머신러닝응용 제12강

Support Vector Machine

첨단공학부 김동하교수

제12강 Support Vector Machine

1	Soft margin과 hard margin의 개념에 대해 학습한다.
2	선형 SVM의 개념과 최적화 문제에 대해 학습한다.
3	비선형 SVM에서 feature map과 kernel trick에 대해 학습한다.

핵심 단어

- Margin
- Support vector
- Soft (Hard) margin SVM
- Kernel trick

12강. Support Vector Machine

01. SVM 개요

1) Support Vector Machine

- Cortes and Vapnik (1995)

- 선형이나 비선형 분류를 할 수 있는 기계학습 방법론
  - 특히 복잡한 분류 문제를 잘 해결.

- 회귀 문제, 이상치 탐색 문제로도 확장 가능.
  - 본 강의에서는 이진 분류 문제만 고려 ($y \in \{-1, 1\}$)

1) Support Vector Machine

- 확률적 모형을 가정하지 않음.
  - 확률 추정 없이 직접 분류 결과에 대해 예측.

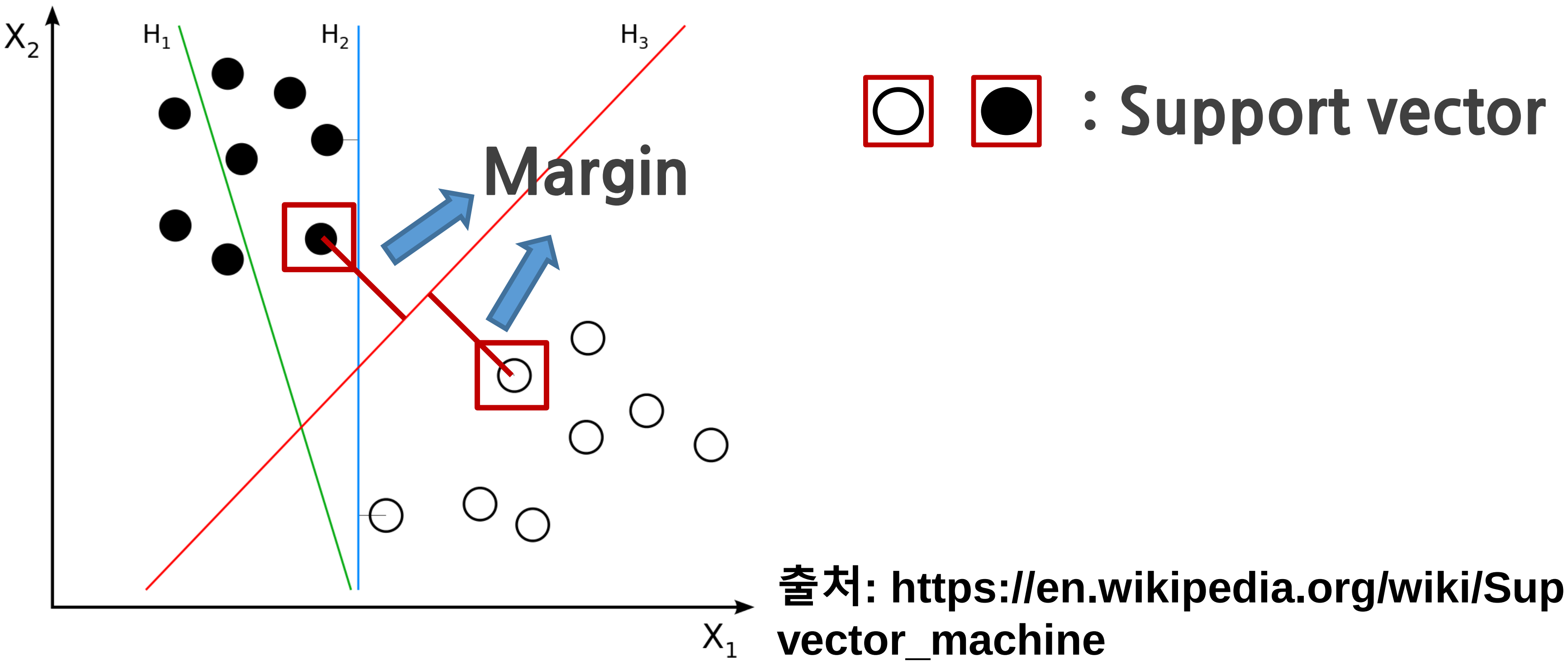
- 두 클래스 사이에 가장 너비가 큰 분류 경계선을 찾기 때문에 Large margin classification이라고도 함.

