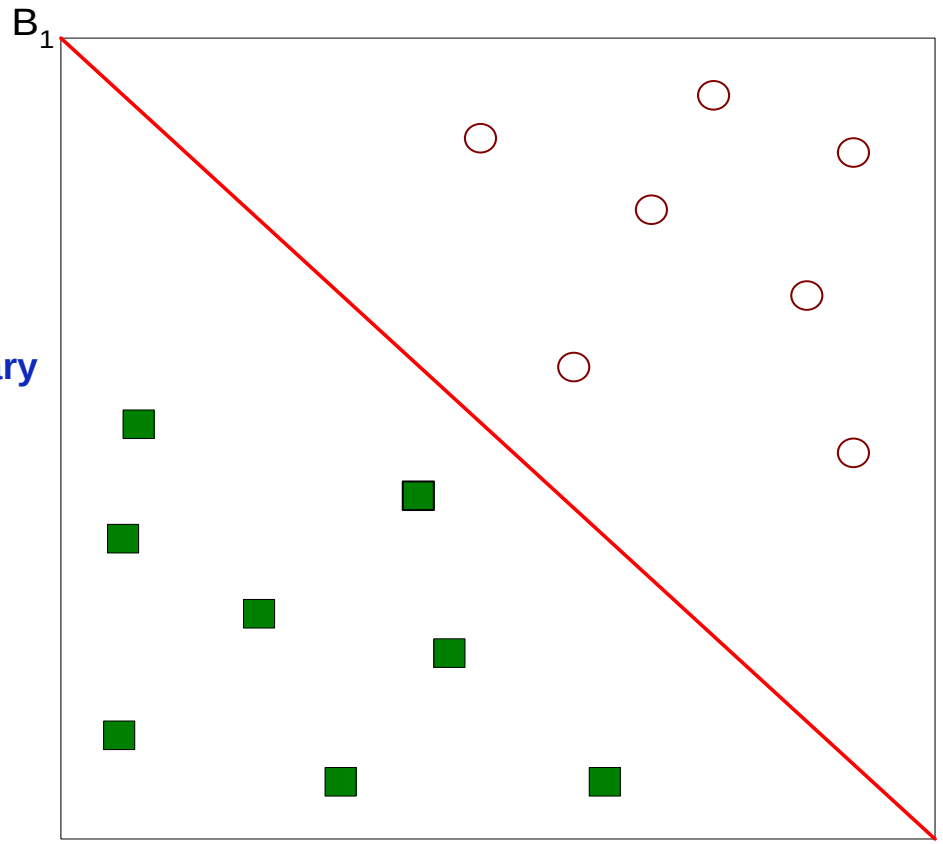


- 2차원 공간에서의 초평면: 1차원 직선
- 3차원 공간에서의 초평면: 2차원 평면

Support Vector Machine

One possible solution

A linearly separable data set



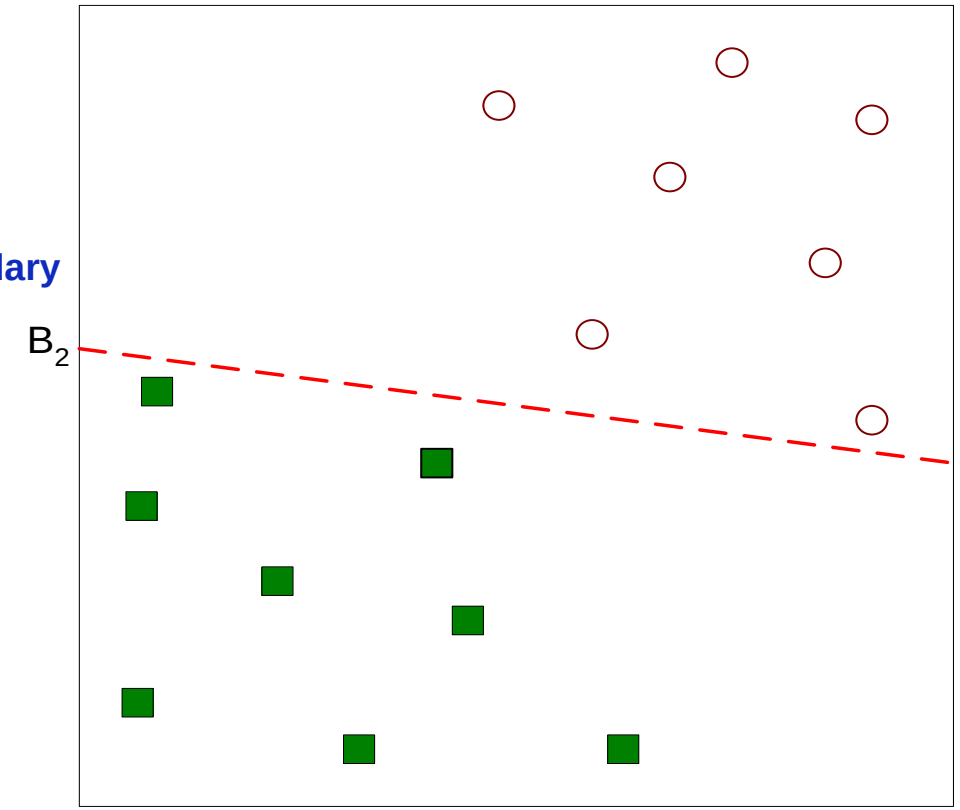
B1: decision boundary

Support Vector Machine

Another possible solution

A linearly separable data set

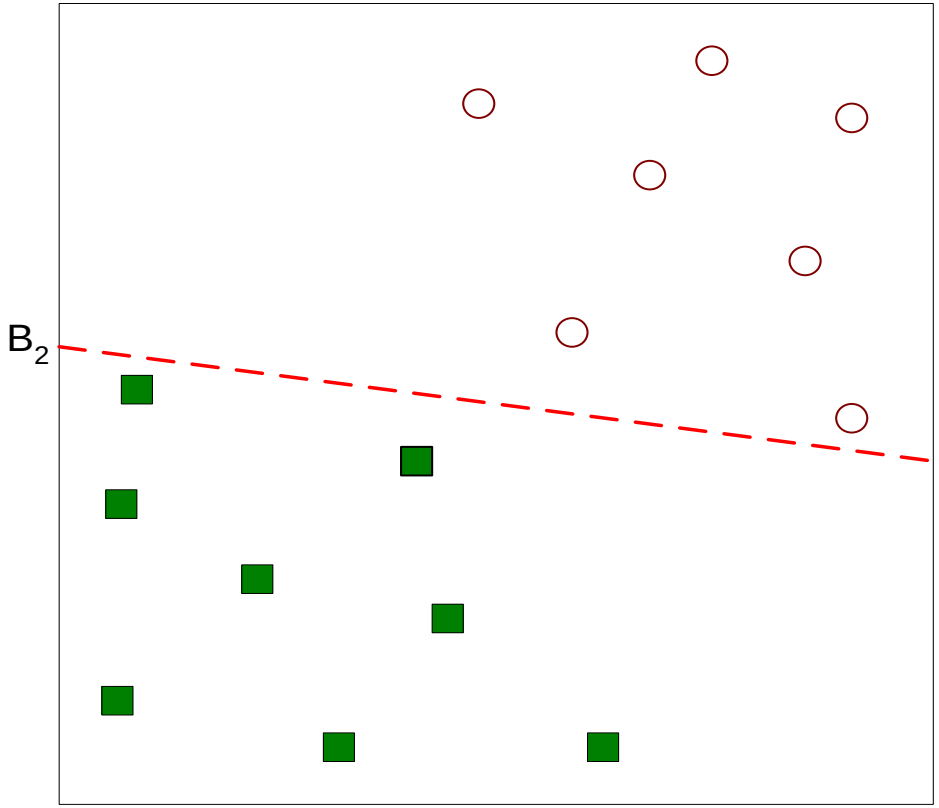
B2: decision boundary



Support Vector Machine

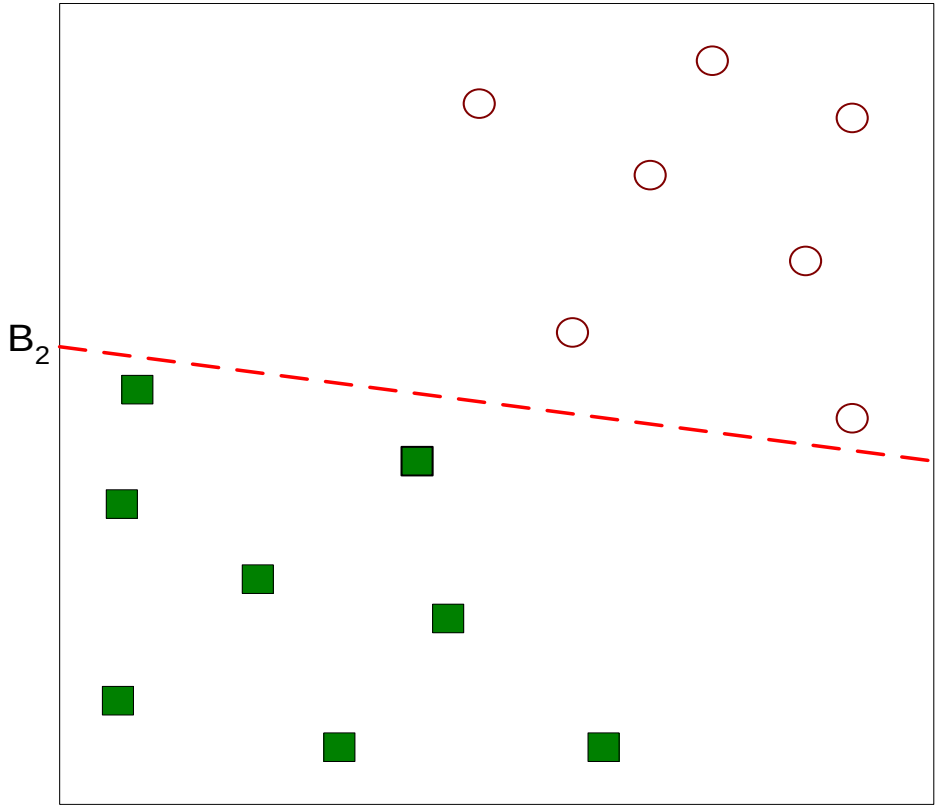
Other possible solutions

A linearly separable data set

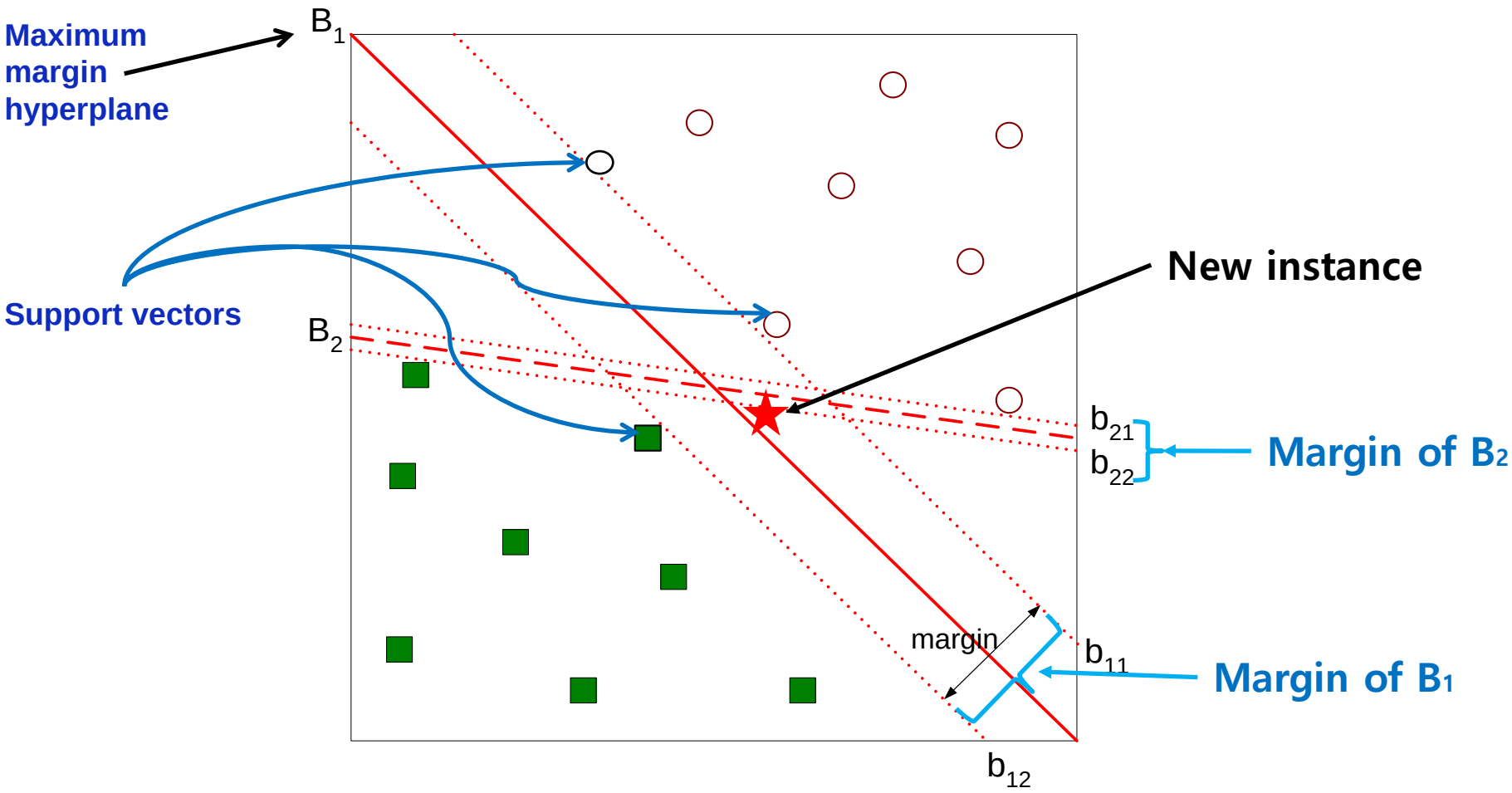


Support Vector Machine

