분류_개요, 최근접이웃

- 권수태 교수

JEONJU UNIVERSITY
The Place for Superstars

1. 분류개요

❖ 개요

- 대표적인 지도학습
- 학습데이터로 주어진 데이터의 피처와 레이블값(결정값,클래스값)을 학습해 모델 생성하고, 모델에 새로운 데이터값이 주어졌을때 레이블값을 예측하는 것
- 회귀 문제에서는 데이터와 잘 적합한 선을 찾으려고 하는데,
- 분류 문제에서는 입력 데이터가 어느 클래스에 속하는 지를 판단할 수 있는 결정 경계(Decision Boundary)를 찾는 것
- 예측할 클래스의 개수에 따라 분류 문제는 또 이진 분류(Binary Classification)와 다중 클래스 분류(Multiclass Classification)로 나눔

1. 분류개요

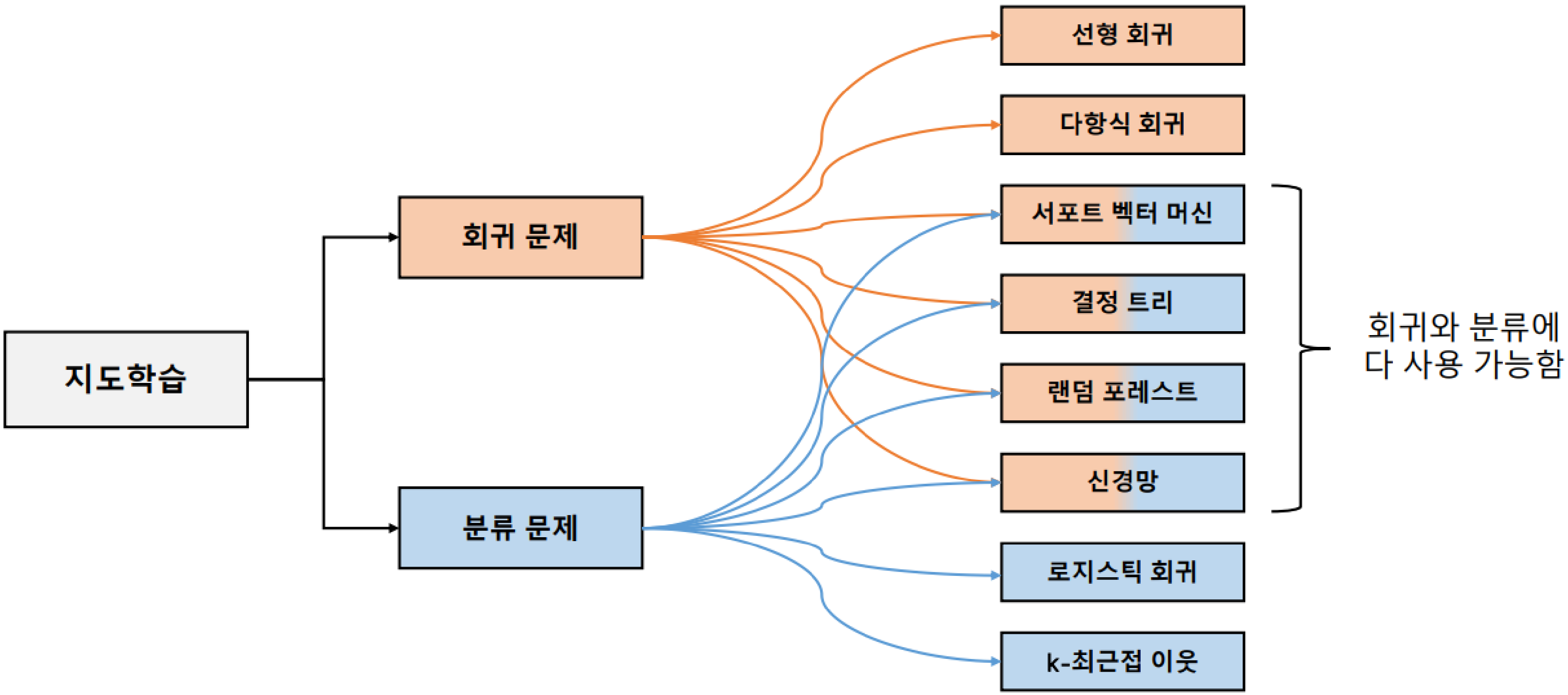
❖ 개요

- ➢ 분류알고리즘
  - ✓ 근접 거리를 기본으로 하는 최소 근접 알고리즘
