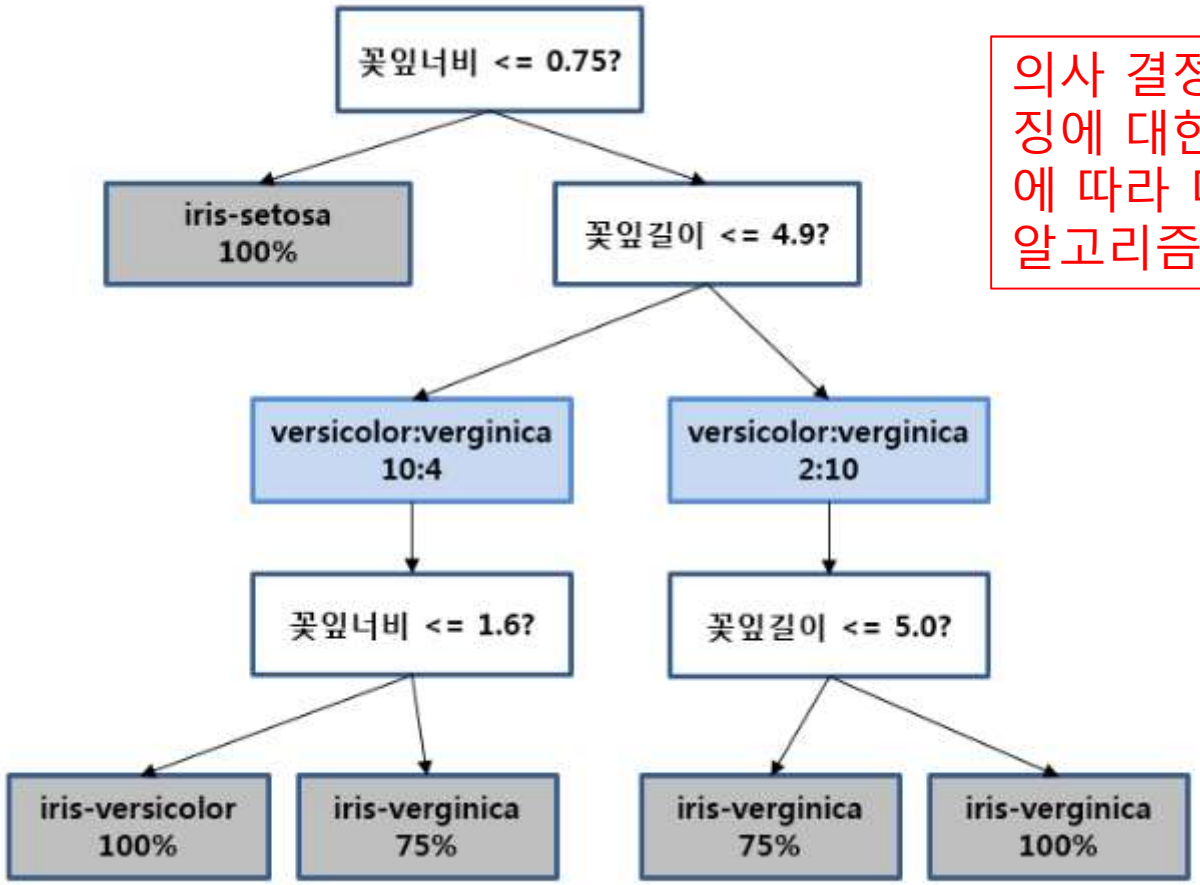
의사결정트리(Decision Tree)

의사결정트리(Decision Tree)

의사결정나무(decision tree)는 여러 가지 규칙을 순차적으로 적용하면서 독립 변수 공간을 분할하는 분석 모형이다. 즉, 어떤 항목에 대한 관측값(꽃잎너비,꽃잎길이 등.)과 목표값(품종)을 연결시켜주는 모형으로 나무(Tree) 모양의 그래프로 표현될 수 있다.



의사 결정 나무는 데이터의 특징에 대한 질문을 하면서 응답에 따라 데이터를 분류해가는 알고리즘이다.

의사결정트리(Decision Tree)

–기계학습 중 하나로 특정 항목에 대한 의사 결정 규칙(Decision rule)을 나무 형태로 분류해 나가는 분석 기법

– 분류의 기준을 선택하는 방법, 나무 구조를 만드는 방법, 나무구조를 정리하는 가지치기 방법 등에 따라 다양한 알고리즘이 개발되어 있다.

– 통계학에 기반한 CART 및 CHAID (카이스퀘어, T검정, F검정 등을 사용)와 기계학습 계열인 C4.5, C5.0 및 ID3 (엔트로피, 정보 이득 등을 사용) 등의 알고리즘이 존재

의사결정트리(Decision Tree)

(2) 목적

– **세분화(Segmentation)** : 데이터를 비슷한 특성을 갖는 몇 개의 그룹으로 분할해 그룹별 특성을 발견하는 경우 또는 각 고객이 어떤 집단에 속하는지를 파악하고자 하는 경우
ex) 시장세분화, 고객세분화

– **분류(Classification)** : 관측개체를 여러 예측변수들에 근거해 목표변수의 범주를 몇 개의 등급으로 분류하고자 하는 경우
ex) 고객을 신용도에 따라 우량/불량으로 분류

– **예측(Prediction)** : 자료에서 규칙을 찾아내고 이를 이용해 미래의 사건을 예측하고자 하는 경우 ex) 고객속성에 따라서 대출한도액을 예측

– **차원축소 및 변수선택(Data reduction and variable screening)** : 매우 많은 수의 예측변수 중에서 목표변수에 큰 영향을 미치는 변수들을 골라내고자 하는 경우

– **교효작용효과의 파악**(Interaction effect identification) : 여러 개의 예측변수들을 결합해 목표변수에 작용하는 규칙(교호작용효과)을 파악하고자 하는 경우

– **범주의 병합 또는 연속형 변수의 이산화(Binning)** : 범주형 목표변수의 범주를 소수의 몇 개로 병합하거나 연속형 목표변수를 몇 개의 등급으로 이산화 하고자 하는 경우

의사결정트리(Decision Tree)

(3) 구성요소

- 뿌리마디(root node): 시작되는 마디로 전체 자료를 포함
- 자식마디(child node): 하나의 마디로부터 분리되어 나간 2개 이상의 마디들
- 부모마디(parent node): 주어진 마디의 상위마디
- 끝마디(terminal node): 자식마디가 없는 마디

의사결정트리(Decision Tree)

(4) 의사결정분석 방법

– **분석단계 : 성장단계 -> 가지치기 단계 -> 타당성 평가 단계 -> 해석 및 예측 단계**

① 성장단계 : 각 마디에서 적절한 최적의 분리규칙을 찾아서 나무를 성장시키는 과정, 적절한 정지규칙을 만족하면 중단한다.

② **가지치기 단계** : 오차를 크게 할 위험이 높거나 부적절한 추론규칙을 가지고 있는 가지 또는 불필요한 가지를 제거

- **나무의 가지치기** : 너무 큰(깊이가 깊은) 나무모형은 자료를 과대적합하고 너무 작은 나무모형은 과소적합 할 위험이 있어 마디에 속한 자료가 일정 수 이하일 경우, 분할을 정지하고 가지치기를 실시(사전가지치기, 사후가지치기)

– **불순도에 따른 분할 측도** : 카이제곱 통계량, 지니지수, 엔트로피지수

③ 타당성 평가 단계 : 이익도표(gain chart), 위험도표(risk chart), 혹은 시험자료를 이용하여 의사결정나무를 평가

④ 해석 및 예측 단계 : 구축된 나무모형을 해석, 예측 모형을 설정한 후 예측

의사결정트리(Decision Tree)

(5) 불순도

- 의사결정트리 학습에서 각 노드에서 분기하기 위한 최적의 질문은 정보이득(information Gain)이라는 값이 최대가 되도록 만들어주는 것이 핵심이다. 어느 특정 노드에서 m개의 자식 노드로 분기되는 경우 정보이득은 아래의 식으로 정의한다.

리프 노드가 순수해질 때까지 반복

부모 노드의 데이터 불순도

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j)$$

부모 노드의 불순도와 자식 노드의 불순도 합의 차이

자식 노드의 데이터 불순도의 합

여기서, ***f***는 분기를 시키기 위한 트레이닝 데이터의 특성값(예를 들어, 꽃잎 길이 또는 꽃잎 너비)이며 ***Dp***는 부모 노드에 존재하는 데이터 세트, ***Dj***는 *j*번째 자식노드에 존재하는 데이터 세트입니다. ***I(Dp)***는 ***Dp***의 데이터 불순도(impurity), ***I(Dj)***는 ***Dj***의 데이터 불순도를 의미합니다. 그리고 ***Np***와 ***Nj***는 ***Dp***의 데이터 개수와 ***Dj***의 데이터의 개수입니다.

- **데이터 불순도** : 데이터가 제대로 분류되지 않고 섞여 있는 정도를 말하는데, 정보이득 IG는 자식노드의 데이터의 불순도가 작으면 작을수록 커지게 된다.

의사결정트리(Decision Tree)

(5) 불순도

- 정보이득 IG 는 자식 노드의 데이터 불순도가 작으면 작을수록 커지게 된다.

- 계산을 단순화하기 위해 보통 의사결정트리에서 분기되는 자식 노드의 개수는 2개로 하는데, 이를 이진 의사결정트리(Binary Decision tree)라고 부른다. 특정 노드에서 분기되는 2개의 자식노드를 각각 L과 R첨자를 나타내서 표현하여 위 식을 단순화하면 아래와 같다.