- Which is better? B_1 or B_2?
- How can we define “better”?



Support Vector Machine: Maximum Margin Classification



☑ Margin을 최대화하는 hyperplane을 선택: B_1이 B_2보다 더 좋은 classifier

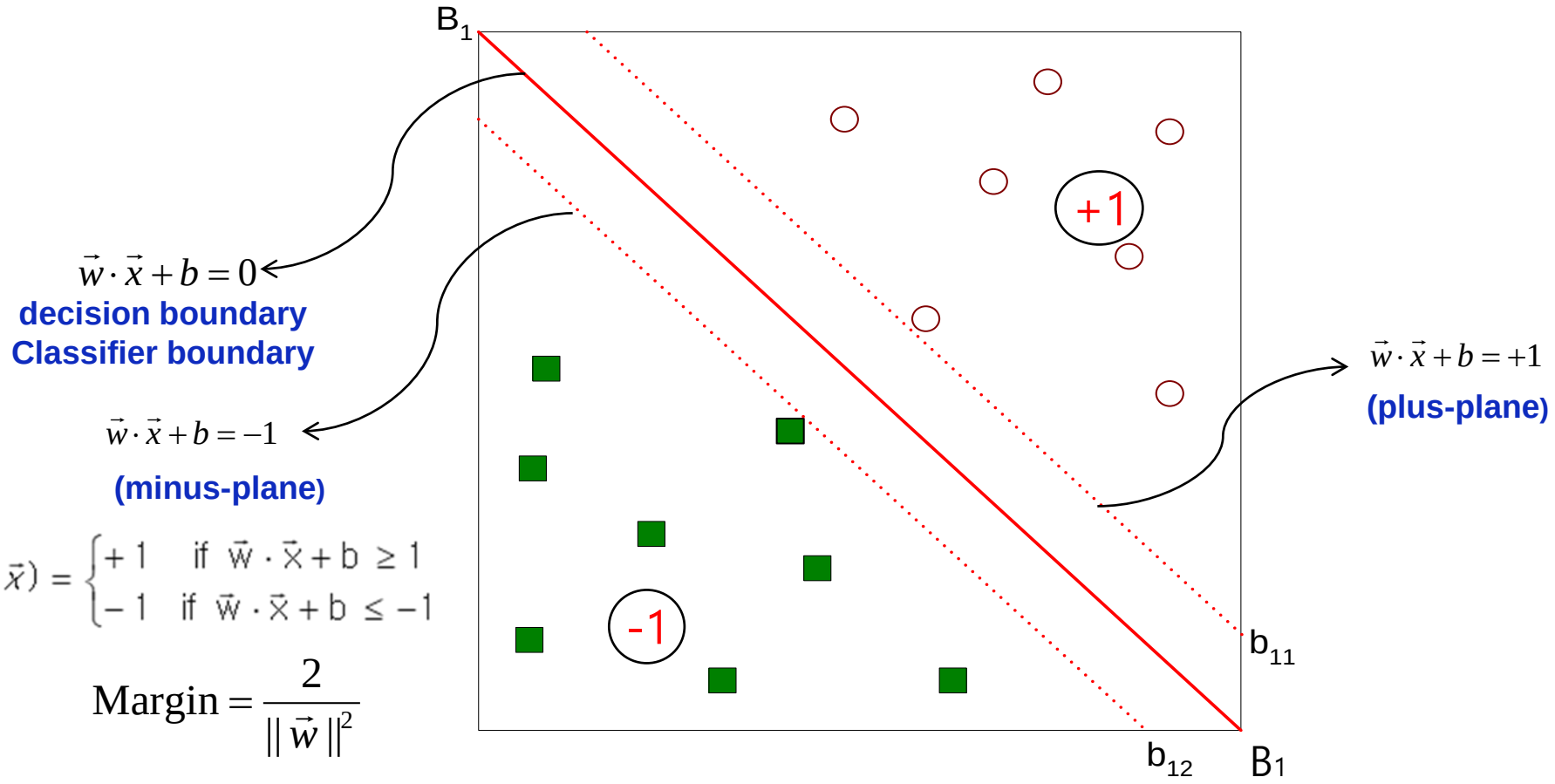
Support Vector Machine: Maximum Margin Classification

- Margin을 최대화하는 hyperplane을 선택: B_1이 B_2보다 더 좋은 classifier
- **Margin :** decision boundary로부터 가장 가까운 데이터까지의 거리 (또는 양 쪽의 합)
- Margin이 클 수록 generalization error가 낮아지는 경향이 있음
  - Margin이 작을 경우, decision boundary에 대한 작은 섭동이 데이터에 대한 분류에 큰 영향을 미칠 수 있음 : overfitting이 될 가능성이 높음
- 관련된 이론적 뒷받침 존재 & 실제 문제의 적용에서도 좋은 성능을 보임