1) Support Vector Machine

◆ Margin을 최대로 하는 경계선은 H_3

◆ Support vector

- 분류 경계선과 가장 가까운 관측치



출처: https://en.wikipedia.org/wiki/Support-vector_machine

1) Support Vector Machine

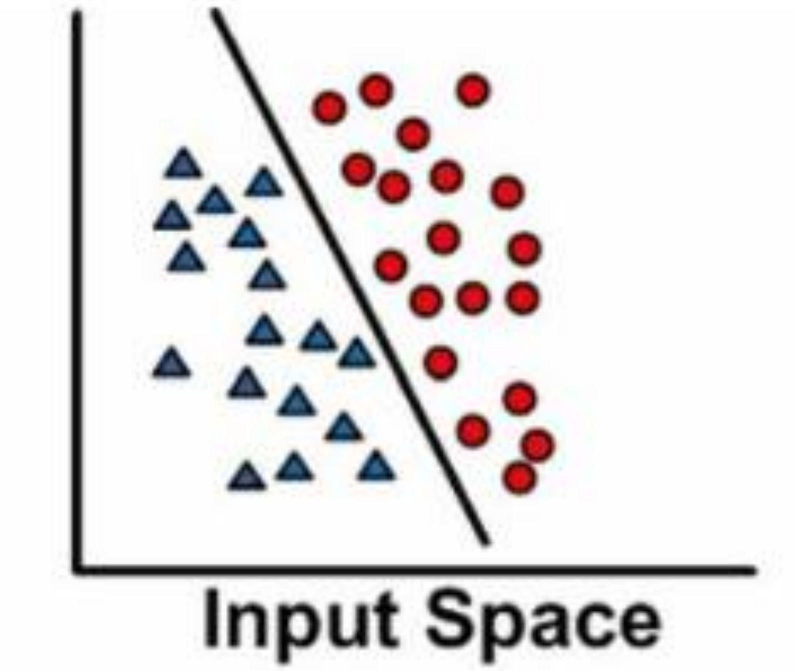
- SVM의 분류 경계선은 margin 계산에 민감하게 반응하므로 변수들 사이의 스케일을 맞춰주는 작업이 필요.

2) Linear SVM vs. Nonlinear SVM

◆ 선형 SVM

- 선형 분류 경계선으로 클래스를 분류.

$$sign(w^T x + b)$$



출처: https://www.researchgate.net/figure/Motivation-behind-non-linear-SVM-classifier_fig2_272520997

2) Linear SVM vs. Nonlinear SVM

◆비선형 SVM

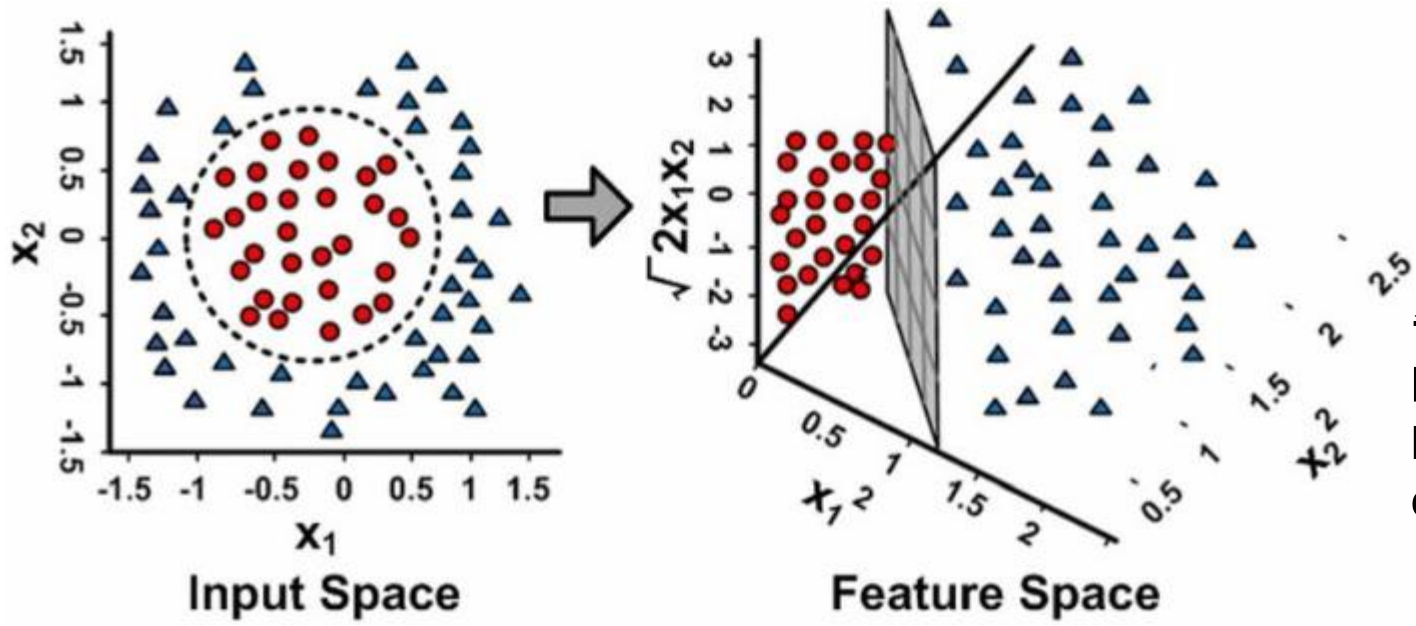
- 비선형 분류 경계선으로 클래스를 분류.
- 비선형 함수를 통해 데이터를 변형하고, 변형된 공간에서 선형 경계선을 활용.