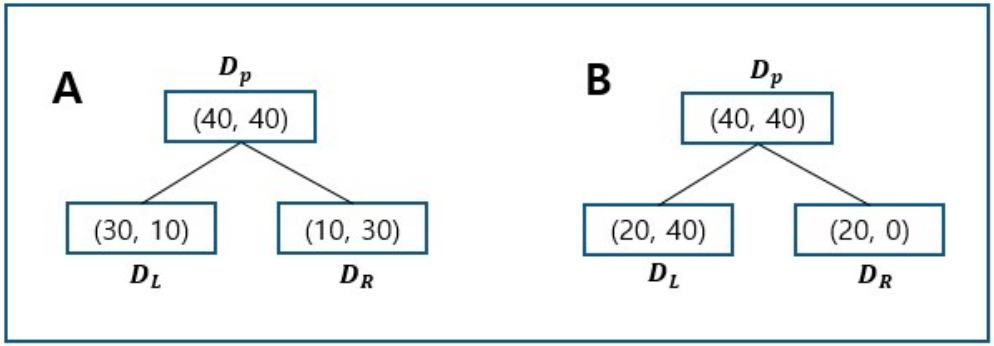
$$IG(D_p, f) = I(D_p) - \frac{N_L}{N_p} I(D_L) - \frac{N_R}{N_p} I(D_R)$$

자식 노드의 데이터 불순도

의사결정트리(Decision Tree)

(5) 불순도

- **데이터 불순도 예)**



의사결정트리 A의 경우 불순도 *I(Dp), I(DL), I(DR)* 및 정보이득 *IG* 계산

$$I_E(D_p) = 1 - 0.5 = 0.5$$

$$I_E(D_L) = 1 - \frac{3}{4} = 0.25$$

$$I_E(D_R) = 1 - \frac{3}{4} = 0.25$$

따라서 정보이득 계산식에 의한 정보이득 값은 다음과 같습니다.

$$IG_E = 0.5 - \frac{40}{80} \times 0.25 - \frac{40}{80} \times 0.25 = 0.25$$

의사결정트리 B의 경우 불순도 *I(Dp), I(DL), I(DR)* 및 정보이득 *IG* 계산

$$I_E(D_p) = 1 - 0.5 = 0.5$$

$$I_E(D_L) = 1 - \frac{4}{6} = \frac{1}{3}$$

$$I_E(D_R) = 1 - 1 = 0$$

따라서 정보이득계산 식에 의한 정보이득 값은 다음과 같습니다.

$$IG_E = 0.5 - \frac{60}{80} \times \frac{1}{3} - 0 = 0.25$$

의사결정트리(Decision Tree)

(5) 불순도

- **데이터 불순도 예)**

이제 정보이득을 계산하는 원리는 대충 이해했겠지요.

비슷한 방법으로 데이터 불순도를 계산하는 방법으로 지니 인덱스, 엔트로피를 적용하였을 경우 A, B에서 정보이득은 다음과 같이 계산됩니다.

지니 인덱스로 계산한 경우

- A에서 정보이득 : 0.125
- B에서 정보이득 : 약 0.16

엔트로피로 계산한 경우

- A에서 정보이득 : 0.19
- B에서 정보이득 : 0.31

의사결정트리에서 정보이득이 최대가 되는 것을 채택해야 하므로, 분류오류를 적용하였을 때는 A, B가 모두 같은 값이 나와서 어떤 것을 선택하더라도 무관했지만, 지니 인덱스로 계산하거나 엔트로피로 계산한 경우에는 B가 A보다 정보이득 값이 크므로 B를 선택하게 될 것입니다.

의사결정트리(Decision Tree)

(6) 불순도 측정방법

- **데이터 불순도를 측정하는 방법은 아래와 같이 3가지가 있다.**

① 지니 인덱스(Gini Index)

② 엔트로피(entropy)

③ 분류 오류(Classification Error)

지니 인덱스

$$I_G(t) = 1 - \sum_{i=1}^{c} p(i\backslash t)^2$$

엔트로피

$$I_H(t) = -\sum_{i=1}^{c} p(i\backslash t)\log_2 p(i\backslash t)$$

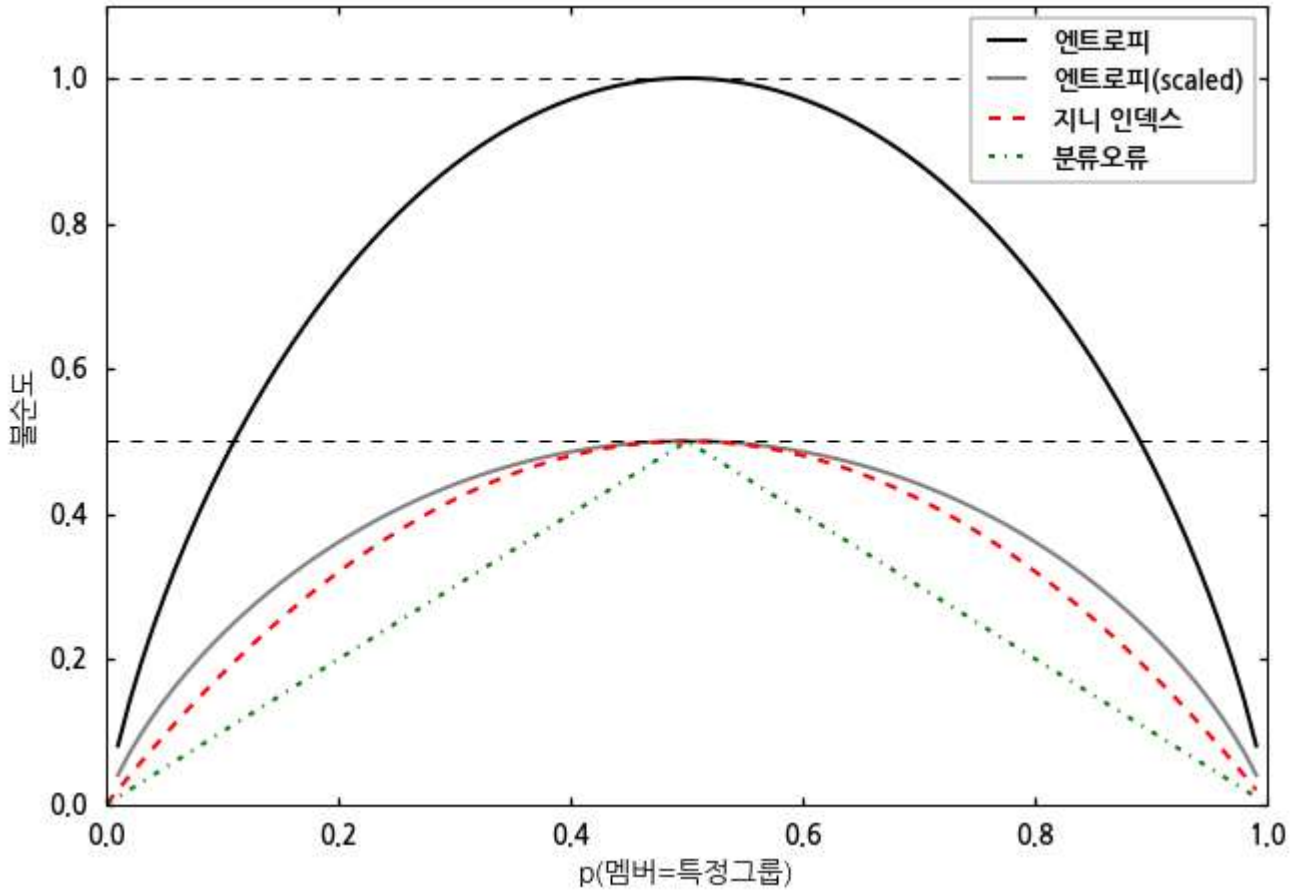
분류오류

$$I_E(t) = 1 - \max\{p(i\backslash t)\}$$

의사결정트리(Decision Tree)

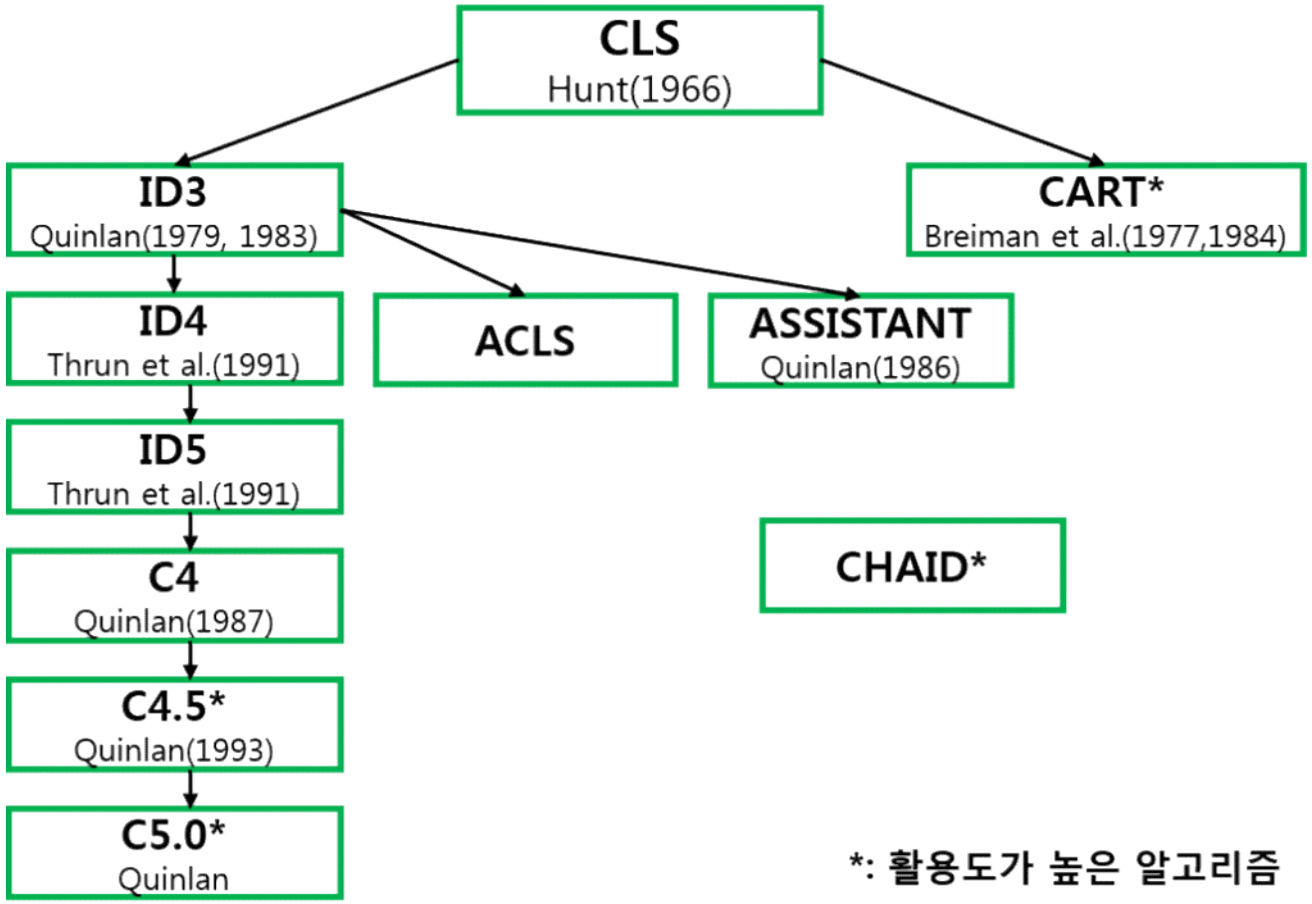
(6) 불순도 측정방법

- **데이터 불순도**



의사결정트리(Decision Tree)