Linear SVM: separable case

Linearly separable case

- 각 class label이 +1, -1로 주어진 경우



$\vec{w} \cdot \vec{x} + b = 0$
decision boundary
Classifier boundary

$\vec{w} \cdot \vec{x} + b = -1$
(minus-plane)

$\vec{w} \cdot \vec{x} + b = +1$
(plus-plane)

$$f(\vec{x}) = \begin{cases} +1 & \text{if } \vec{w} \cdot \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \cdot \vec{x} + b \leq -1 \end{cases}$$

$$\text{Margin} = \frac{2}{\| \vec{w} \|^2}$$

Linear SVM: separable case

- N개의 학습 데이터로 이루어진 binary classification의 경우

$$(x_i, y_i)\ (i = 1, \ldots, N)\ where\ y_i \in \{-1, 1\}$$

- Linear classifier의 decision boundary는 아래와 같이 표현 가능

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

- Square point에 대하여:

$$\mathbf{w} \cdot \mathbf{x_s} + b = k\ , where\ k < 0$$

- Circle point에 대하여:

$$\mathbf{w} \cdot \mathbf{x_c} + b = k'\ , where\ k' > 0$$

- Test 데이터에 대하여는 아래와 같이 분류 가능

$$y = \begin{cases} 1 & if\ \mathbf{w} \cdot \mathbf{z} + b > 0 \\ -1 & if\ \mathbf{w} \cdot \mathbf{z} + b < 0 \end{cases}$$

Linear SVM: separable case

- 다음과 같이 parameter들을 rescale이 가능

  boundary에 있는 point들에 대하여,

$$b_{i1} : w \cdot x + b = 1 \quad \text{if } y_i = 1$$
$$b_{i2} : w \cdot x + b = -1 \quad \text{if } y_i = -1$$

- Boundary의 안 쪽에 있는 point들에 대하여,

$$b_{i1} : w \cdot x + b > 1 \quad \text{if } y_i = 1$$
$$b_{i2} : w \cdot x + b < -1 \quad \text{if } y_i = -1$$

- 두 식을 종합하면

$$y_i(w \cdot x + b) >= 1$$

- 이 경우 margin $\rho = 2/\|w\|$로 표현 가능

Linear SVM: separable case

Quadratic optimization problem:

Find $\mathbf{w}$ and b such that

$$\rho = 2r = \frac{2}{\|\mathbf{w}\|} \text{ is maximized}$$

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Reformulation:

Find $\mathbf{w}$ and b such that

$\mathbf{\Phi}(\mathbf{w}) = \|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w}$ is minimized

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Lagrangian dual 및 Karush-Kuhn-Tucker condition을 이용하여 SVM의 해를 구함

(편의상 앞으로 목적함수에 1/2를 추가)

Karush-Kuhn-Tucker (KKT) Condition (Optional)

Given general problem

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{subject to} \quad & h_i(x) \leq 0, \quad i = 1, \dots m \\ & \ell_j(x) = 0, \quad j = 1, \dots r \end{aligned}$$

The **Karush-Kuhn-Tucker conditions** or **KKT conditions** are:

- $0 \in \partial f(x) + \sum_{i=1}^{m} u_i \partial h_i(x) + \sum_{j=1}^{r} v_j \partial \ell_j(x)$ (stationarity)
- $u_i \cdot h_i(x) = 0$ for all i (complementary slackness)
- $h_i(x) \leq 0$, $\ell_j(x) = 0$ for all i, j (primal feasibility)
- $u_i \geq 0$ for all i (dual feasibility)

Linear SVM: separable case

$$\boldsymbol{Min}\ \boldsymbol{L}(\boldsymbol{w},\boldsymbol{b},\boldsymbol{\lambda}) = \frac{1}{2}\|w\|^2 - \sum_i^N \lambda_i y_i (w \cdot x_i + b) + \sum_i^N \lambda_i$$ …Lagrangian

K.K.T condition

$$\frac{\partial}{\partial w} L_p(w,b,\lambda) = w - \sum_i^N \lambda_i y_i x_i = 0$$

$$\frac{\partial}{\partial b} L_p(w,b,\lambda) = -\sum_i^N \lambda_i y_i = 0$$

…Gradient of the Lagrangian=0

$$y_i (w \cdot x_i + b) - 1 \geq 0 \quad \forall i$$ …Primal Feasibility

$$\lambda_i \geq 0 \quad \forall i$$ …Dual Feasibility

$$\lambda_i (y_i (w \cdot x_i + b) - 1) = 0 \quad \forall i$$ …Complementarity Conditions

Linear SVM: separable case

여기서,

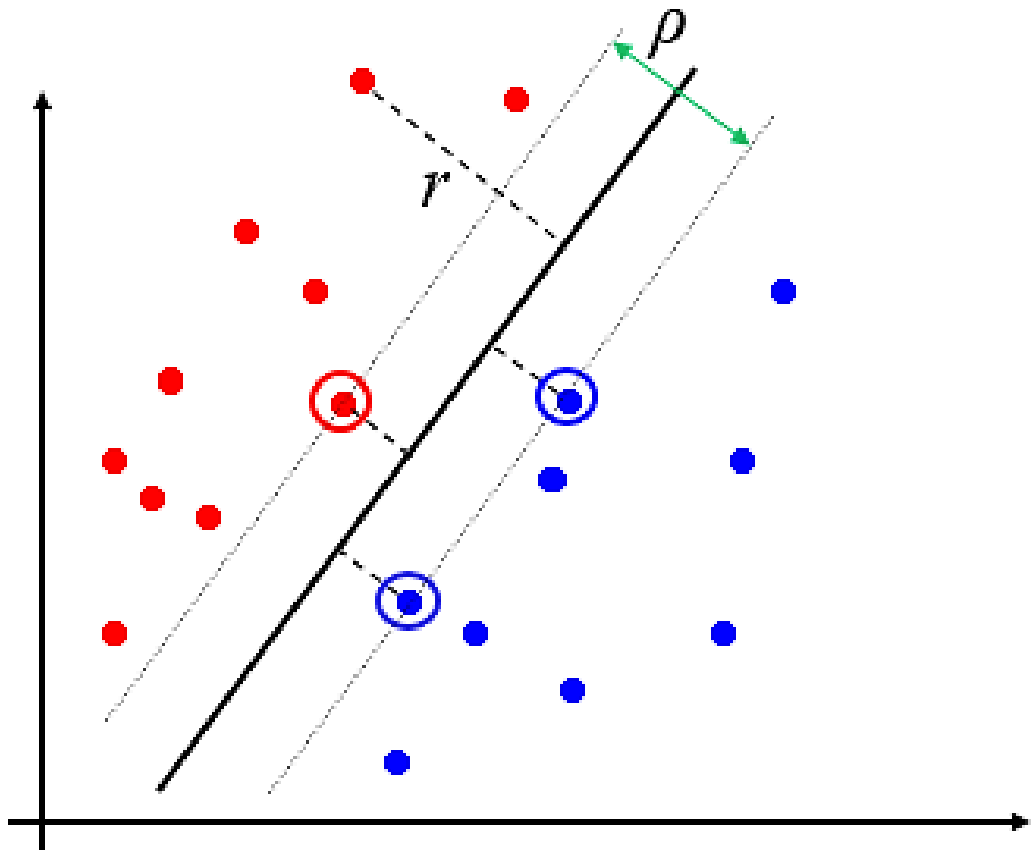
$\lambda_i[y_i(w \cdot x_i+b)-1]=0$ 을 살펴보면

training instance X_i 가 $y_i(w \cdot x_i+b)-1=0$ 을 만족하지 않는 경우에는 $\lambda_i=0$이 되어야 하는데, 이것은 training instance가 $y_i(w \cdot x_i+b)-1=0$ 인 hyperplane상에 없는 경우에만 $\lambda_i=0$이 됨을 의미함 (Complementary slackness)

training instance가 $y_i(w \cdot x_i+b)-1=0$ 을 만족하는 경우에만 $\lambda_i>0$의 값을 갖는다.
즉, $\lambda_i>0$인 경우 training instance X는 $y_i(w \cdot x_i+b)-1=0$ 인 hyperplane b_{i1}, 또는 b_{i2} 상에 위치하는데, 이러한 training instance들을 support vector라 한다.

hyperplane의 w, b값은 support vectors에 의해서만 결정되며, support vector가 아닌 다른 training instance는 w, b값을 결정하는데 사용되지 않는다.