  - ✓ 베이즈 통계와 생성모델에 기반한 나이브 베이즈
  - ✓ 독립변수와 종속변수의 선형 관계성에 기반한 로지스틱 회귀
  - ✓ 개별 클래스 간의 최대 분류 마진을 효과적으로 찾아주는 서포트 벡터 머신
  - ✓ 데이터 균일도에 따른 규칙 기반의 결정트리
  - ✓ 서로 다른 (또는 같은) 머신러닝 알고리즘을 결합한 앙상블
  - ✓ 심층 연결 기반의 신경망

1. 분류개요

❖ 머신러닝 알고리즘의 유형

➢ 지도학습

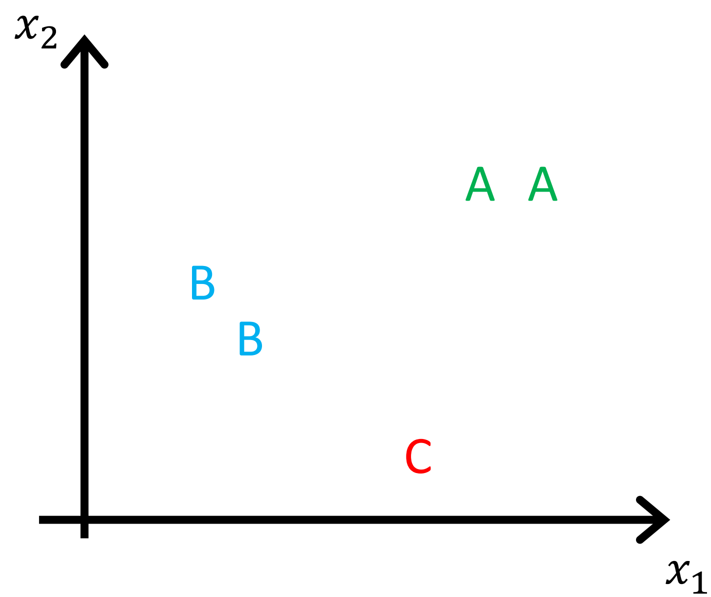


1. 분류개요

❖다중 클래스 분류 문제

- 날씨
  - ✓ 맑음, 흐림, 비, 눈
- 이메일 카테고리
  - ✓ 할일, 가족, 친구, 여행, 취미
- 성적

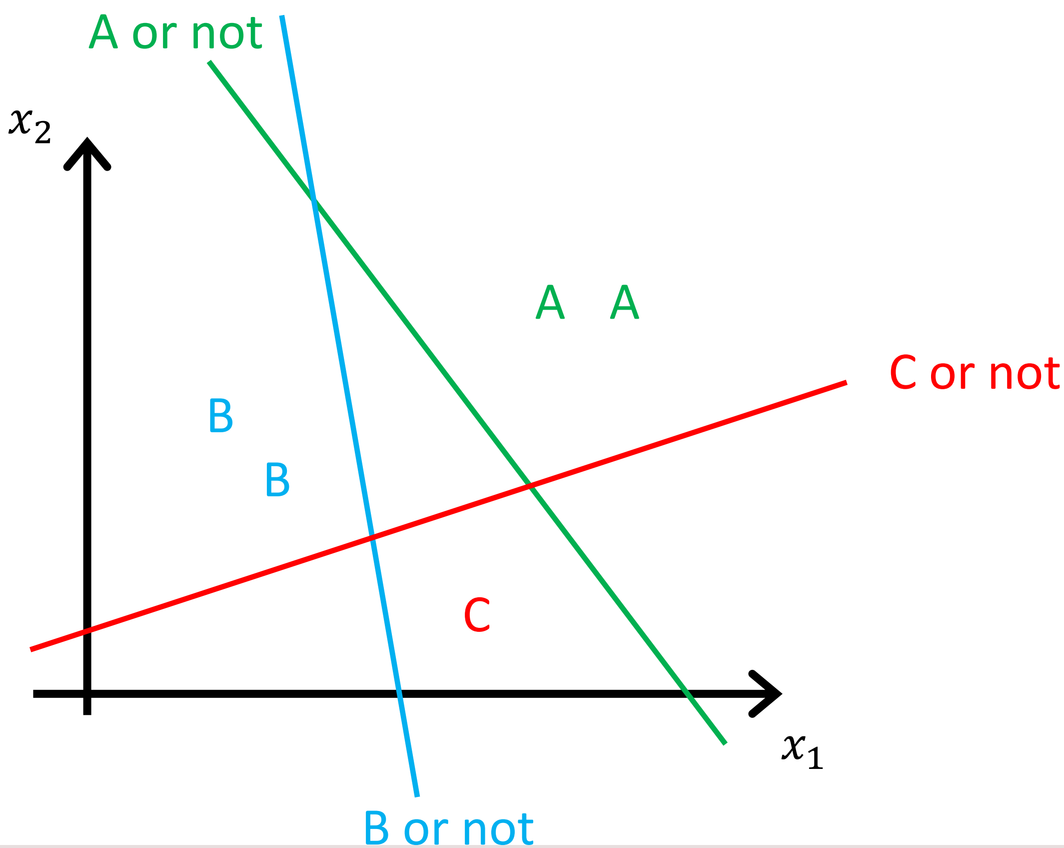
공부시간 (x1)	출석(x2)	성적(y)
10	5	A
9	5	A
3	3	B
2	4	B
7	1	C