$$sign(w^T\phi(x) + b)$$

2) Linear SVM vs. Nonlinear SVM

◆비선형 SVM

- $\phi(\cdot): R^p \rightarrow R^m$: feature function
- 입력 변수를 선형 분리가 쉽도록 변형해주는 함수
- 무한차원으로도 확장할 수 있음 (Kernel trick).



출처: https://www.researchgate.net/figure/Motivation-behind-non-linear-SVM-classifier_fig2_272520997

3) Hard Margin vs. Soft Margin

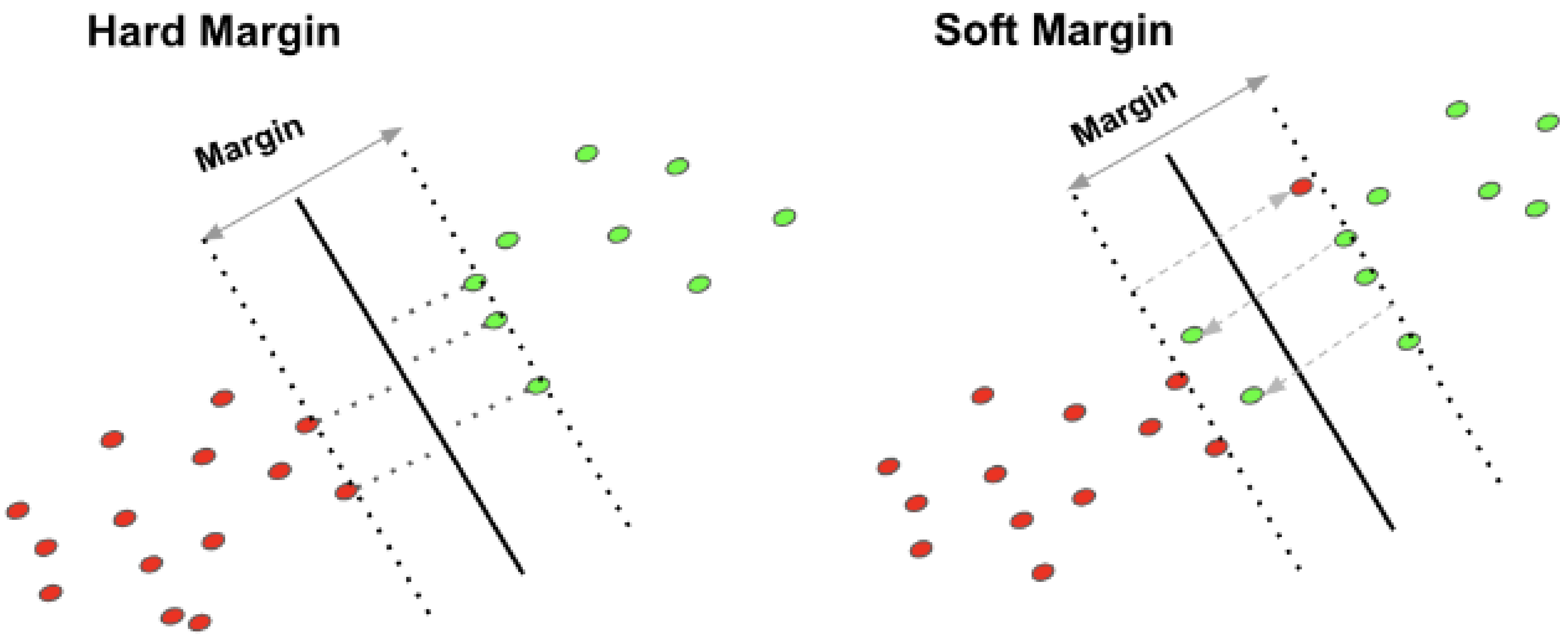
◆Hard Margin 방법

- 두 클래스가 하나의 선(혹은 평면)으로 완벽하게 나눠지는 경우에 적용 가능.

◆Soft Margin 방법

- 일부 샘플들이 분류 경계선의 분류 결과에 반하는 경우를 일정 수준 이하로 허용하는 방법.

3) Hard Margin vs. Soft Margin



출처: https://medium.com/@shubhamsalokhe/support-vector-machine-d5ef4b3de532

02. Linear SVM

1) Hard margin

- 학습 데이터: $(x_1, y_1), \dots, (x_n, y_n)$

- 다음의 문제를 최소화하는 기울기 w와 절편항 b를 찾는다.