Linear SVM: support vectors



Linear SVM: separable case

- 주어진 Lagrangian primal 문제는 Lagrangian dual 문제로 변경이 가능
- Primal problem: minimization on w, b, λ

$$\min \quad Lp(w,b,\lambda) = \frac{1}{2}\mathrm{w}^{\mathrm{T}}\mathrm{w} - \sum_{\mathrm{i}}^{\mathrm{N}} \lambda_i y_i (w^T \cdot x_i + b) + \sum_{i}^{N} \lambda_i$$

$$w = \sum_{i}^{N} \lambda_i y_i x_i \quad , \sum_{i}^{N} \lambda_i y_i = 0$$

- Dual problem: maximization of λ

$$\max \quad L_d(\lambda) = -\frac{1}{2}\sum_{i}^{N}\sum_{j}^{N} \lambda_i \lambda_j y_i y_j x_i^T \cdot x_j + \sum_{i}^{N} \lambda_i$$

$$s.t. \quad \lambda_i \geq 0, \ \sum_{i}^{N} \lambda_i y_i = 0, \quad \forall\ i$$

$$\lambda_i [y_i (w^T x_i + b) - 1] = 0$$

Linear SVM: separable case

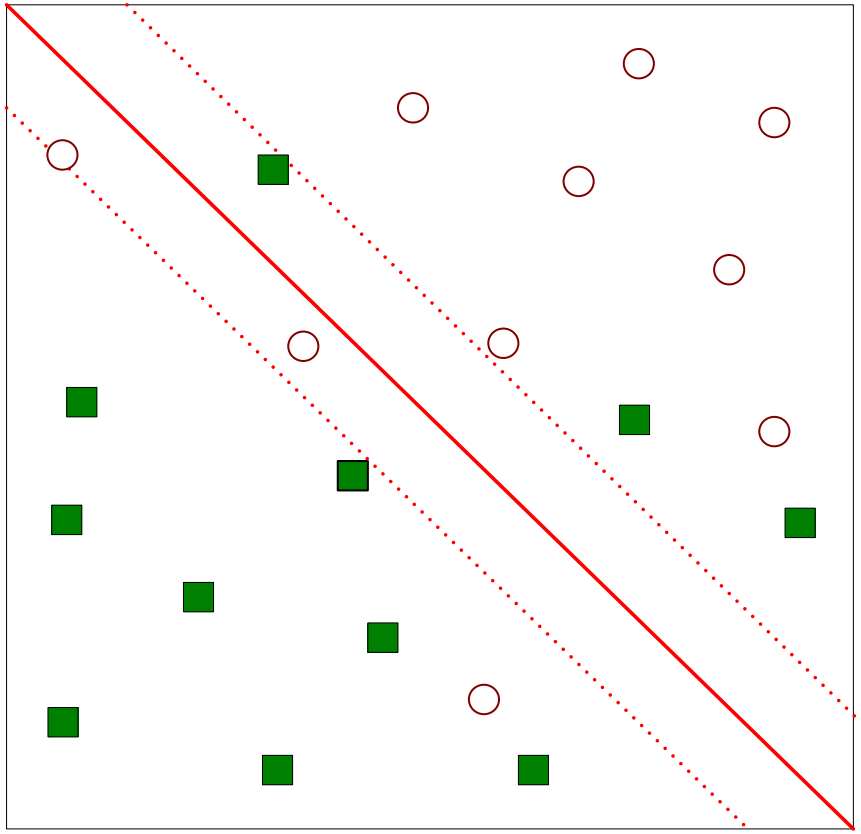
- Dual Problem을 풀어 λ를 구해내면, decision boundary를 아래와 같이 표현 가능

$$\left(\sum_{i=1}^{N} \lambda_i y_i x_i \cdot x\right) + b = 0$$

- 따라서 λ=0인 데이터(support vector가 아닌 데이터)들은 decision boundary에 영향을 주지 못함

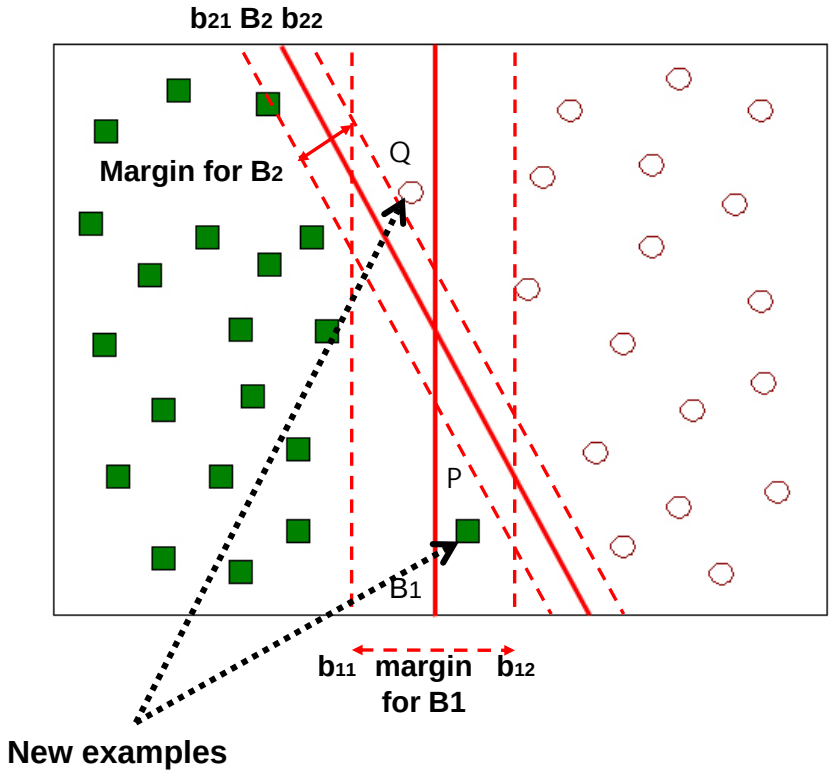
Linear SVM: non-separable case

Linearly non-separable case



Linear SVM: non-separable case

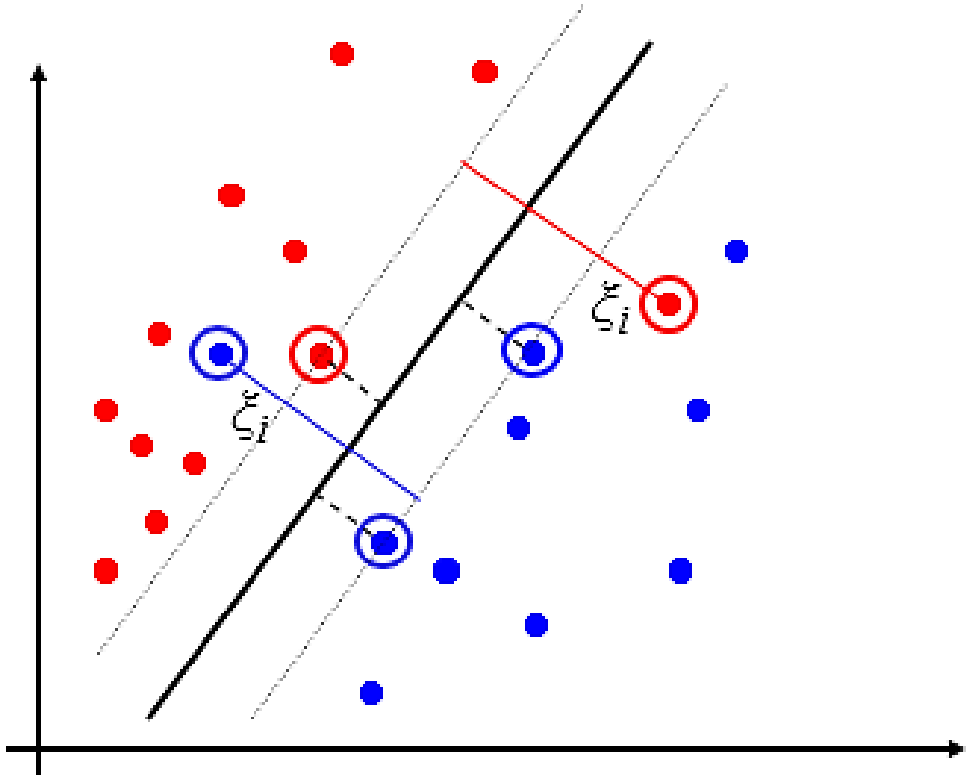
- Soft margin 방법 사용



- 작은 오차는 허용하여 decision boundary를 구함
- 따라서 오차와 margin 간의 trade-off 관계에 대한 조절이 필요
- 오차를 위하여 slack variable ξ 도입

Linear SVM: non-separable case

- ***Slack variables*** ξ_i **can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft*.**

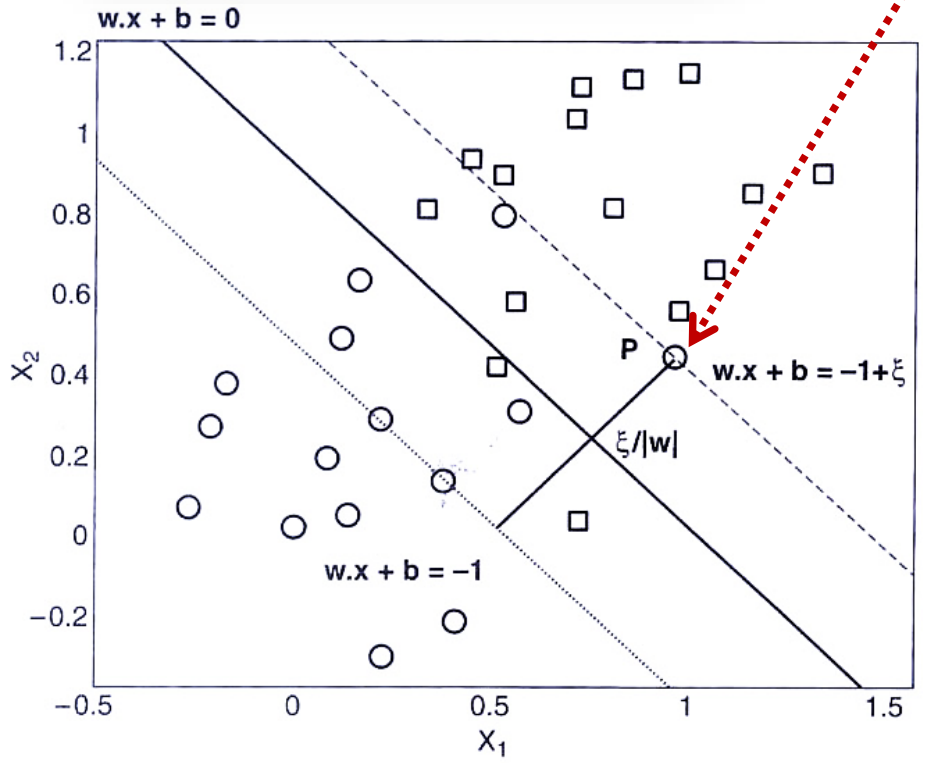


Linear SVM: non-separable case

$$w \cdot x_i + b \geq 1 - \xi_i \quad \text{if } y_i=1$$

$$w \cdot x_i + b \leq -1 + \xi_i \quad \text{if } y_i=-1$$

,where $\xi_i \geq 0$



P: one of the instances that violates the constraints in Eq 5.35

ξ provides an estimate of the error of the decision boundary on the training example P

$$wx_1 + b = -1 + \xi$$

$$wx_2 + b = -1$$

$$w(x_1 - x_2) = \xi$$

$$\|w\| \cdot d = \xi$$

$$d = \frac{\xi}{\|w\|}$$

Linear SVM: non-separable case

새로운 목적함수

Find $\mathbf{w}$ and b such that

$\Phi(\mathbf{w}) = \mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$, $\xi_i \geq 0$