JEONJU
UNIVERSITY

❖다중 클래스 분류 문제

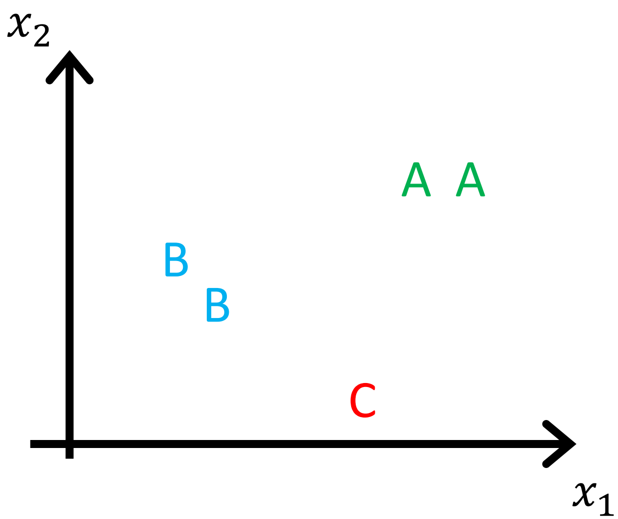
➢ 이진 분류 방법을 이용하여 다중 클래스 분류는 어떻게 할까?

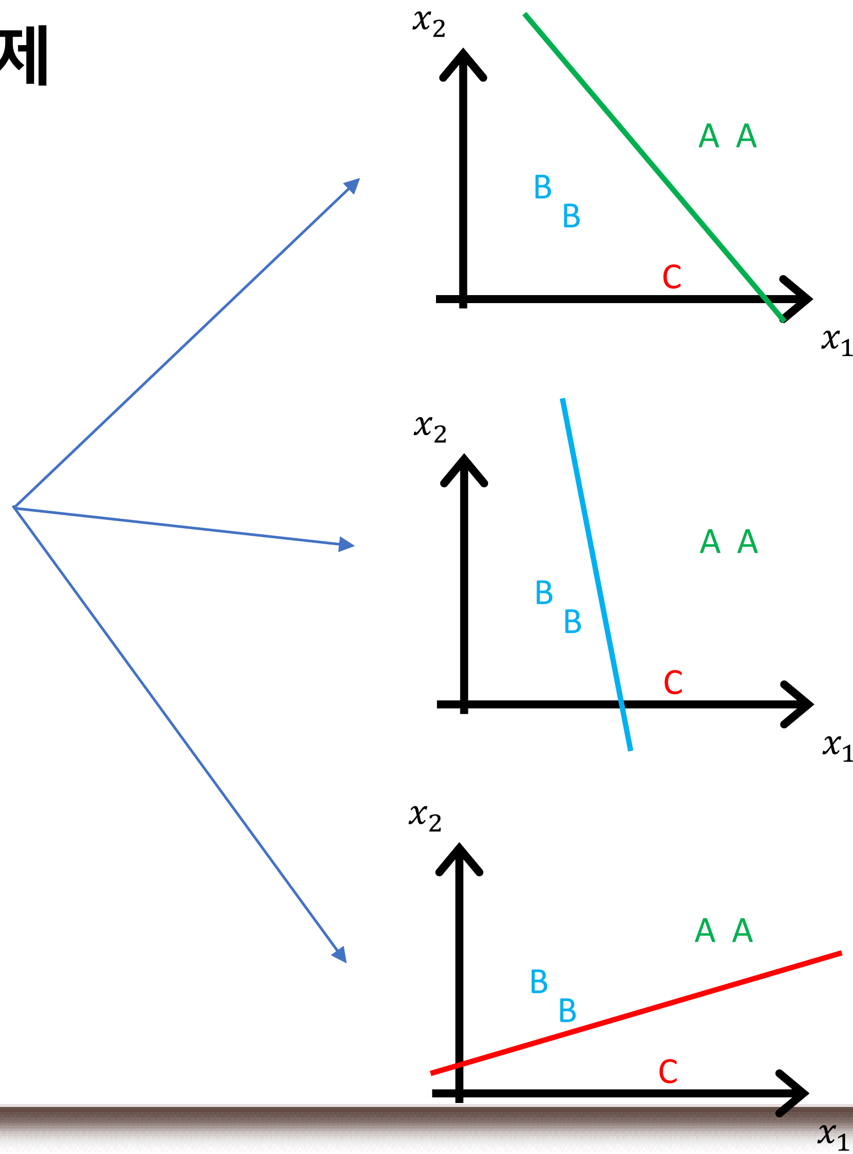


1. 분류개요

❖다중 클래스 분류 문제

➢ **One-vs-all**





Class 1 : A
Class 2 : B
Class 3 : C
$y \in \{1,2,3\}$

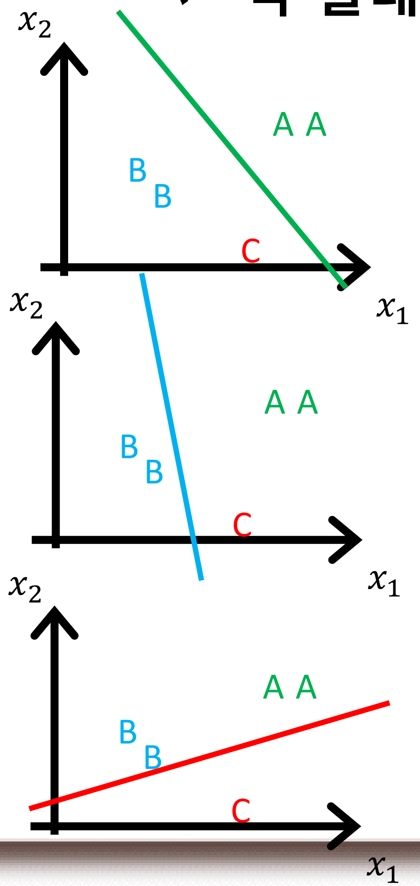
JEONJU UNIVERSITY

1. 분류개요

❖다중 클래스 분류 문제

➢ One-vs-all 알고리즘

✓ **각 클래스에 대해 로지스틱 회귀 분류모델 학습**



$h^{(1)}(x)$

Class 1 : x → [선형회귀 모델 → Sigmoid 함수] →

$h^{(2)}(x)$

Class 2 : x → [선형회귀 모델 → Sigmoid 함수] → $P(y=2|x;\Theta)$

$h^{(3)}(x)$

Class 3 : x → [선형회귀 모델 → Sigmoid 함수] → $P(y=3|x;\Theta)$

1. 분류개요

❖다중 클래스 분류 문제

➢ **Multinomial classification**

공부시간 (x1)	출석 (x2)	성적 (y)
10	5	A
9	5	A
3	3	B
2	4	B
7	1	C

$$\begin{bmatrix} \theta_0^{(1)} & \theta_1^{(1)} & \theta_2^{(1)} \\ \theta_0^{(2)} & \theta_1^{(2)} & \theta_2^{(2)} \\ \theta_0^{(3)} & \theta_1^{(3)} & \theta_2^{(3)} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \theta_0^{(1)} x_0 + \theta_1^{(1)} x_1 + \theta_2^{(1)} x_2 \\ \theta_0^{(2)} x_0 + \theta_1^{(2)} x_1 + \theta_2^{(2)} x_2 \\ \theta_0^{(3)} x_0 + \theta_1^{(3)} x_1 + \theta_2^{(3)} x_2 \end{bmatrix}$$