(7) 의사결정분석 알고리즘의 종류



의사결정트리(Decision Tree)

(7) 의사결정분석 알고리즘의 종류

– 알고리즘들은 기본적인 가지 생성 방식은 유사하며 가지를 분리하는 방식(분리에 사용될 변수 및 기준을 선택하는 방식)에서의 약간의 차이를 가짐

알고리즘	분리기준(목표변수)	비고
ID3	Entropy	다지분리(범주)
C4.5	Information Gain	다지분리(범주) 및 2진분리(연속)
C5.0	Information Gain	C4.5와 거의 유사
CHAID	카이제곱(범주), F검정(연속)	통계적 접근 방식
CART	Gini Index(범주), 분산의 차이(연속)	통계적 접근 방식, 항상 2진 분리

의사결정트리(Decision Tree)

(7) 의사결정분석 알고리즘의 종류

① CART(Classification and Regression Tree)

전체 데이터셋을 갖고 시작하여 반복해서 두 개의 자식 노드를 생성하기 위해 모든 예측 변수를 사용하여 데이터 셋의 부분집합을 쪼갬으로써 의사결정 트리를 생성함

$$Gini\ Index = \sum_{j=1}^{c} P(j)\big(1 - P(j)\big) = 1 - \sum_{j=1}^{c} P(j)^2 = 1 - \sum_{j=1}^{c} \left(\frac{n_j}{n}\right)^2$$

$$\Delta G = \text{분류전 G} - \left\{\left(\frac{\mathrm{n_L}}{\mathrm{N}}\right)\mathrm{G}_{\text{Left Child Node}} + \left(\frac{\mathrm{n_R}}{\mathrm{N}}\right)\mathrm{G}_{\text{Right Chid Node}}\right\}$$

N: Parent Node의 데이터 개수
n_L: 왼쪽 Child Node의 데이터 개수
n_R: 오른쪽 Child Node의 데이터 개수

n : 현재 노드에 있는 데이터의 개수
n_j: 현재 노드에서 class j에 속한 데이터 개수
c : 전체 class의 수

의사결정트리(Decision Tree)

(7) 의사결정분석 알고리즘의 종류

① CART(Classification and Regression Tree) : rpart라이브러리

전체 데이터셋을 갖고 시작하여 반복해서 두 개의 자식 노드를 생성하기 위해 모든 예측 변수를 사용하여 데이터 셋의 부분집합을 쪼갬으로써 의사결정 트리를 생성함

ex) 남자 500명 중 응답자(100명), 비응답자(400명)

2번에 걸쳐 응답자를 뽑을 때, 2번 모두 응답자일 확률은 (100/500)^2 = (1/5)^2

2번에 걸쳐 비응답자를 뽑을 때, 2번 모두 비응답자일 확률은 (400/500)^2=(4/5)^2

여기서 Gini Index 는 1 – (1/5)^2 – (4/5)^2로

임의로 2개의 케이스(사람)를 뽑을 때, 2개의 케이스가 각각 다른 목적변수 값을 가질 확률로 해석 할 수 있음

CART는 Gini Index가 작아지는 방향으로 움직이며, Gini Index를 가장 많이 감소시켜 주는 변수가 영향을 가장 많이 끼치는 변수(제일 위에 있는 변수)가 된다.

목적변수가 범주형인 경우 지니지수, 연속형인 경우 분산을 이용해 이진분리를 사용

개별 입력변수 뿐만 아니라 입력변수들의 선형결합들 중 최적의 분리를 찾을 수 있음

의사결정트리(Decision Tree)

(7) 의사결정분석 알고리즘의 종류

② CHAID(Chi-Squard Autiomatic Interaction Detection)

나무를 구축하는 방식이 CART와 흡사하나 데이터를 분할하는 방식에 차이가 있음

최적의 예측변수를 선택하는데 있어 Gini Index 대신 Chi-squared 검정(목표변수 : 범주형)과 F-검정(목표변수 : 연속형)을 사용함

$$\chi^2 = \sum_j \frac{(O_j - E_j)^2}{E_j}$$

즉, **입력변수가 반드시 범주형 자료여야 하며 수치형 자료는 범주형으로 변환 후 사용**

(ex 나이 -> 10대, 20대 .. 등)

의사결정트리(Decision Tree)

(7) 의사결정분석 알고리즘의 종류

③ C4.5 와 C5.0

-다지분리(multiple split)이 가능하고 범주형 입력 변수의 범주 수만큼 분리 가능

-불순도의 측도로 엔트로피 지수 사용

-엔트로피(entropy)는 주어진 데이터 집합의 혼잡도를 의미, 주어진 데이터 집합에 레코드들이 서로 다른 종류(클래스)들이 많이 섞여있으면 엔트로피가 높고, 같은 종류(클래스)의 레코드들이 많이 있으면 엔트로피가 낮음

-엔트로피는 0에서 1사이의 값을 가지며, 가장 혼잡도가 높은 상태의 값이 1, 하나의 클래스로만 구성된 상태의 값이 0임

-엔트로피가 높은 상태에서 낮은 상태가 되도록 나무 모양을 생성해 나감

ex) 10명의 회원 중 5명(우수회원), 5명(일반회원)

$$Entropy(5,5) = -\frac{5}{10}\log_2\frac{5}{10} - \frac{5}{10}\log_2\frac{5}{10} = 1.0$$

분류가 잘 안되면 불순도가 높아짐(1에 가까워짐)

의사결정트리(Decision Tree)

(7) 의사결정분석 알고리즘의 종류

③ C4.5 와 C5.0

ex) 10명의 회원 중 0명(우수회원), 10명(일반회원)

$$Entropy(10,0) = -\frac{10}{10}\log_2\frac{10}{10} - \frac{0}{10}\log_2\frac{0}{10} = 0.0$$

분류가 잘 되면 불순도가 낮아짐

ex) 14명의 회원 중 9명(우수회원), 5명(일반회원)

$$Entropy(9,4) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940$$

의사결정트리(Decision Tree)

(7) 의사결정분석 알고리즘의 종류

④ 조건부 추론나무(Conditional Inference Tree)

CART(rpart가 구현한 의사 결정 나무) 같은 의사결정 나무의 두 가지 문제를 해결한다.

1) 통계적 유의성에 대한 판단 없이 노드를 분할하는데 따른 과적합(overfitting)

2) 다양한 값으로 분할 가능한 변수가 다른 변수에 비해 선호되는 문제

-조건부 추론 나무는 조건부 분포에 따라 변수와 반응값(분류) 사이의 연관 관계를 측정하여 노드 분할에 사용할 변수를 선택한다. 또 의사 결정나무에서 노드를 반복적으로 분할하면서 발생하는 문제인 다중 가설 검정을 고려한 절차를 적용하여 적절한 시점에 노드의 분할을 중단한다.

➔다중 비교 문제는 통계적 추론을 통시에 여러 대상에 적용할 때 신뢰 구간이 참 값을 포함하지 않게 되거나 귀무 가설을 잘못 기각하는 가능성이 높아지는 현상

➔party::ctree() 사용

의사결정트리(Decision Tree)

(7) 의사결정분석 알고리즘의 종류

⑤ Information Gain(정보이득)

- Information Gain(정보이득)이란 어떤 속성을 선택함으로 인해서 데이터를 더 잘 구분하게 되는 것을 의미함

ex) 수능등급을 구분하는데 있어 수학 점수가 영어 점수보다 변별력이 더 높다고 가정하면, 수학 점수 속성이 영어 점수 속성 보다 정보이득이 높다고 할 수 있음

$$InfoGain(A) = \underbrace{I(s_1, s_2, \ldots, s_m)}_{\text{상위 노드의 엔트로피}} - \underbrace{E(\text{속성}A)}_{\text{하위 노드의 엔트로피}}$$

- 정보 이득은 상위노드의 엔트로피에서 하위노드의 엔트로피를 뺀 것을 의미함
- E(속성A)의 값은 A라는 속성을 선택했을 때 하위로 작은 m개의 노드로 나누어진다고 하면 하위 각 노드들의 엔트로피를 계산한 후 각 노드들의 레코드 개수를 가중치로 하여 엔트로피 값을 평균한 값임. 즉, InfoGain(A)의 의미는 속성 A를 선택했을 때의 정보이득 양을 계산하는 수식이며, 값이 클수록 속성 A의 변별력이 좋다는 것을 의미함

의사결정트리(Decision Tree)

(8) 의사결정분석의 장단점

장점	단점
• 설명력이 높다. • 결과에 대한 근거를 나뭇가지 형태로 추적할 수 있다. • 빠르고 변수 선택 능력이 있다. • 많은 변수들을 대상으로 종속변수에 영향이 높은 변수를 선택 할 수 있다.	• ~~종속변수가 연속형일 때 쓸 수 없다.~~ • 설명변수가 연속형일 때 낮은 예측 능력을 보일 수 있다. • 자료의 추가에 의하여 나무구조가 바뀔 수 있다. • 비선형데이터에는 적합하지 못하다.

- 모델의 시각화가 쉽고, 가독성이 높다
- 변수의 표준화 작업(scale)이 필요 없다.
- 과대적합을 해결해야 한다.
  ***너무 쪼개지는 걸 막기 위해 각 규칙이 포함된 관측 값의 최소 갯수를 제한해야 한다(minsplit)**
  * 가지치기를 통해 불필요한 판별기준을 없앤다
- (단점) 단일결정 tree는 과대적합이 발생하여 일반화 성능이 저하된다.
  통계적 유의성에 대한 판단없이 노드를 분할한다
  다양한 값으로 분할 가능한 변수가 다른 변수에 비해 선호되는 문제를 갖고 있다.