모수 C를 통하여 margin과 error의 trade-off 조절

Separable case와 마찬가지로 Lagrangian dual을 통하여 풀이 가능

Linear SVM: non-separable case

The modified objective function and constraints:

$$f(w) = \frac{\|w\|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i\right)^k$$

$$w \cdot x_i + b \geq 1 - \xi_i$$

$$w \cdot x_i + b \leq -1 + \xi_i$$

,where $\xi_i \geq 0$

Lagrangian for the constrained optimization(k=1)

$$Lp = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \lambda_i \{y_i(w \cdot x_i + b) - 1 + \xi_i\} - \sum_{i=1}^{N} \mu_i \xi_i$$

$$y_i(x_i \cdot w + b) - 1 + \xi_i \geq 0$$

$$\xi_i \geq 0$$

Given conditions

$$\lambda_i \geq 0$$

$$\mu_i \geq 0$$

Lagrange multiplier conditions

$$\lambda_i \{y_i(x_i \cdot w + b) - 1 + \xi_i\} = 0$$

$$\mu_i \xi_i = 0$$

KKT complementary conditions

Linear SVM: non-separable case

Separable case와 같은 형태의 decision boundary를 도출

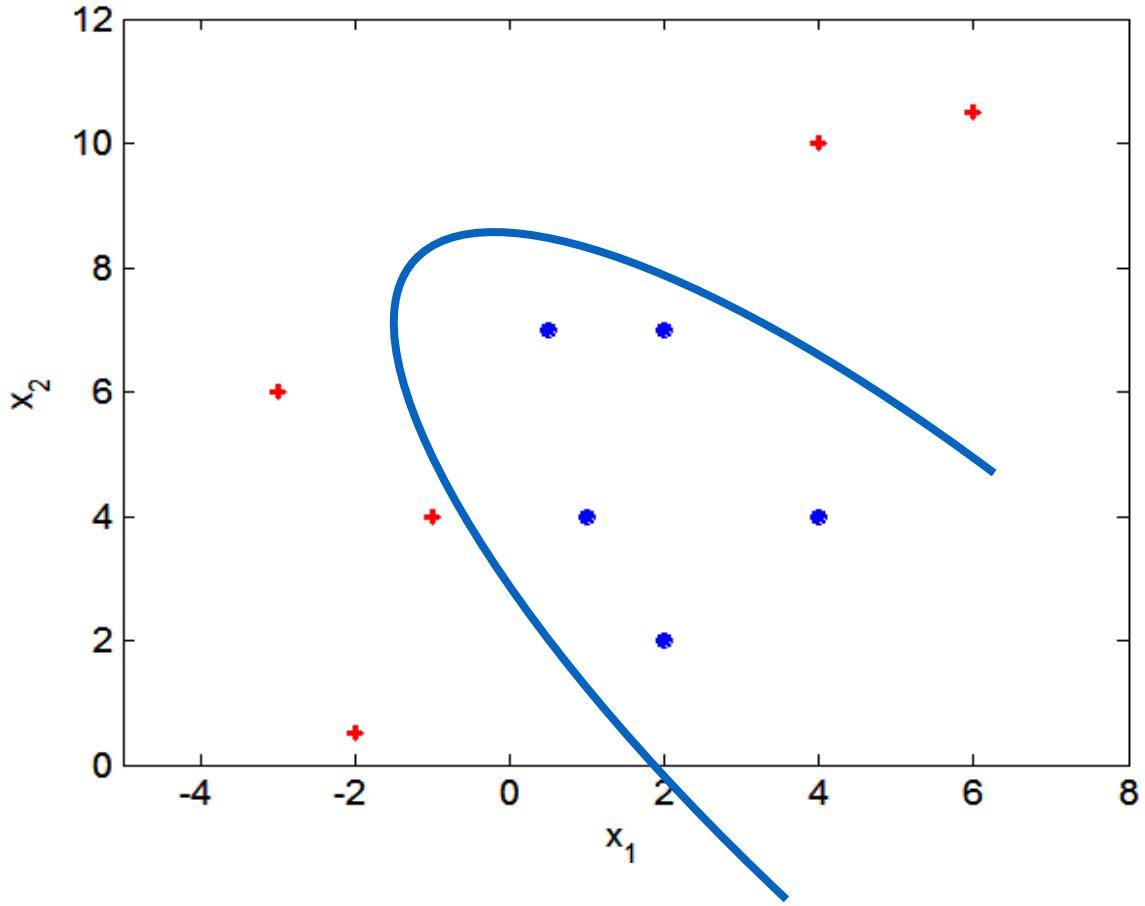
$$(\sum_{i=1}^{N} \lambda_i y_i x_i \bullet x) + b = 0$$

차이점

- Support vector on boundary:
- Bounded support vector (samples off the boundary):

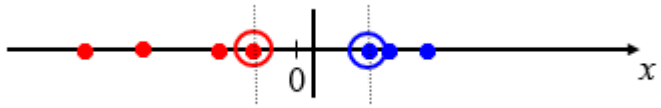
Nonlinear SVM

Decision boundary 가 비선형인 경우?

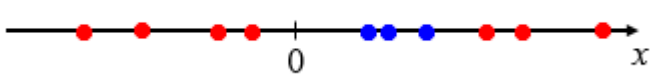


Nonlinear SVM

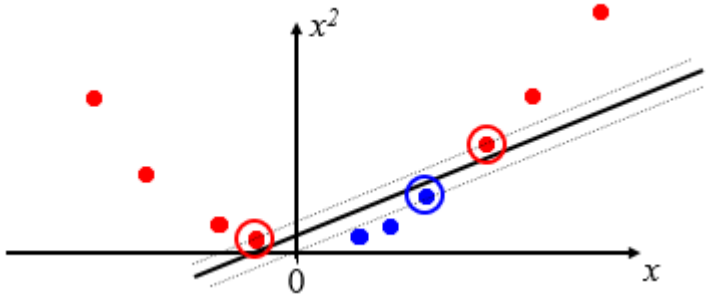
- 선형분리가 가능한 경우:



- 선형분리가 불가능한 경우:

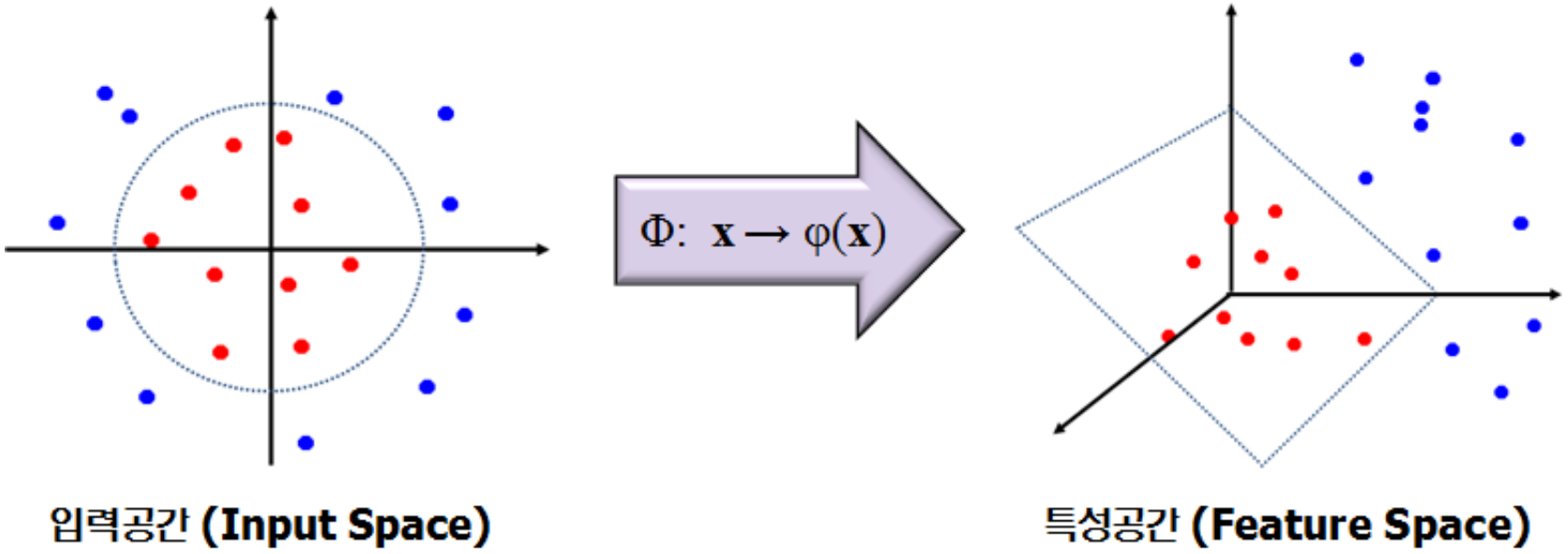


- 고차원 mappling을 통해 해결이 가능할까?