❖다중 클래스 분류 문제

➢ **Multinomial classification**

$$\sigma\left(\begin{bmatrix}\theta_0^{(1)}x_0+\theta_1^{(1)}x_1+\theta_2^{(1)}x_2\\ \theta_0^{(2)}x_0+\theta_1^{(2)}x_1+\theta_2^{(2)}x_2\\ \theta_0^{(3)}x_0+\theta_1^{(3)}x_1+\theta_2^{(3)}x_2\end{bmatrix}\right)=\begin{bmatrix}P(y=1|x;\Theta^{(1)})\\ P(y=2|x;\Theta^{(2)})\\ P(y=3|x;\Theta^{(3)})\end{bmatrix}$$

➢ **다중 클래스인 경우 효율적으로 가설함수의 값을 [0,1] 제한**

✓ **Sigmoid 함수 대신 선형가설 값에 대한 확률적 표현 적용**

$$\begin{bmatrix}\theta_0^{(1)}x_0+\theta_1^{(1)}x_1+\theta_2^{(1)}x_2\\ \theta_0^{(2)}x_0+\theta_1^{(2)}x_1+\theta_2^{(2)}x_2\\ \theta_0^{(3)}x_0+\theta_1^{(3)}x_1+\theta_2^{(3)}x_2\end{bmatrix}=\begin{bmatrix}2.0\\1.0\\0.1\end{bmatrix}\longrightarrow\mathcal{F}\begin{bmatrix}2.0\\1.0\\0.1\end{bmatrix}=\begin{bmatrix}0.66\\0.24\\0.10\end{bmatrix}$$

1. 분류개요

❖다중 클래스 분류 문제

- ➢ **Softmax 함수**
  - ✓ **Sigmoid 함수를 대체**
  - ✓ **모든 가설값을 0~1로 제한**
  - ✓ **전체 클래스에 대한 합이 1 : 확률 정규화**

$$\mathcal{F}\begin{bmatrix}2.0\\1.0\\0.1\end{bmatrix} \longrightarrow \boxed{S\left(y^{(i)}\right) = \frac{e^{y^{(i)}}}{\sum_i e^{y^{(i)}}}} \longrightarrow \begin{bmatrix}0.66\\0.24\\0.10\end{bmatrix}$$

score　　　　　　　　probability

1. 분류개요

❖다중 클래스 분류 문제

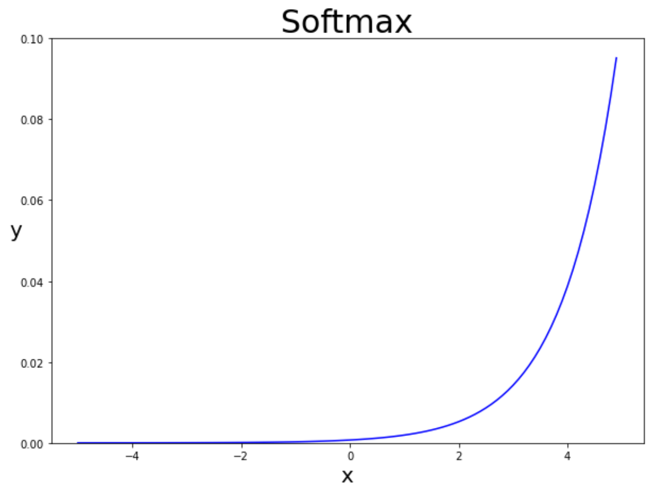
➢ **Softmax 함수**

✓ **총 클래스가 3 일때,**

$$softmax(z) = \left[\frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}}, \frac{e^{z_2}}{e^{z_1}+e^{z_2}+e^{z_3}}, \frac{e^{z_3}}{e^{z_1}+e^{z_2}+e^{z_3}}\right]$$

$$= [p_1, p_2, p_3]$$

단조증가함수이기 때문에 소프트맥스에
들어가는 인자들의 대소관계는 변하지 않음
지수함수를 적용하면 아무리 작은 값의
차이라도 확실히 구별될 정도로 커짐
미분은 원래값과 동일