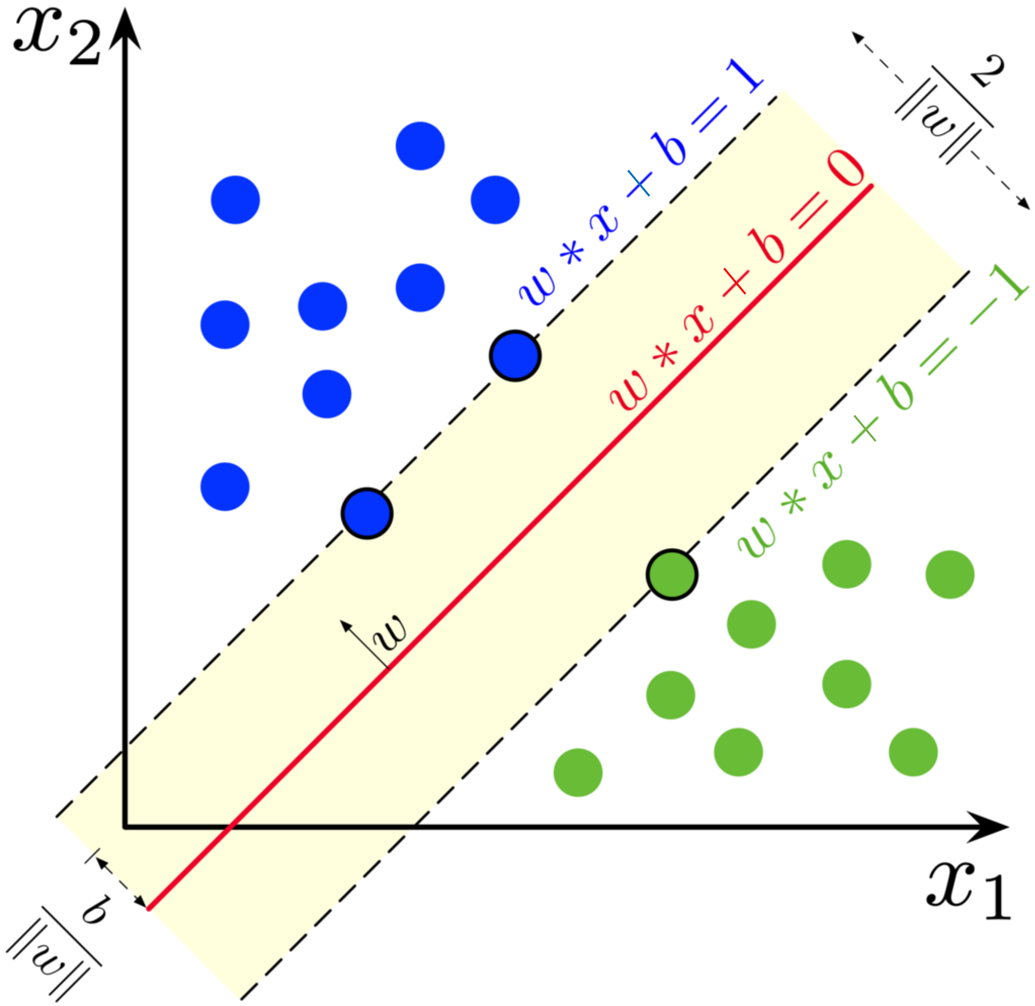
$$\min_{w,b} \|w\|_2^2$$

$$s.t.$$

$$y_i(w^T x_i + b) \geq 1 \; for \; i = 1, \dots, n$$

1) Hard margin

◆ 아래의 그림으로 해석할 수 있다.



출처: https://en.wikipedia.org/wiki/Support-vector_machine

2) Soft margin

- C: 분류 결과에 반하는 경우를 일정 수준으로 조정하는 조율 모수