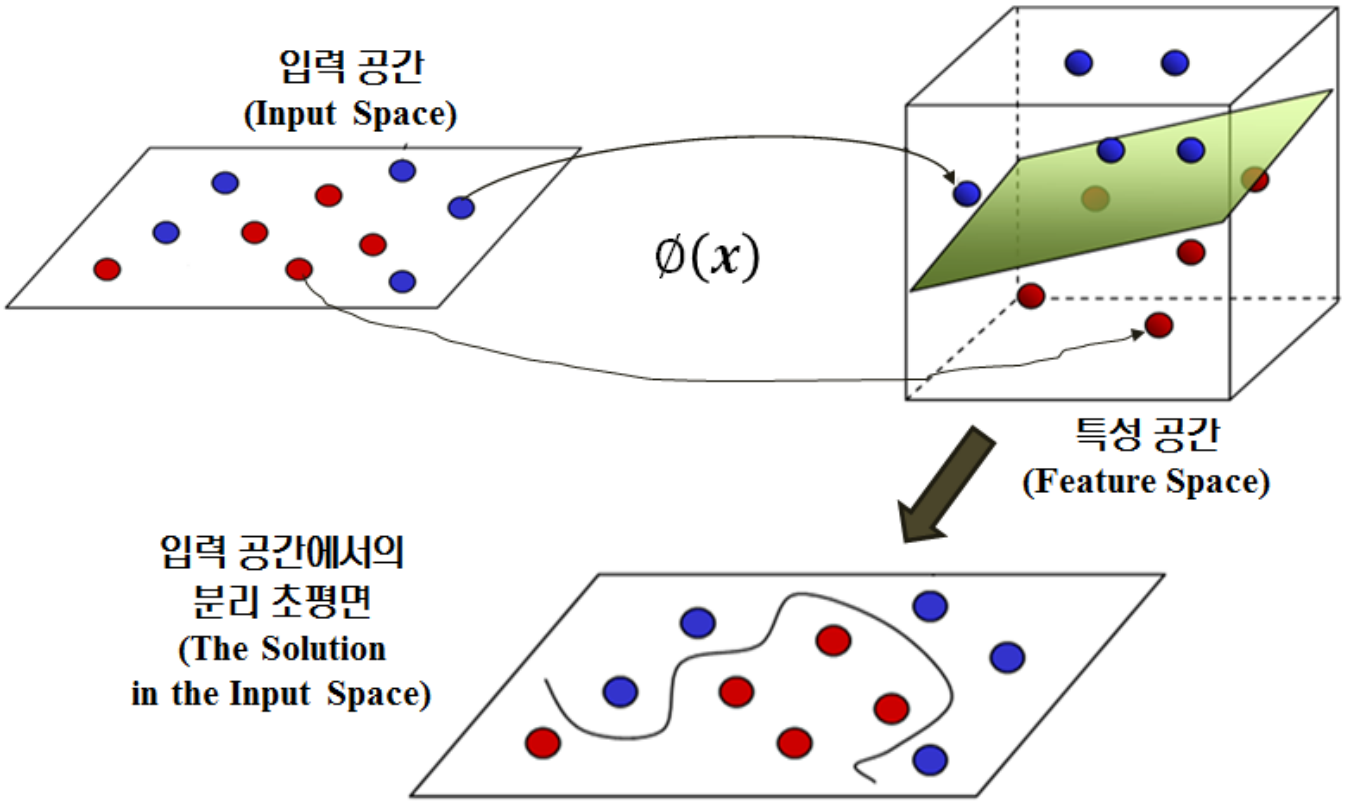
Nonlinear SVM: feature space

- General idea: 입력 공간을 학습 데이터의 선형 분리가 가능한 고차원 특성 공간 (feature space)으로 mapping이 가능하다



Nonlinear SVM: feature space

☑ 고차원 공간에서의 선형 분리는, 입력 공간에서의 비선형 분리로 표현될 수 있다



Nonlinear SVM

- Nonlinear SVM을 입력 공간이 아닌 특성 공간에서 문제를 해결할 경우 기존의 linear SVM과 같은 방법으로 풀어낼 수 있음

$$\min \frac{\|w\|^2}{2}$$

$$\text{s.t.} \quad y_i\,(w \cdot \Phi(x_i) + b) \geq 1, \; i = 1,2,\ldots,N$$

- Decision boundary

$$f(z) = sign(w \cdot \Phi(z) + b)$$

$$= sign(\sum_{i=1}^{n} \lambda_i y_i \Phi(x_i) \cdot \Phi(z) + b)$$

- 얼마나 고차원으로 mapping을 해야할까?
  - 고차원으로 mapping할수록 두 mapped feature vector의 내적의 계산에 많은 자원이 소요됨

Nonlinear SVM: kernel trick

- Linear SVM은 입력 벡터의 내적에만 의존: $K(x_i, x_j) = x_i^T x_j$

- 만일 데이터가 특정 변환 Φ: $x \rightarrow \varphi(x)$ 에 의해 고차원 특성 공간으로 사영되더라도, 특성공간에서의 특성벡터의 내적에만 의존:

$$K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$$

- *Kernel function* 은 특성 공간에서의 내적 값과 같은 결과를 주는 함수

- Example:

2-dimensional vectors $x=[x_1 \;\; x_2]$;
let $K(x_i, x_j)=(x_i^T x_j)^2$, then find $\varphi(x)$

- 따라서 kernel function은 데이터를 간접적으로 고차원으로 사영함 (직접적으로 $\varphi(x)$를 정의할 필요 없음)

Nonlinear SVM: kernel trick

- 매번 K(xi,xj)= φ(xi) Tφ(xj)를 만족하는 φ(x)를 찾는 것은 번거로움
- Mercer's theorem:

  Every positive semi-definite symmetric function is a kernel
- Positive semi-definite 함수는 Positive semi-definite한 Gram matrix와 연관됨

$$\mathrm{K}=\begin{bmatrix} K(\mathbf{x}_1,\mathbf{x}_1) & K(\mathbf{x}_1,\mathbf{x}_2) & K(\mathbf{x}_1,\mathbf{x}_3) & \dots & K(\mathbf{x}_1,\mathbf{x}_n) \\ K(\mathbf{x}_2,\mathbf{x}_1) & K(\mathbf{x}_2,\mathbf{x}_2) & K(\mathbf{x}_2,\mathbf{x}_3) & & K(\mathbf{x}_2,\mathbf{x}_n) \\ & & & & \\ \dots & \dots & \dots & \dots & \dots \\ K(\mathbf{x}_n,\mathbf{x}_1) & K(\mathbf{x}_n,\mathbf{x}_2) & K(\mathbf{x}_n,\mathbf{x}_3) & \dots & K(\mathbf{x}_n,\mathbf{x}_n) \end{bmatrix}$$

Nonlinear SVM: positive semi-definite

$$\forall z \neq 0,\ z^T A z > 0 \rightarrow positive\ definite$$

$$\forall z \neq 0,\ z^T A z \geq 0 \rightarrow positive\ semidefinite$$

$$\forall z \neq 0,\ z^T A z < 0 \rightarrow negative\ definite$$

$$\forall z \neq 0,\ z^T A z \leq 0 \rightarrow negative\ semidefinite$$

Nonlinear SVM: kernel 함수의 종류

Linear: $K(x_i, x_j) = x_i^T x_j$

- Mapping Φ: x → φ(x), where φ(x) is x itself

Polynomial of power p: $K(x_i, x_j) = (1 + x_i^T x_j)^p$

- Mapping Φ: x → φ(x), where φ(x) has $\binom{d+p}{p}$ dimensions

Gaussian (radial-basis function): $K(x_i, x_j) = e^{\{-||x_i - x_j||^2/2\sigma^2\}}$

- Mapping Φ: x → φ(x), where φ(x) is *infinite-dimensional*: every point is mapped to *a function* (a Gaussian); combination of functions for support vectors is the separator.