❖다중 클래스 분류 문제

➢ 비용함수

✓ **Cross-entropy**

$$\mathcal{F}\begin{bmatrix}2.0\\1.0\\0.1\end{bmatrix} \longrightarrow \left(y^{(i)}\right) = \frac{e^{y^{(i)}}}{\sum_i e^{y^{(i)}}}$$

$$\begin{bmatrix}0.66\\0.24\\0.10\end{bmatrix} \qquad D(S,L) = -\sum_i L^{(i)}\log(s^{(i)}) \qquad \begin{bmatrix}1.0\\0.0\\0.0\end{bmatrix}$$

S : 예측값

L : 정답

1. 분류개요

❖ 오차행렬/혼동행렬 (Confusion Matrix)

➢ 분류 문제에서 예측 오류가 얼마인지, 어떤 유형의 오류가 발생하고 있는지를 함께 나타내는 지표

<table>
<tr><td colspan="2" rowspan="2">데이터 총수
= P + N</td><td colspan="2">예측 상황</td></tr>
<tr><td>Positive (PP)</td><td>Negative (PN)</td></tr>
<tr><td rowspan="2">실제 상황</td><td>Positive (P)</td><td>True positive (TP)</td><td>False negative (FN)
[Type II error]</td></tr>
<tr><td>Negative (N)</td><td>False positive (FP)
[Type I error]</td><td>True negative (TN)</td></tr>
</table>

1. 분류개요

❖ 오차행렬/혼동행렬 (Confusion Matrix)

➢ 분류 문제에서 예측 오류가 얼마인지, 어떤 유형의 오류가 발생하고 있는지를 함께 나타내는 지표

True Positive

예측상황

실제상황

	A	B	C	D
A				
B				
C				
D				

1. 분류개요

❖ 오차행렬/혼동행렬 (Confusion Matrix)

➢ 분류 문제에서 예측 오류가 얼마인지, 어떤 유형의 오류가 발생하고 있는지를 함께 나타내는 지표

True Negative for A

예측상황

실제상황

	A	B	C	D
A				
B				
C				
D				

JEONJU UNIVERSITY

1. 분류개요

❖ 오차행렬/혼동행렬 (Confusion Matrix)

➢ 분류 문제에서 예측 오류가 얼마인지, 어떤 유형의 오류가 발생하고 있는지를 함께 나타내는 지표

False Positive for A

예측상황

		A	B	C	D
	A				
실제상황	B				
	C				
	D				

JEONJU
UNIVERSITY

1. 분류개요

❖ 오차행렬/혼동행렬 (Confusion Matrix)

➢ 분류 문제에서 예측 오류가 얼마인지, 어떤 유형의 오류가 발생하고 있는지를 함께 나타내는 지표

False Negative for A

예측상황

실제상황

	A	B	C	D
A				
B				
C				
D				

1. 분류개요

❖ 오차행렬/혼동행렬 (Confusion Matrix)