의사결정트리(Decision Tree)

(9) R 패키지

○ tree 모형 R 패키지 소개

- tree 패키지 : binary recursive partitioning (이진 반복 분할) 방법론
- rpart 패키지 : CART (classification and regression trees) 방법론

· 연속 속도 빠르지만 (엔트로피, 지니계수) 과적합화 위험성 존재

· Pruning 과정을 통해 최적화 필요 **(가지치기)**

- party 패키지 : Unbiased recursive partitioning based on permutation tests방법

· p-test를 거친 Significance를 기준으로 가지치기를 할 변수를 결정

· 별도의 가지치기를 할 필요가 없으나 입력 변수의 레벨이 31개까지로 제한

· party::ctree

의사결정트리(Decision Tree)

(9) R 패키지

-C50 패키지

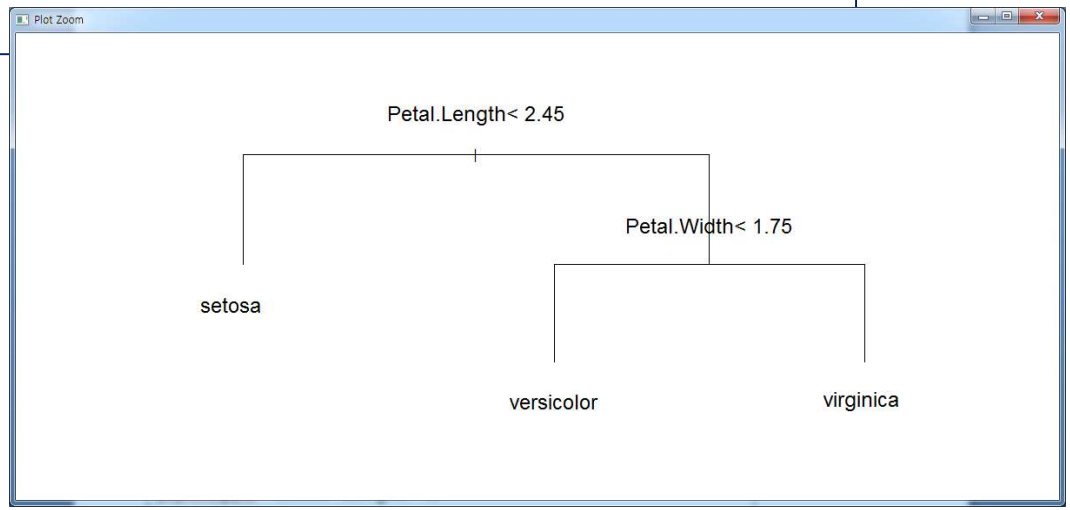
```
install.packages("C50")
library(C50)
```

의사결정트리(Decision Tree)

(9) R 패키지

① rpart 패키지 사용 실습(iris)

```
#cart(지니계수) - rpart
install.packages("rpart")
library(rpart)

help(rpart)
m<-rpart(Species~.,data=iris)
m
plot(m,margin=.2,compress=T)
text(m,cex=1.5)
```

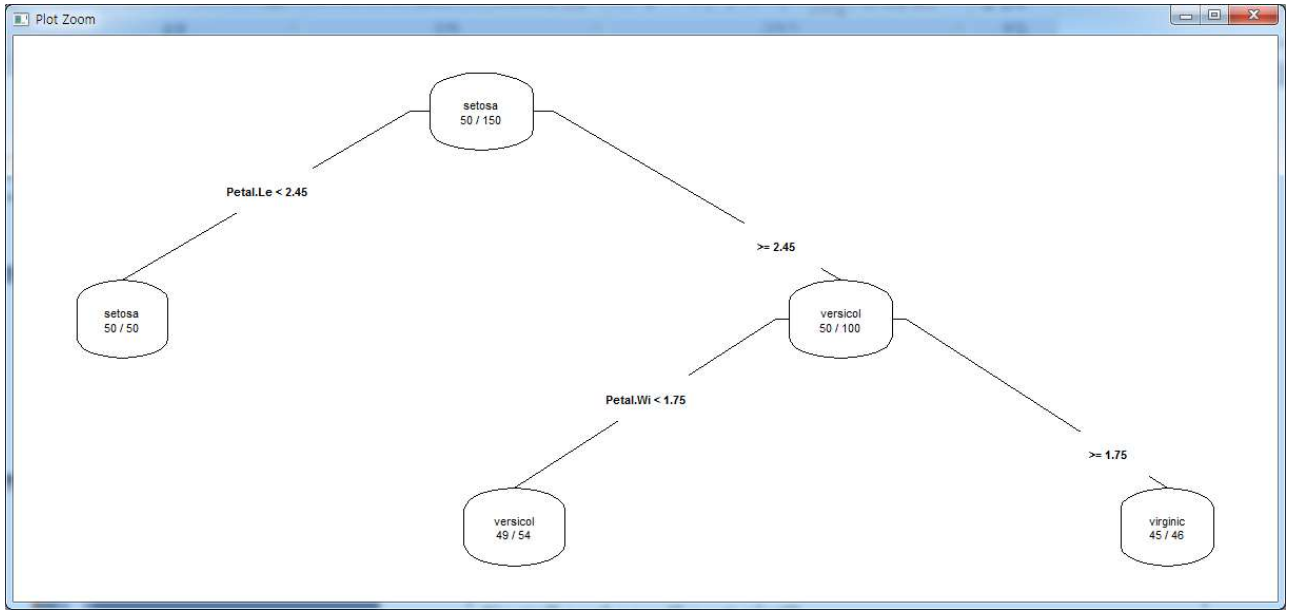


의사결정트리(Decision Tree)

(9) R 패키지

- rpart 패키지 사용 실습(iris)

```
...(이어서)

#install.packages("rpart.plot")
library(rpart.plot)
prp(m,type=4,extra=2,digits=3)
```



의사결정트리(Decision Tree)

(9) R 패키지

② Party::ctree 패키지 사용 실습(iris)

```
...(이어서)
#CTREE
>install.packages("party")
>library(party)

>m2<-ctree(Species~.,data=iris)
>m2
>plot(m2)

#노드5의 경우  setosa 다음 종의 막대 그래프가 가장 높으나 그 #
종이 무엇인지 그림으로 식별이 불가능하다. 두 번째 종은
#iris$Species의 레벨을 확인하면 versicolor임을 알수 있다.

> levels(iris$Species)
```

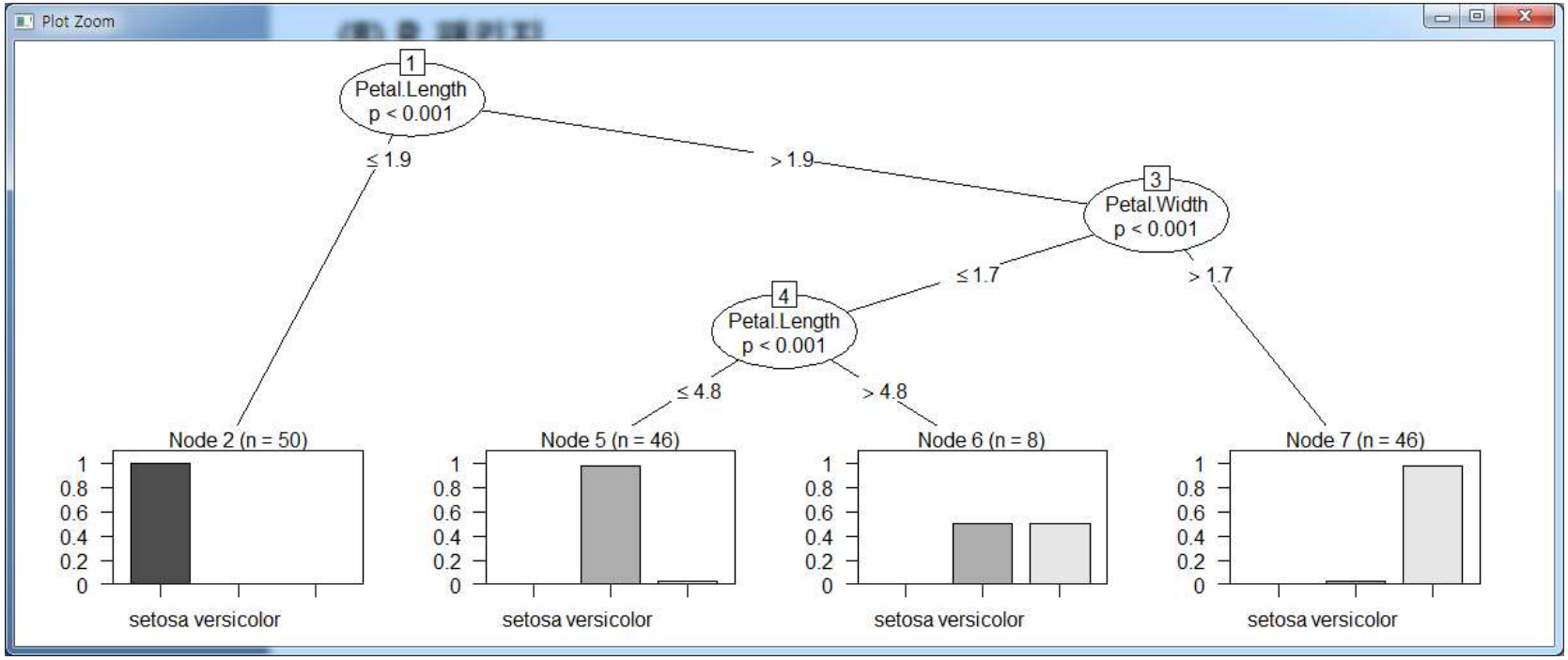
```
> levels(iris$Species)
[1] "setosa"     "versicolor" "virginica"
```

의사결정트리(Decision Tree)

(9) R 패키지

- party 패키지 사용 실습(iris)

[결과화면]



의사결정트리(Decision Tree)

(9) R 패키지

③ party 패키지 사용 실습2 (airquality)

```
#ctree()함수를 이용 의사결정트리 생성
#party 패키지 설치
install.packages("party")#설치가 안될시 R studio 관리자 권한으로 실행
library(party)
library(datasets)#R에서 기본으로 제공하는 데이터셋 이용
str(airquality)#airquality 변수확인 , New york의 대기정보

#formula생성
#온도에 영향을 미치는 변수를 알아보기위해 Temp변수를 종속변수로 두고
#나머지를 독립변수로 지정한다.
formula <- Temp ~ Solar.R + Wind + Ozone성

#분류모델 생성
air_ctree <- ctree(formula, data = airquality)
air_ctree
#분류모델 결과 시각화
 plot(air_ctree)
```

의사결정트리(Decision Tree)

(9) R 패키지

- party 패키지 사용 실습2(airquality)