SVM 실습 – 모듈 import와 데이터 생성과 학습

Make_blobs 함수를 통해 분류 데이터 생성 및 학습(fit)

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.datasets.samples_generator import make_blobs
X, y = make_blobs(n_samples=40, centers=2, random_state=20)
clf = svm.SVC(kernel='linear')
clf.fit(X, y)
```

추후에 'linear'를 변경해서 kernel을 조절할 수 있음

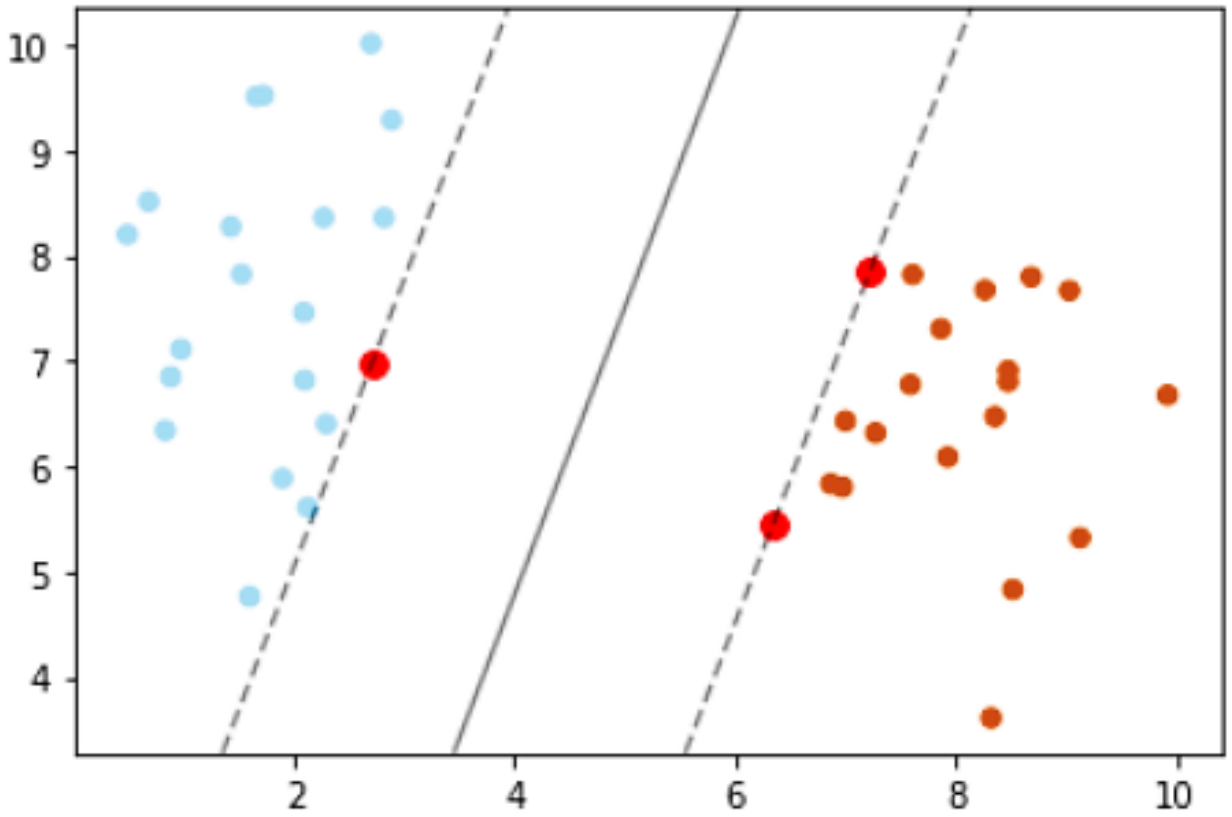
SVM 실습 – 학습된 모델의 시각화

학습된 모델을 토대로 시각화

```
plt.scatter(X[:,0], X[:,1], c=y, s=30, cmap=plt.cm.Paired)
# 초평면(Hyper-Plane) 표현
ax = plt.gca()
xlim = ax.get_xlim()
ylim = ax.get_ylim()
xx = np.linspace(xlim[0], xlim[1], 30)
yy = np.linspace(ylim[0], ylim[1], 30)
YY, XX = np.meshgrid(yy, xx)
xy = np.vstack([XX.ravel(), YY.ravel()]).T
Z = clf.decision_function(xy).reshape(XX.shape)
ax.contour(XX, YY, Z, colors='k', levels=[-1,0,1], alpha=0.5, linestyles=['--', '-', '--'])
# 지지벡터(Support Vector) 표현
ax.scatter(clf.support_vectors_[:,0], clf.support_vectors_[:,1], s=60, facecolors='r')
plt.show()
```

SVM 실습 – 시각화 결과 (linear)

- 빨간 점은 support vector
- 굵은 회색 선이 hyperplane



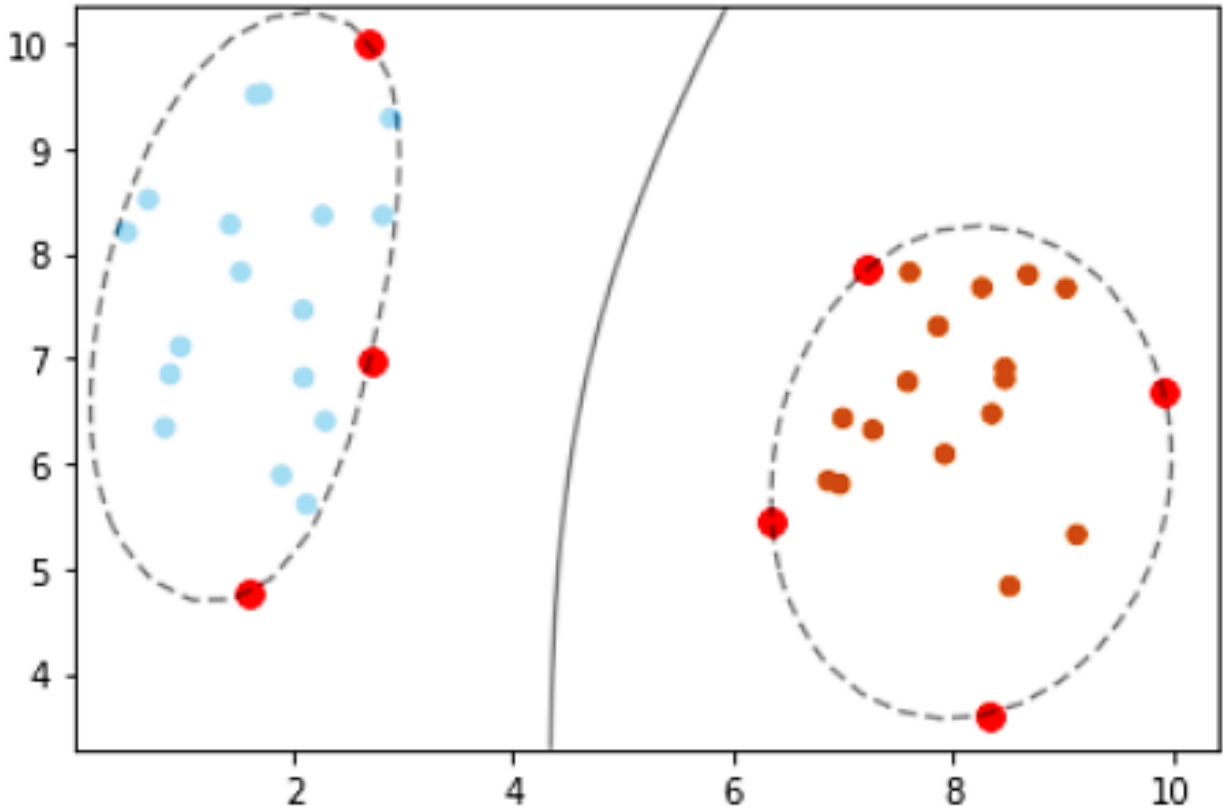
SVM 실습 – 모듈 import와 데이터 생성과 학습

‘linear’를 ‘rbf’로 변경해서 kernel을 조절

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.datasets.samples_generator import make_blobs
X, y = make_blobs(n_samples=40, centers=2, random_state=20)
clf = svm.SVC(kernel='rbf')
clf.fit(X, y)
```

SVM 실습 – 시각화 결과(rbf)

- 빨간 점은 support vector
- 굵은 회색 선이 hyperplane



실습 – Iris Data load

```
iris = load_iris()
iris_frame=pd.DataFrame(data=np.c_[iris['data'],iris['target']],columns=iris['feature_names'] + ['target'])
iris_frame['target'] = iris_frame['target'].map({0:"setosa",1:"versicolor",2:"virginica"})
X=iris_frame.iloc[:,:-1]
Y=iris_frame.iloc[:,[-1]]
iris_frame
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

실습 – Iris Data를 이용한 svm 학습

Sepal의 length와 width를 이용한 svm