- Accuracy: 예측한 전체 데이터들 중에 정확히 예측한 데이터 수의 비율

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

- Error rate: 예측한 전체 데이터들 중에 잘못 예측한 데이터 수의 비율

$$Error\ rate = \frac{FP+FN}{TP+TN+FP+FN}$$

- Precision: 양성으로 예측한 데이터들 중에 실제로 양성인 데이터의 비율

$$Precision = \frac{TP}{TP+FP}$$

- Recall: 실제로 양성인 데이터를 양성으로 정확히 예측한 경우의 비율

$$Recall = \frac{TP}{TP+FN}$$

- Specificity: 실제로 음성인 데이터를 음성으로 정확히 예측한 경우의 비율

$$Specificity = \frac{TN}{FP+TN}$$

2. 최근접 이웃

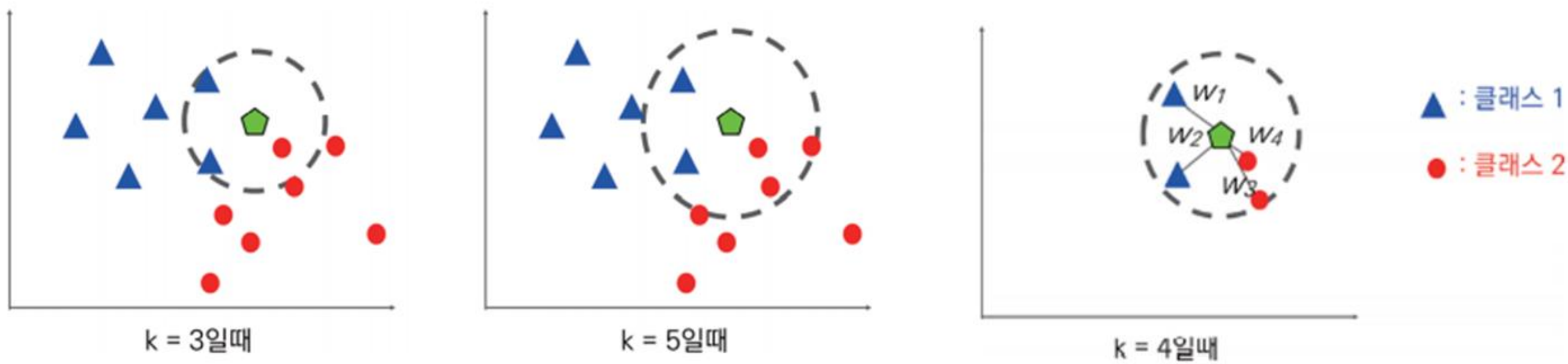
❖ k-최근접 이웃(k-Nearest Neightbor)

- ➢ 매우 단순한 방법으로 새로운 관측치를 분류 및 예측할 수 있는 방법
  - ✓ 모델 생성 없이 인접 데이터를 분류/예측에 사용
  - ✓ 즉, 관측치만을 이용하여 새로운 데이터에 대한 예측을 진행
- ➢ 모든 데이터를 메모리에 저장한 후 이를 바탕으로 예측시도
- ➢ 비슷한 특성을 가진 데이터는 비슷한 범주에 속하는 경향이 있다는 가정하에 사용

2. 최근접 이웃

❖ k-최근접 이웃(k-Nearest Neightbor)

➢ 특징 공간에 분포하는 데이터에 대하여 k개의 가장 가까운 이웃을 살펴보고 다수결 방식으로 데이터의 레이블을 할당 하는 분류방식

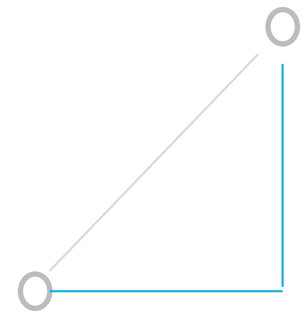


$$w_i = \frac{1}{d^2_{(new,x_i)}}$$

2. 최근접 이웃

❖ k-최근접 이웃(k-Nearest Neightbor)

- K 결정
  - ✓ 매우 작을 경우는 데이터의 지역적 특성을 지나치게 반영하여 분류 정확도가 낮아질 수 있음(overfitting)
  - ✓ 매우 클 경우는 underfitting 발생가능



- 거리측정
  - ✓ 데이터 정규화 혹은 표준화 필요
  - ✓ 유클리드 (Euclidean Distance) $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$
  - ✓ 맨하탄 (Manhattan Distance) $d = |x_2 - x_1| + |y_2 - y_1|$
  - ✓ 마할라노비스(Mahalanobis Distance): 데이터들의 분포를 고려하여 정규화 한 뒤에 유클리드 거리를 계산 $d = \sqrt{(X - Y)^T \Sigma^{-1} (X - Y)}$