결과를 보면 온도를 결정하는 1순위가 Ozone이고 2순위가 Wind라는 것을 알 수 있으며, Solar.R은 영향을 안 미치는 것을 볼 수 있다. ***는 마지막 노드** 라는 것을 의미한다.

```
> air_ctree

         Conditional inference tree with 5 terminal nodes

Response:  Temp
Inputs:  Solar.R, Wind, Ozone
Number of observations:  153

1) Ozone <= 37; criterion = 1, statistic = 56.086
  2) Wind <= 15.5; criterion = 0.993, statistic = 9.387
    3) Ozone <= 19; criterion = 0.964, statistic = 6.299
      4)*  weights = 29
    3) Ozone > 19
      5)*  weights = 69
  2) Wind > 15.5
    6)*  weights = 7
1) Ozone > 37
  7) Ozone <= 65; criterion = 0.971, statistic = 6.691
    8)*  weights = 22
  7) Ozone > 65
    9)*  weights = 26
```

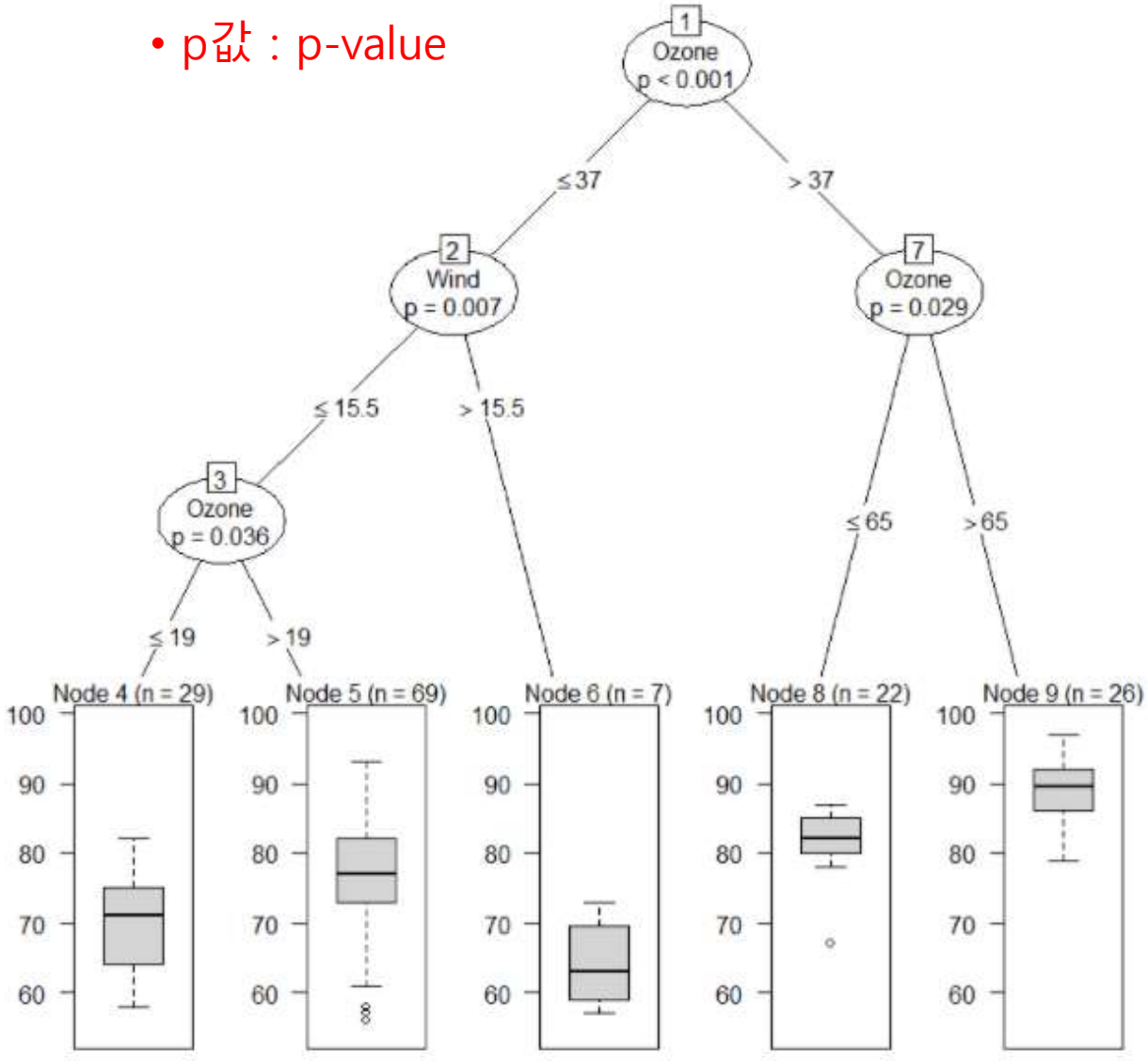
의사결정트리(Decision Tree)

(9) R 패키지

- party 패키지 사용 실습2 (airquality)

plot(air_ctree)

• p값 : p-value



• p값 : p-value

의사결정트리(Decision Tree)

(9) R 패키지

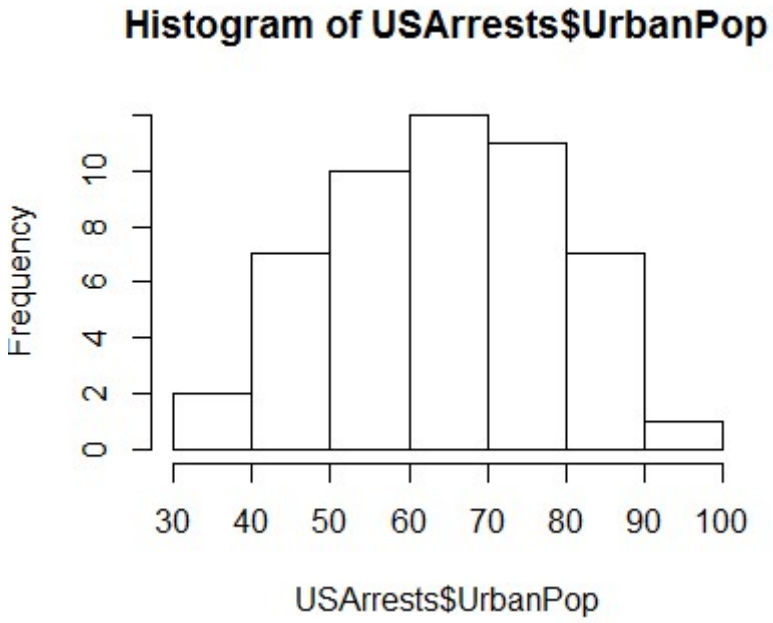
④ party 패키지 사용 실습2

```
data("USArrests")

# statistics about violent crime rates by us state.
# Murder: Murder arrests (per 100,000)
# Assault: Assault arrests (per 100,000)
# UrbanPop: Percent urban population # target으로 사용
# Rape: Rape arrests (per 100,000)

head(USArrests)
nrow(USArrests) # 50

# install.packages("rpart")
library(rpart) #load the rpart package

t1 <- rpart(UrbanPop ~ ., data = USArrests)
# Anova, Poisson, Exponential 등을 split function으로 선택해 사용 가능
t1
hist(USArrests$UrbanPop) # 타겟의 값 분포를 확인
plot(sort(USArrests$UrbanPop))
```

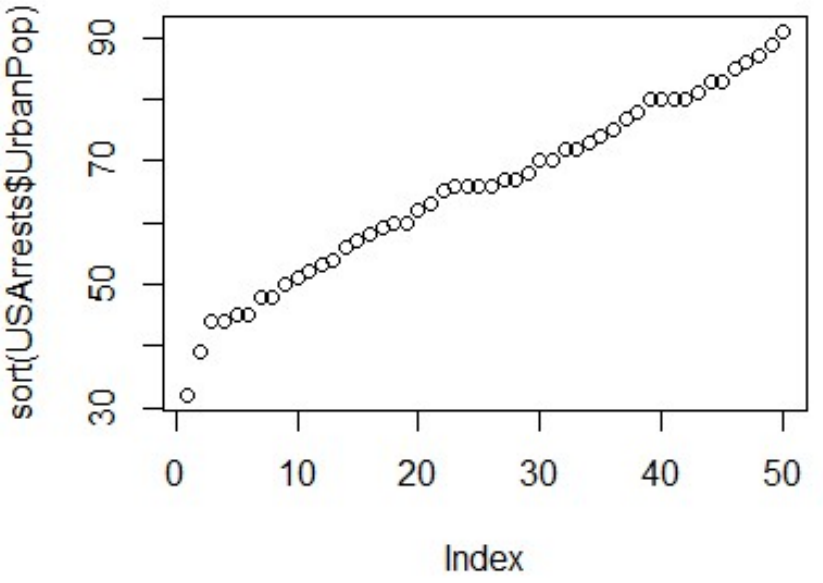
의사결정트리(Decision Tree)

(9) R 패키지

④ party 패키지 사용 실습2



hist(USArrests$UrbanPop)
# 타겟의 값 분포를 확인



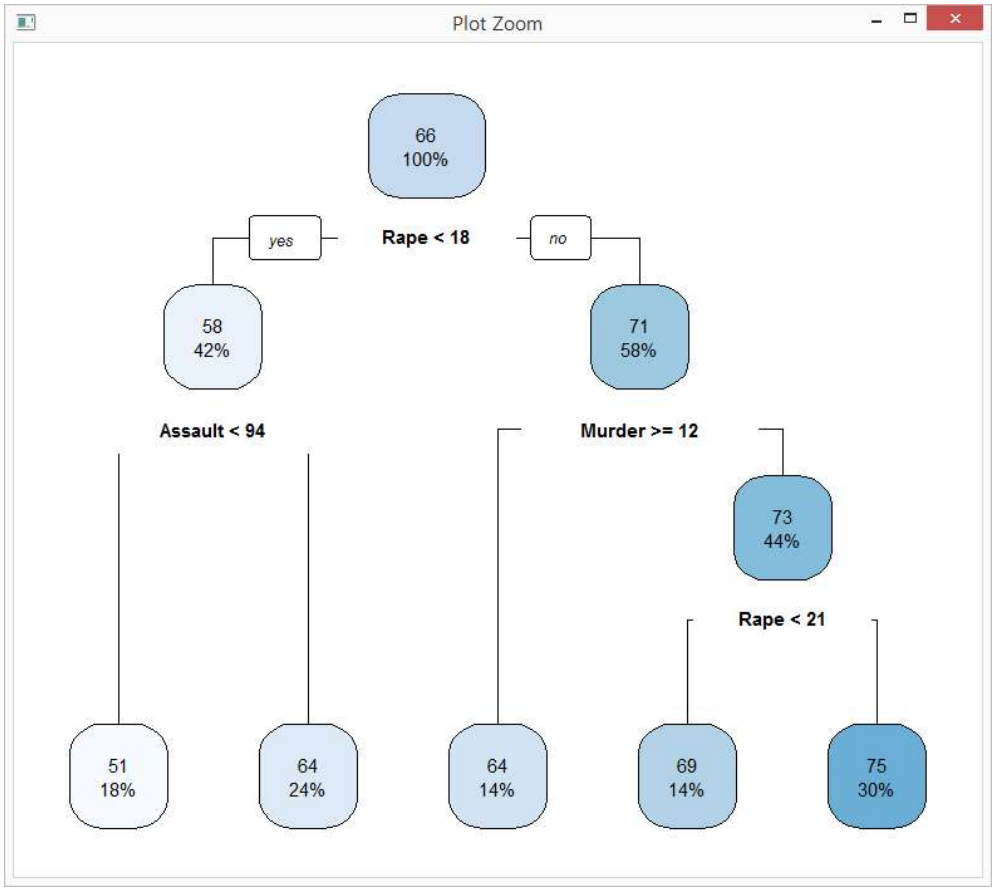
plot(sort(USArrests$UrbanPop))