```
clf = svm.SVC(kernel='linear')
import matplotlib.colors as colors
df1 = iris_frame[["sepal length (cm)", "sepal width (cm)",'target']]
X = df1.iloc[:,0:2]
Y = df1.iloc[:,2].replace({'setosa':0,'versicolor':1,'virginica':2}).copy()
clf.fit(X,Y)
```

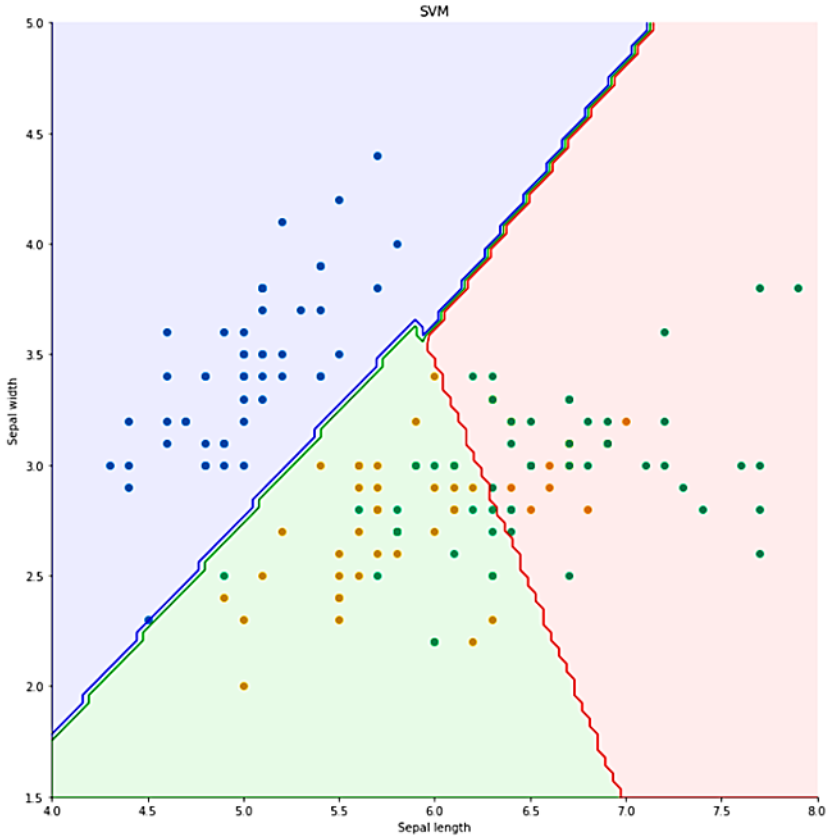
실습 – Iris Data와 svm 시각화

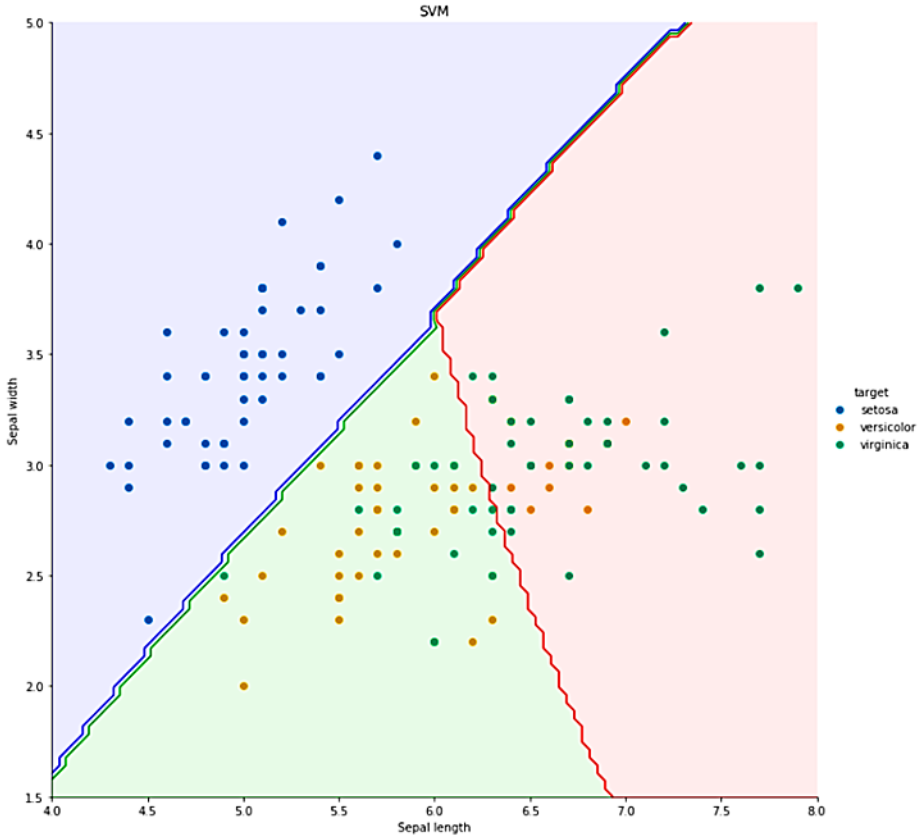
import seaborn as sn 해줘야 합니다.

```
N = 100
X_ = np.linspace(4, 8, N)
Y_ = np.linspace(1.5, 5, N)
X_, Y_ = np.meshgrid(X_, Y_)
color_list = ['Blues','Greens','Reds']
my_norm = colors.Normalize(vmin=-1.,vmax=1.)
g = sn.FacetGrid(iris_frame, hue="target", size=10, palette = 'colorblind') .map(plt.scatter, "sepal length (cm)",
my_ax = g.ax
zz = np.array( [clf.predict( [[xx,yy]])[0] for xx, yy in zip(np.ravel(X_), np.ravel(Y_)) ] )
Z = zz.reshape(X_.shape)
#Plot the filled and boundary contours
my_ax.contourf( X_, Y_, Z, 2, alpha = .1, colors = ('blue','green','red'))
my_ax.contour( X_, Y_, Z, 2, alpha = 1, colors = ('blue','green','red'))
# Add axis and title
my_ax.set_xlabel('Sepal length')
my_ax.set_ylabel('Sepal width')
my_ax.set_title('SVM')

plt.show()
```

실습 – Iris Data와 svm 시각화

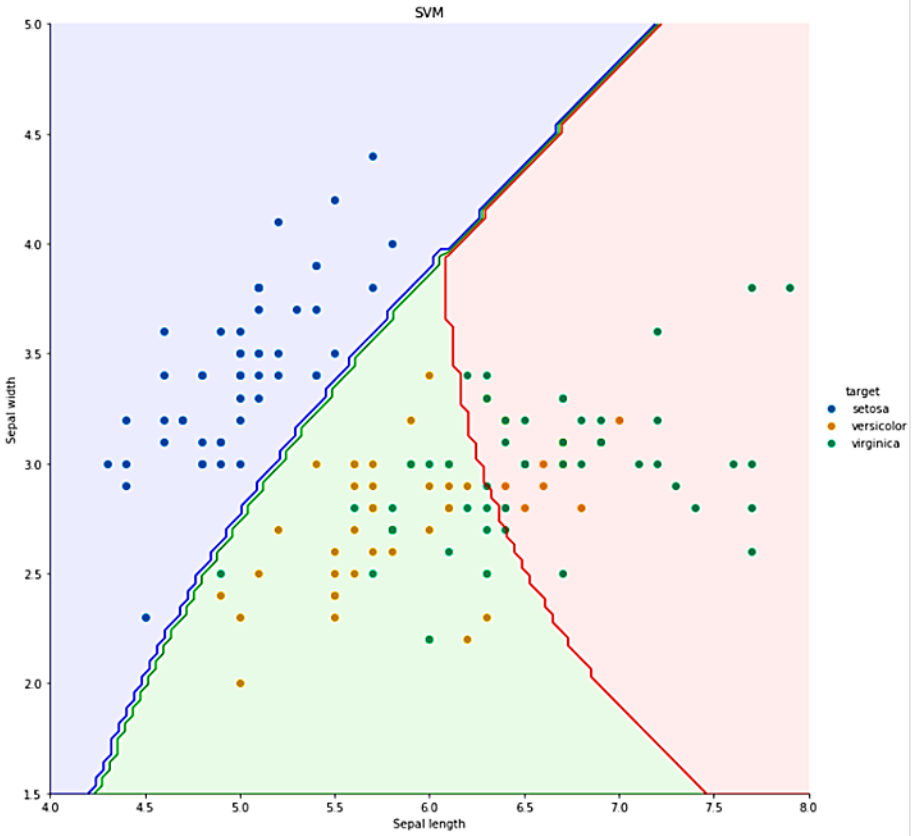




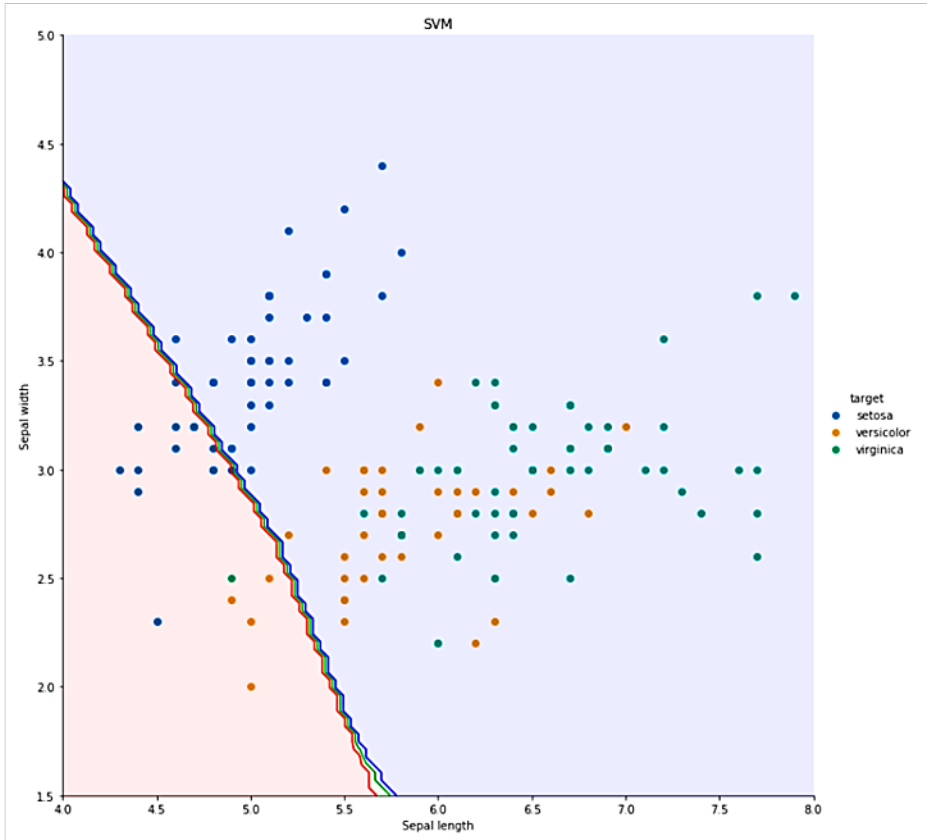
☑ Kernel = 'linear'

☑ Kernel = 'rbf'

실습 – Iris Data와 svm 시각화



☑ Kernel = ‘poly’



☑ Kernel = ‘sigmoid’