- 다음의 문제를 최소화하는 기울기 w와 절편항 b를 찾는다.

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

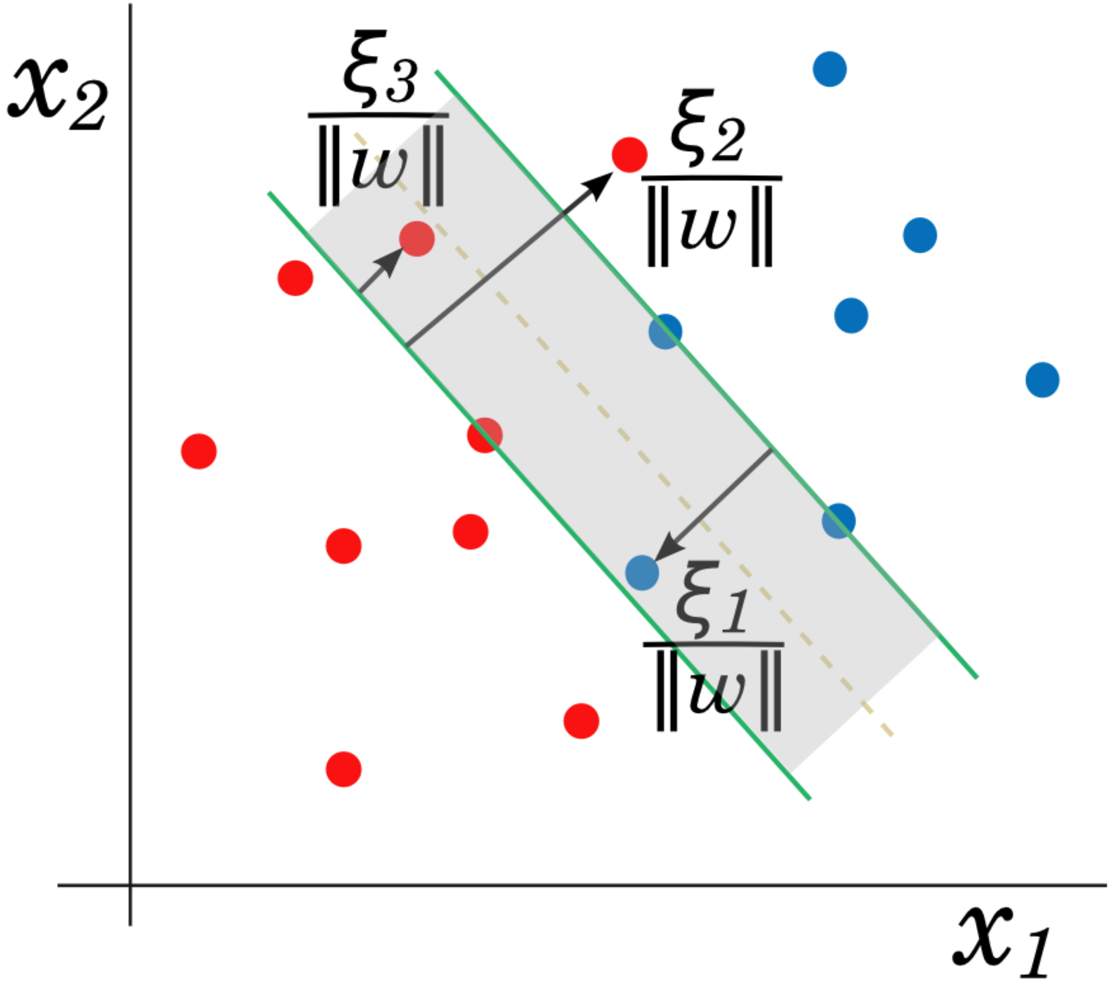
$$s.t.$$

$$y_i\left(w^T x_i + b\right) \geq 1 - \xi_i \; for \; i = 1, \ldots, n$$

$$\xi_i \geq 0 \; for \; i = 1, \ldots, n$$

2) Soft margin

◆아래의 그림으로 해석할 수 있다.