의사결정트리(Decision Tree)

(9) R 패키지

⑤ party 패키지 사용 실습2

```
# 트리 플롯 그리기
#install.packages("rattle")
#install.packages("rpart.plot")

library(rattle)
library(rpart.plot)
library(RColorBrewer)

rpart.plot(t1) # 트리 플롯 그리기 기본형
```

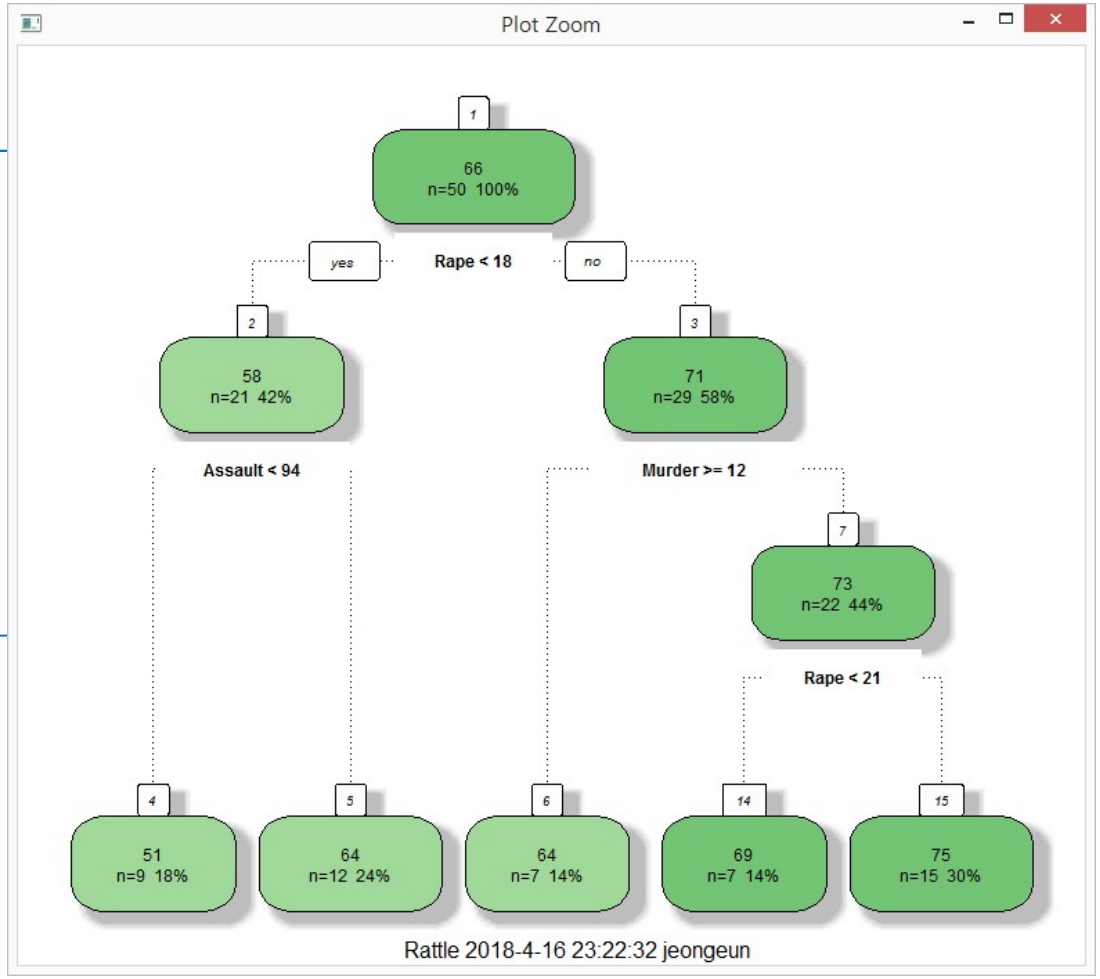


의사결정트리(Decision Tree)

(9) R 패키지

⑥ party 패키지 사용 실습2

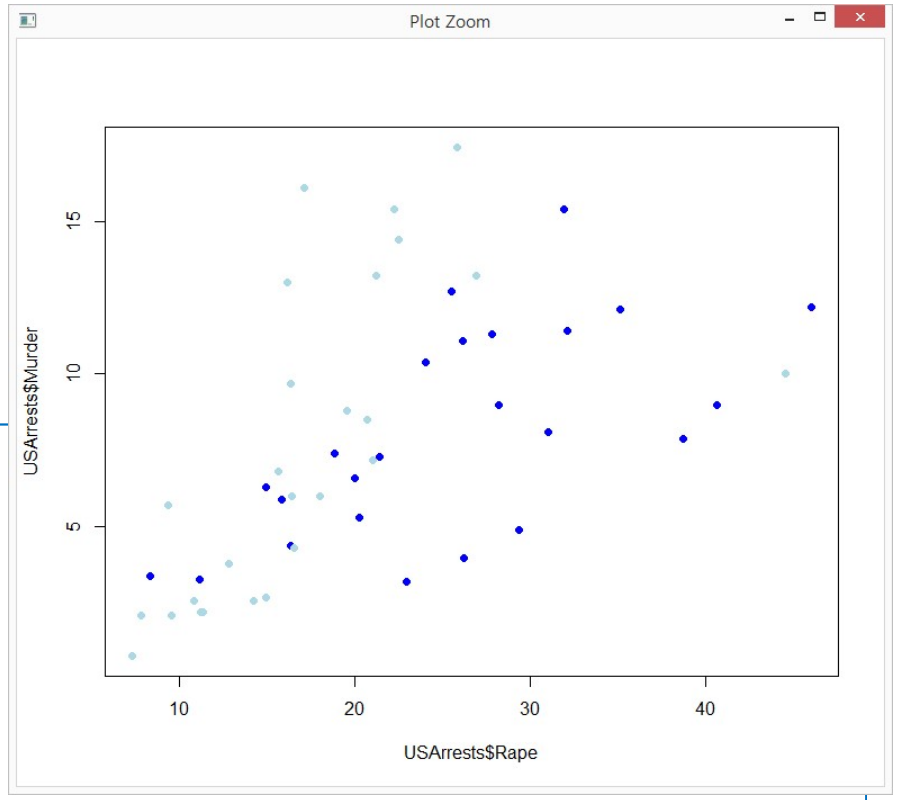
```
# 트리 플롯 그리기
#install.packages("rattle")
#install.packages("rpart.plot")

library(rattle)
library(rpart.plot)
library(RColorBrewer)

fancyRpartPlot(t1)
```



의사결정트리(Decision Tree)

(9) R 패키지

⑦ party 패키지 사용 실습2




```
# 트리 플롯 그리기
#install.packages("rattle")
#install.packages("rpart.plot")

library(rattle)
library(rpart.plot)
library(RColorBrewer)

# 살인에 비해 강간비율이 높은 주가 인구 많음
plot(USArrests$Rape, USArrests$Murder,
     col=ifelse(USArrests$UrbanPop>median(USArrests$UrbanPop),
                "blue", "lightblue"),pch=19)
```

의사결정트리(Decision Tree)

(10) 파이썬 구현

- 파이썬

```
from sklearn.tree import DecisionTreeClassifier
tree1 = DecisionTreeClassifier(criterion='entropy', max_depth=1).fit(X, y)
```

- R 코드

```
library(rpart)
tree1 = rpart(y ~ X , data)

library(party)
```

Accuracy, Precision, Recall Rate 을 이용하여 성능을 평가한다.

예를들어 1을 1일이라고 예측한 비율이 전체에서 차지하는 것이 Accuracy(Accuracy : 전체 샘플 중 클래스를 맞게 예측한 샘플 수의 비율

),

관심대상이 1인 경우 1이라고 예측한 것이 Precision(예를들어, Precision이 높아야 되는 경우 : CRM에서의 캠페인 대상고객 선정)

실제 1인 경우에 예측을 1로 한 비중을 Recall(Recall : 특정 클래스에 속한다고 예측한 샘플 중 실제로 해당 클래스에 속하는 샘플 수)

의사결정트리(Decision Tree)

(10) 파이썬 구현

- 모델평가

```
confusion_matrix(y, tree1.predict(X))
```

Accuracy, Precision, Recall Rate 을 이용하여 성능을 평가한다.