JEONJU UNIVERSITY

2. 최근접 이웃

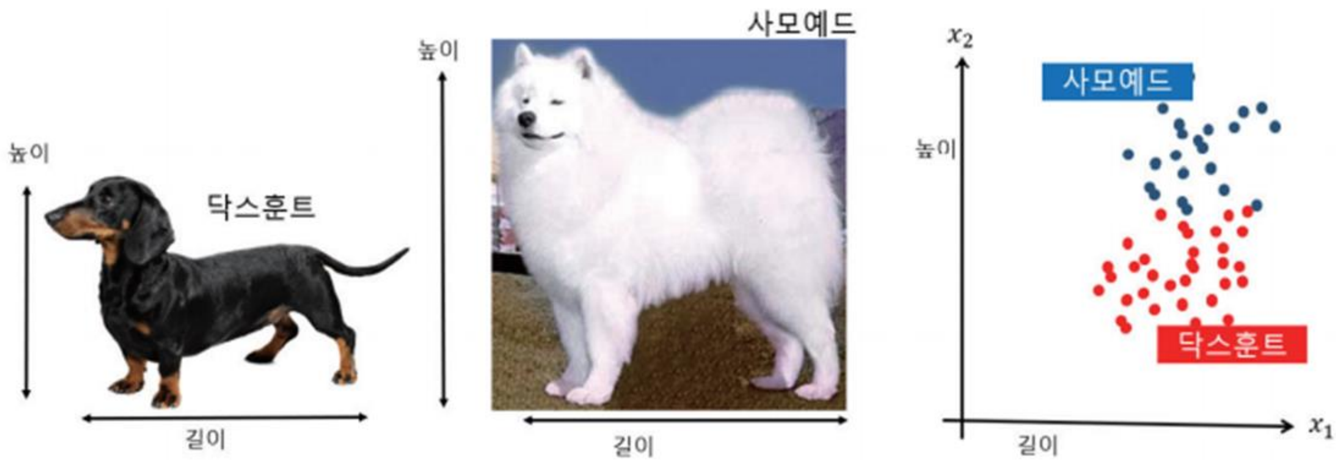
❖ k-최근접 이웃(k-Nearest Neightbor)

- 장점
  - ✓ 알고리즘이 매우 단순하고 직관적이며, 사전 학습이나 특별한 준비 시간이 필요 없다는 점
  - ✓ 다른 알고리즘에 비해 구현하기가 쉬움
- 단점
  - ✓ 특징 공간에 있는 모든 데이터에 대한 정보가 필요
  - ✓ 데이터 인스턴스, 클래스, 특징의 요소들의 개수가 많다면, 많은 메모리 공간과 계산 시간이 필요
  - ✓ 모델을 생성하지 않기 때문에 특징과 클래스 간 관계를 이해하는데 제한적

2. 최근접 이웃

❖ k-최근접 이웃(k-Nearest Neightbor)

- 개의 품종 중에는 닥스훈트dachshund와 사모예드samoyed라는 종이 있는데, 두 종은 몸의 높이와 길이 비율이 서로 다름
- 일반적으로 닥스훈트는 몸통 길이에 비해서 높이가 낮고, 사모예드의 경우 몸통 길이와 높이가 비슷한 특징을 가짐
- 몸통의 길이와 높이 특징feature을 각각 x_1, x_2라고 두고 이 개들의 표본 집합에 대하여 x_1, x_2를 측정하면 아래 그림의 오른쪽과 같은 산포도 그래프를 얻을 수 있음



2. 최근접 이웃

❖ k-최근접 이웃(k-Nearest Neightbor)

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

# 닥스 훈트의 몸 길이와 몸 높이
dach_length = [77, 78, 85, 83, 73, 77, 73, 80]
dach_height = [25, 28, 19, 30, 21, 22, 17, 35]

# 사모예드의 몸 길이와 몸 높이
samo_length = [75, 77, 86, 86, 79, 83, 83, 88]
samo_height = [56, 57, 50, 53, 60, 53, 49, 61]

d_data = np.column_stack((dach_length, dach_height))
d_label = np.zeros(len(d_data))   # 닥스훈트는 0으로 레이블링
s_data = np.column_stack((samo_length, samo_height))
s_label = np.ones(len(s_data))   # 사모예드는 1로 레이블링
```

2. 최근접 이웃

❖ k-최근접 이웃(k-Nearest Neighbtbor)

```
newdata = [[79, 35]]

dogs = np.concatenate((d_data, s_data))
labels = np.concatenate((d_label, s_label))

dog_classes = {0:'Dachshund', 1:'Samoyed'}

k = 3    # k를 3으로 두고 kNN 분류기를 만들어 보자
knn = KNeighborsClassifier(n_neighbors = k)
knn.fit(dogs, labels)
y_pred = knn.predict(newdata)
print('데이터', newdata, ', 판정 결과:', dog_classes[y_pred[0]])
```


Thank You !

JEONJU UNIVERSITY
The Place for Superstars