출처: https://www.efavdb.com/svm-classification

12강. Support Vector Machine

03. Nonlinear SVM

1) Feature map

- 비선형 SVM은 입력값을 feature map $\phi(\cdot): R^p \rightarrow R^m$을 이용해 변환한 후 선형 SVM을 사용하는 방법.

- Feature map
  - 복잡한 함수일 수록 변환된 feature space에서 y를 잘 나누는 선형 분류 경계선이 존재할 가능성이 높음.
  - 무한 차원으로 변환하면 어떨까? $(m = \infty)$

1) Feature map

- 무한 차원의 feature space에서 선형 분류 경계선을 찾으려면 무한 개의 모수가 필요함.

- 하지만, kernel trick 덕분에 n개의 모수만 필요함.

  - Kernel trick 이란 무엇일까?

2) Dual problem

◆ **선형 SVM 문제를 다음과 같은 문제로 변환할 수 있음. (Dual Problem)**

$$\min_{c_1,\ldots,c_n} -\sum_{i=1}^{n} c_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j c_i c_j x_i^T x_j$$

$$s.t.$$

$$\sum_{i=1}^{n} c_i y_i = 0,$$

$$0 \le c_i \le \frac{1}{2nC} \; for \; i = 1, \ldots, n$$

2) Dual problem

◆ 위의 문제의 해를 $\hat{c}_1, \dots, \hat{c}_n$이라 할 때, 분류 예측값은 다음과 같이 계산되어짐.

$$\hat{f}(x) = sign(w^T x + b)$$
$$= sign\left(\left[\sum_{i=1}^{n} \hat{c}_i y_i x_i^T x\right] + b\right).$$

- $w = \sum_{i=1}^{n} \hat{c}_i y_i x_i$
- $b = -w^T x_k + y_k$
  - (x_k, y_k): support vector

3) Kernel trick

◆마찬가지로 비선형 SVM을 dual problem으로 변환하면 다음과 같음.

$$\min_{c_1,\ldots,c_n} -\sum_{i=1}^{n} c_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j c_i c_j \boxed{K(x_i, x_j)}$$

$$\phi(x_i)^T \phi(x_j)$$

Kernel!

$$s.t.$$

$$\sum_{i=1}^{n} c_i y_i = 0\,,$$

$$0 \le c_i \le \frac{1}{2nC}\ for\ i = 1, \ldots, n$$

3) Kernel trick

◆따라서, 최적의 분류 경계선을 찾기 위해 알아야 하는 것은 feature map 그 자체가 아닌 kernel 값들임.

◆즉, 무한차원의 feature map이더라도 kernel만 정의되면 문제를 쉽게 풀 수가 있음!

- Kernel trick!

3) Kernel trick

◆위의 문제의 해를 $\hat{c}_1, \ldots, \hat{c}_n$이라 할 때, 분류 예측값은 다음과 같이 계산되어짐.

$$\hat{f}(x) = sign(w^T\phi(x) + b)$$
$$= \boxed{sign\left(\left[\sum_{i=1}^{n} \hat{c}_i y_i K(x_i, x)\right] + b\right)}.$$

- $w = \sum_{i=1}^{n} \hat{c}_i y_i x_i$
- $b = -w^T\phi(x_k) + y_k = -\left[\sum_{j=1}^{n} \hat{c}_j y_j K\left(x_j, x_k\right)\right] + y_k$
  - (x_k, y_k): support vector

4) 다양한 kernel 함수들

- 앞서 언급했듯이, 비선형 SVM은 feature map을 이용한 kernel이 정의되면 최적의 분류 경계선을 구할 수 있음.