예를들어 1을 1일이라고 예측한 비율이 전체에서 차지하는 것이 Accuracy(Accuracy : 전체 샘플 중 클래스를 맞게 예측한 샘플 수의 비율

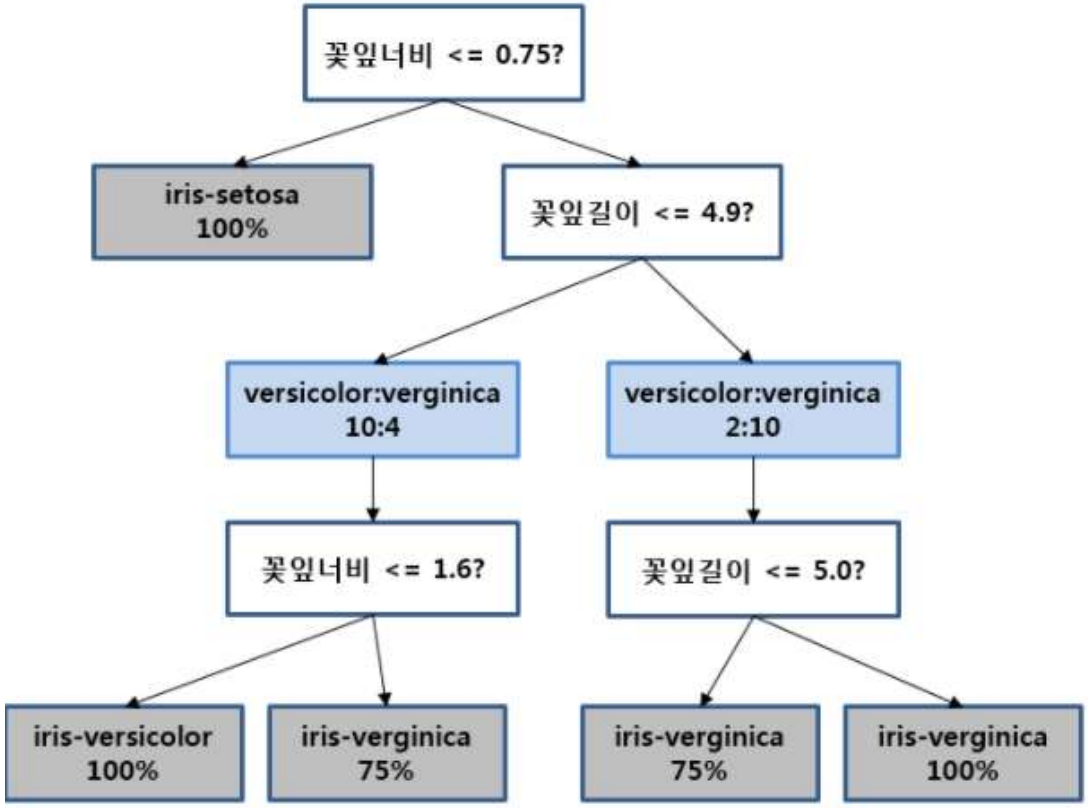
),

관심대상이 1인 경우 1이라고 예측한 것이 Precision(예를들어, Precision이 높아야 되는 경우 : CRM에서의 캠페인 대상고객 선정)

실제 1인 경우에 예측을 1로 한 비중을 Recall(Recall : 특정 클래스에 속한다고 예측한 샘플 중 실제로 해당 클래스에 속하는 샘플 수)

의사결정트리(Decision Tree)

(10) 파이썬 구현



RandomForest

랜덤포레스트(Random Forest)

• 랜덤 포레스트는 **앙상블** 학습 기법을 사용한 모델이다. 앙상블 학습은 주어진 데이터로부터 여러 개의 모델을 학습한 다음, 예측시 여러 모델의 예측 결과들을 종합해 사용하여 정확도를 높이는 기법을 말한다.

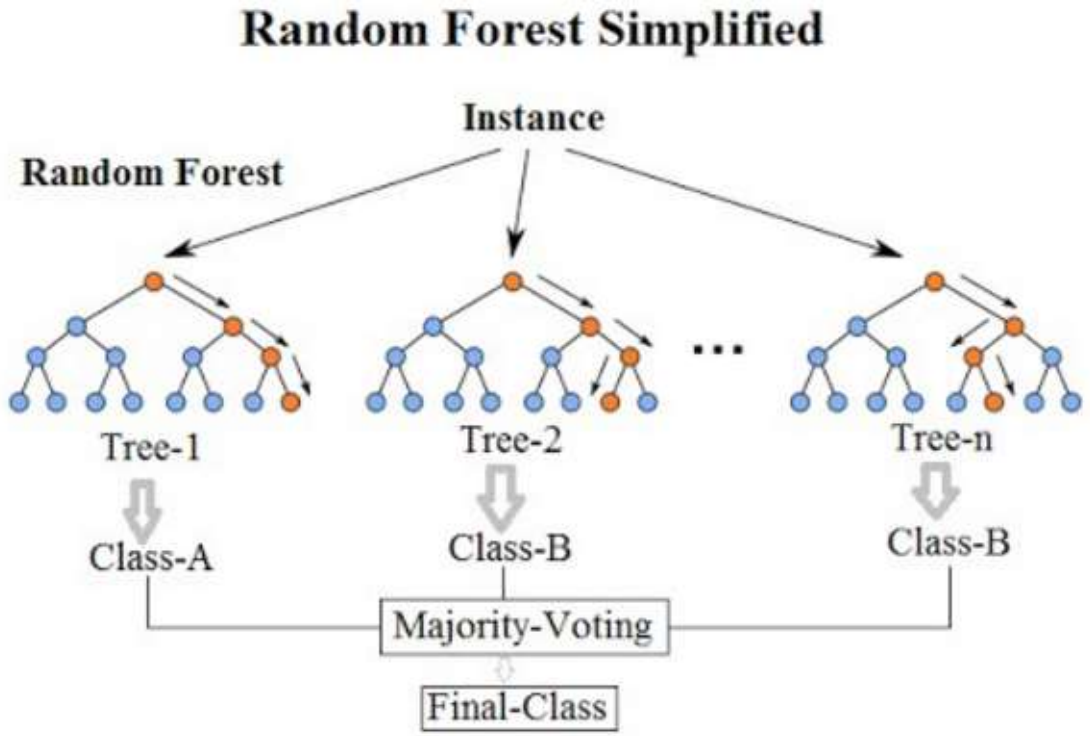
•랜덤포레스는 **두 가지 방법을 사용해 다양한 의사결정 나무를 만든다**.

1) 의사 결정 나무를 만들 때 데이터의 일부를 복원 추출로 꺼내고 해당 데이터에 대해서만 의사 결정 나무를 만드는 방식이다. 즉, 각 의사 결정 나무는 데이터의 일부만을 사용해 만들어진다.

2) 노드 내 데이터를 자식 노드로 나누는 기준을 정할 때 전체 변수가 아니라 일부 변수만 대상으로 하여 가지를 나눌 기준을 찾는 방법이다.

• 새로운 데이터에 대한 예측을 수행할 때는 여러 개의 의사 결정 나무가 내놓은 예측 결과를 투표(voting) 방식으로 합한다. 예를 들어, 총 5개의 의사 결정 나무 중 Y를 예측한 나무가 3개, N을 예측한 나무가 2개면 Y를 최종 결과로 결정하는 방식이다.

랜덤포레스트(Random Forest)



① 동일한 하나의 데이터 집합에서 **임의복원 샘플링(Bootstrap)**을 통해 **여러 개의 훈련용 데이터를 생성**한다.
② 여러 번의 학습을 통해 **여러 개의 트리를 생성**하고, 이를 결합하여 최종적으로 목표변수를 예측한다.
③ 분류모델을 검정데이터에 적용한다.

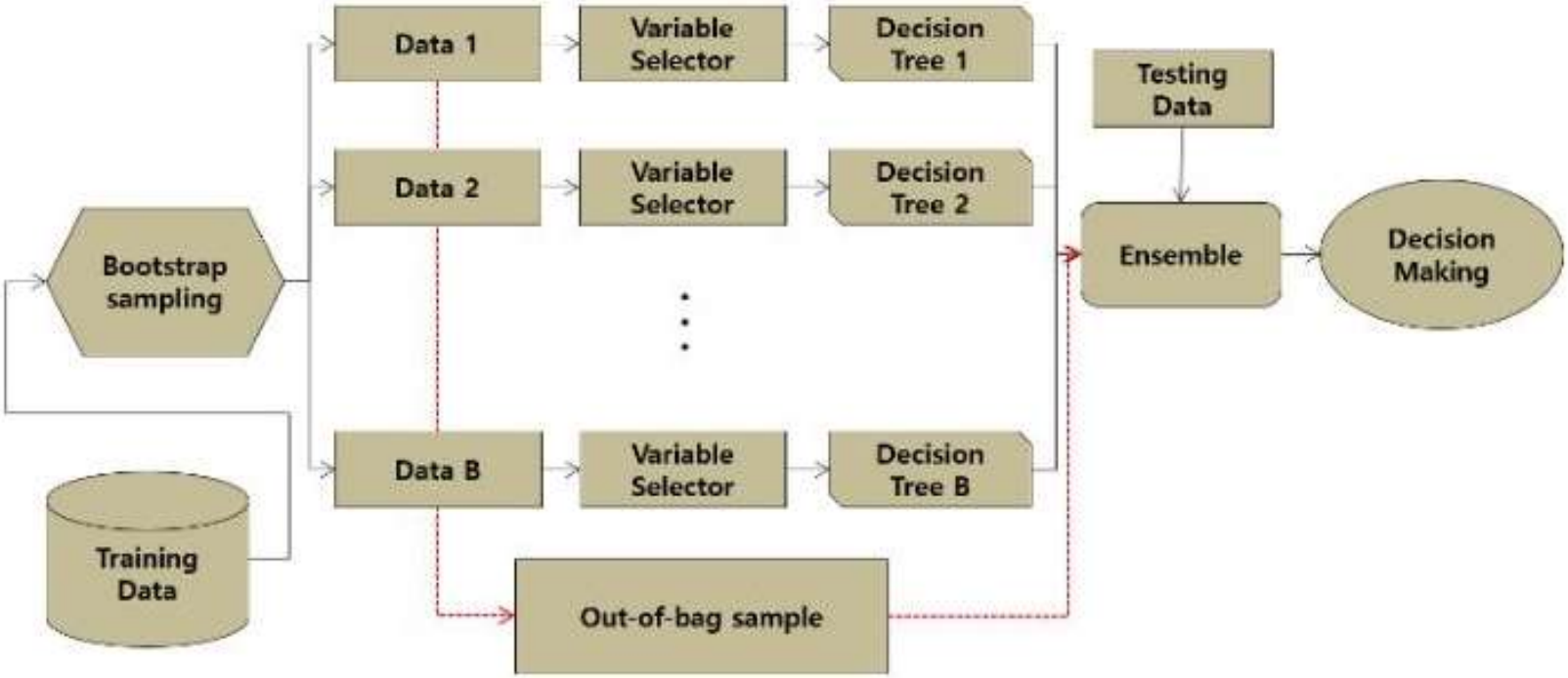
랜덤포레스트(Random Forest)