- 널리 사용되는 kernel 함수는 다음과 같음.
  - Polynomial: $K(a,b) = (\gamma \cdot a^T b + r)^d$
  - Radial Basis: $K(a,b) = \exp(-\gamma \cdot \|a-b\|_2^2)$
  - Sigmoid: $K(a,b) = \tanh(\gamma \cdot a^T b + r)$

12강. Support Vector Machine

04. Python을 이용한 실습

1) 데이터 설명

◆ Iris 데이터셋

- 150개의 붓꽃에 대한 데이터
- 꽃잎의 각 부분의 너비와 길이 등을 측정
- sepal length: 꽃받침의 길이
- sepal width: 꽃받침의 너비
- petal length: 꽃잎의 길이
- petal width: 꽃잎의 너비
- species: 붓꽃의 종류 (setosa, versicolor, virginica)

1) 데이터 설명

◆분석 목표

- 붓꽃의 꽃받침과 꽃잎의 정보로 종류를 예측하는 SVM 을 학습하자.

2) 환경설정

◆필요한 패키지 불러오기

```
import os
import numpy as np
import pandas as pd
import collections
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC
#from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV

from sklearn.metrics import confusion_matrix
import seaborn as sns
```

3) 데이터 불러오기

◆ sns 패키지에 내장되어있는 iris 데이터를 불러오자.

```
iris = sns.load_dataset('iris')
print(iris.shape)
iris.head()
```

```
(150, 5)
```

	sepal_length	sepal_width	petal_length	petal_width	species
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

4) 데이터 전처리

◆종속변수와 설명변수 구분

◆데이터를 분할하고, 설명변수값을 표준화한다.

```
X = iris[['sepal_length','sepal_width',\
              'petal_length','petal_width']]
y = iris.iloc[:,4]
```

```
X_train, X_test, y_train, y_test = \
train_test_split(X, y, test_size = 0.3, random_state=123)
```

```
scaler = StandardScaler()
scaler.fit(X_train)
X_train = pd.DataFrame(scaler.transform(X_train))
X_test = pd.DataFrame(scaler.transform(X_test))
```

5) SVM 적합하기

◆ Grid search 를 통해 최적의 kernel 함수와 그에 필요한 조율 모수들을 구하자.

```
param_grid = [{'kernel': ['linear','sigmoid','rbf','poly'],
               'gamma': [1e-1, 1e-2, 1e-3, 1e-4],
               'C': [1, 10, 100, 1000]}]
best_grid = GridSearchCV(SVC(), param_grid, refit=True, verbose = 1)
best_svm_model = best_grid.fit(X_train, y_train)
best_svm_fit = best_svm_model.predict(X_train)
```

```
Fitting 5 folds for each of 64 candidates, totalling 320 fits
```

5) SVM 적합하기

◆최적의 조율 모수 결과

```
print('GridSearchCV 최적 파라미터:', best_grid.best_params_)
```

```
GridSearchCV 최적 파라미터: {'C': 1, 'gamma': 0.1, 'kernel': 'sigmoid'}
```

5) SVM 적합하기

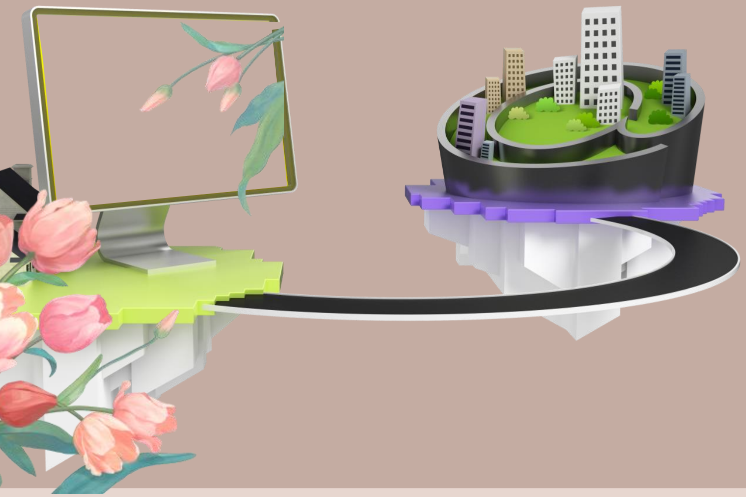
◆시험 자료에 적합하여 성능 확인하기.

```
y_test_pred_svm = best_svm_model.predict(X_test)
confusion_matrix(y_test, y_test_pred_svm)
```

```
array([[18,  0,  0],
       [ 0, 10,  0],
       [ 0,  3, 14]])
```

```
best_svm_model.score(X_test, y_test)
```

```
0.9333333333333333
```



다음시간안내

제12강

Association rule analysis