- 생성방법

FLOW CHART

• Bootstrap : 주어진 훈련 데이터의 중복을 허용해서 원래 데이터와 같은 크기의 데이터를 만드는 과정

• B : 랜덤포레스트시 생성할 하위 트리의 개수

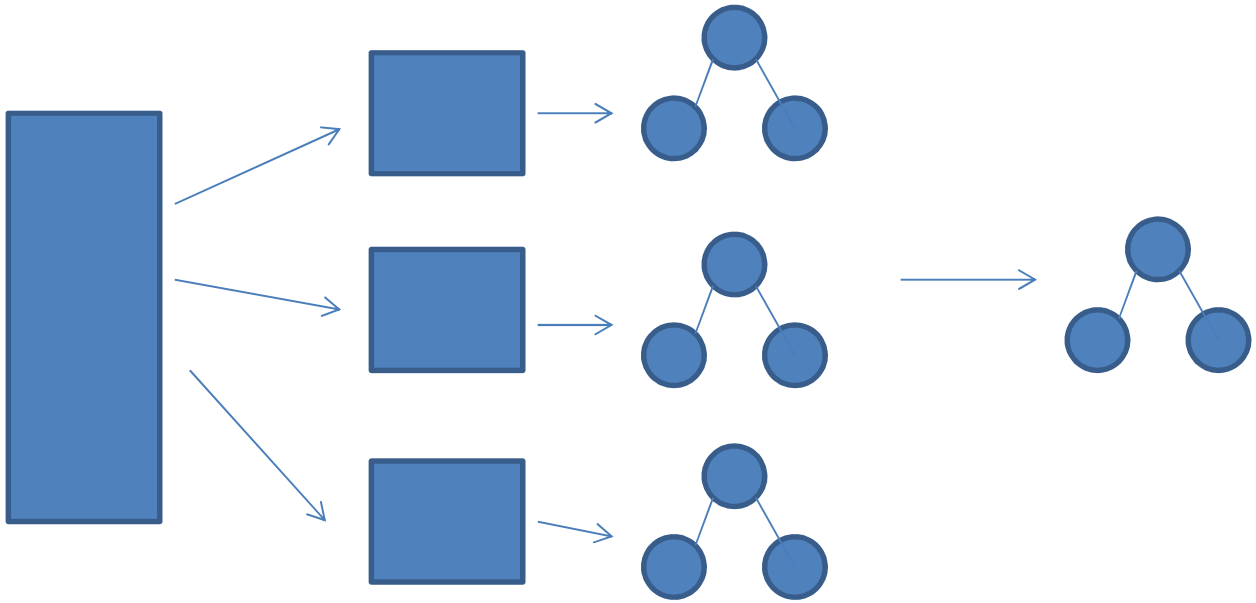


랜덤포레스트(Random Forest)

- 생성방법

• Bootstrap : 주어진 훈련 데이터의 중복을 허용해서 원래 데이터와 같은 크기의 데이터를 만드는 과정

• B : 랜덤포레스트시 생성할 하위 트리의 개수



랜덤포레스트(Random Forest)

- randomForest 패키지

```
> install.packages("randomForest")
> library(randomForest)
> rf<-randomForest(Species ~ . , data=iris)
> rf
```

```
> rf<-randomForest(Species~.,data=iris)
> rf

Call:
 randomForest(formula = Species ~ ., data = iris)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 4%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         50          0         0        0.00
versicolor      0         47         3        0.06
virginica       0          3        47        0.06
```

랜덤포레스트(Random Forest)

평가용 데이터에서의 오분류율을 예측하는 용도 및 변수 중요도를 추정하는 용도로 많이 이용

- **randomForest를 이용한 모델링**

- 랜덤 포레스트는 여러 개의 의사 결정 나무로 구현되고, 각 의사 결정 나무는 데이터의 일부만 사용한다. 랜덤 포레스트 모델을 출력하면 모델 훈련에 사용되지 않은 데이터(Bootstrap 샘플링 과정에서 추출되지 않은 데이터)를 사용한 에러 추정치가 **'OOB(Out of Bug) estimate of error rate'** 항목으로 출력된다. 아이리스에 대한 모델에서는 OOB 에러가 4%였으며, versicolor가 virginica로 예측된 경우가 3개, virginica가 versicolor로 예측된 경우가 3개 있었다.

```
> rf<-randomForest(Species~.,data=iris)
> rf

Call:
 randomForest(formula = Species ~ ., data = iris)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 4%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         50          0         0        0.00
versicolor      0         47         3        0.06
virginica       0          3        47        0.06
```

"평가용(test) 데이터가 4%정도의 오분류를 낼 것이라고 예측한다라는 의미"

예측치

실측치

랜덤포레스트(Random Forest)

- 빠른 모델링을 위한 X와 Y의 직접 지정

#포뮬러 방식 대신 변수를 직접 지정하는 방식이
#메모리를 적게 사용해 속도가 빠르다.
> rf2<-randomForest(iris[,1:4],iris[,5])

```
> rf2<-randomForest(iris[,1:4],iris[,5])
> rf2

Call:
 randomForest(x = iris[, 1:4], y = iris[, 5])
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 4%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         50          0         0        0.00
versicolor      0         47         3        0.06
virginica       0          3        47        0.06
```

랜덤포레스트(Random Forest)

- **변수의 중요도 평가**

다수의 의사 결정 나무로부터 변수가 정확도와 노드 불순도 개선에 기여하는 측면을 측정하므로 평균(Mean)이 언급되어 있다.

```
#변수의 중요도 평가
> rf3<-randomForest(Species~.,data=iris, importance=TRUE)
> importance(rf3)
```

```
> importance(rf3)
                setosa versicolor virginica MeanDecreaseAccuracy
Sepal.Length  5.701070   8.155737  8.256003             11.292997
Sepal.Width   4.378275   1.146595  4.510524              5.098796
Petal.Length 22.174380  34.340750 27.313261             33.575678
Petal.Width  21.780201  31.695396 32.055697             33.100096
             MeanDecreaseGini
Sepal.Length         9.520747
Sepal.Width          2.145927
Petal.Length        43.425924
Petal.Width         44.164295
```

정확도에서는 Petal.Legnth → …
→Sepal.Width 순으로 변수가 중요하다고 판단

MeanDecreaseGini(노드 불순도 개선)
: 이 계수에서는 Petal.Width → Petal.Length→Sepal.Legnth
→ Sepal.Width 순으로 중요

랜덤포레스트(Random Forest)

- **변수의 중요도 평가**

다수의 의사 결정 나무로부터 변수가 정확도와 노드 불순도 개선에 기여하는 측면을 측정하므로 평균(Mean)이 언급되어 있다.

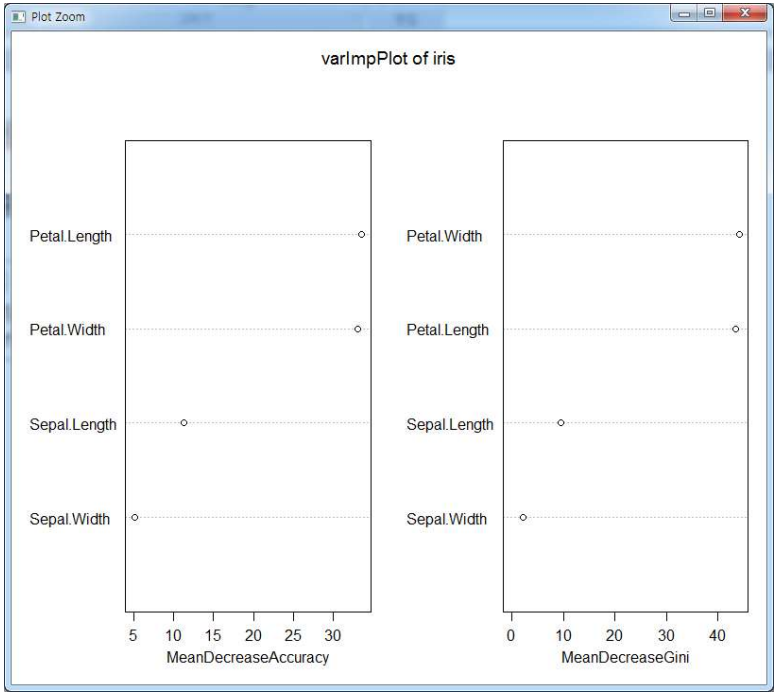
```
#변수의 중요도 평가
> rf3<-randomForest(Species~.,data=iris, importance=TRUE)
> importance(rf3)
```

- 두번째 측정치(**Mean decrease Gini**)는, (각 tree에서) **해당 variable**(=독립변수,feature)**로 splitting되는 지점에서의 decrease in impurity의 총 합, 그리고 이 값을 모든 tree에 대해 평균을 낸 것이다. 두번째 측정치 = decrease in impurity의 총 합, 이 총합을 모든 tree에 대해 평균 낸 것이다**
- **각 tree에서 해당 feature를 기준으로 분류(split)하는 지점에서의 decrease of impurity(불순도)의 총 합을 계산한 후, 이모든 tree의 값들의 평균을 낸 것이 Mean decrease Gini이다. 그 feature가 모델이 분류를 잘하는데 중요하게 작용할 수록 이 값은 커질 것이다.**

랜덤포레스트(Random Forest)

- 변수의 중요도 평가

```
#변수의 중요도 평가
> rf3<-randomForest(Species~.,data=iris, importance=TRUE)
> varImpPlot(rf3, main="varImpPlot of iris")
```



랜덤포레스트(Random Forest)

• 파라미터 튜닝

-randomForest()에는 나무 개수 **ntree**, 각 노드를 자식 노드로 나누는 기준을 정할 때 고려할 변수의 개수 **mtry** 등의 파라미터가 있다. ntree나 mtry를 조절해 모델 성능을 개선할 수 있다.

- ntree와 mtry의 다양한 조합에 대해 모델 성능을 평가해보자.

조합의 목록은 expand.grid()로 만든다.

#ntree를 10,100,200으로, mtry를 3,4로 바꿔가면서 조합한 예
> grid<-expand.grid(ntree=c(10,100,200),mtry=c(3,4))

```
> grid
  ntree mtry
1    10    3
2   100    3
3   200    3
4    10    4
5   100    4
6   200    4
```

랜덤포레스트(Random Forest)

- **파라미터 튜닝**

- 파라미터 조합을 10개로 분할한 데이터에 적용하여 모델의 성능을 평가하는 일을 3회 반복하여 최선의 파라미터를 찾아보자.

```
#파라미터 튜닝 전체 코드
>library(cvTools)
>library(foreach)
>library(randomForest)
>set.seed(719)
>K=10
>R=3
>cv<-cvFolds(NROW(iris),K=K,R=R)
>grid<-expand.grid(ntree=c(10,100,200),mtry=c(3,4))

>result<-foreach(g=1:NROW(grid),.combine=rbind) %do% {
  #모델 훈련

  #예측
}
```

랜덤포레스트(Random Forest)

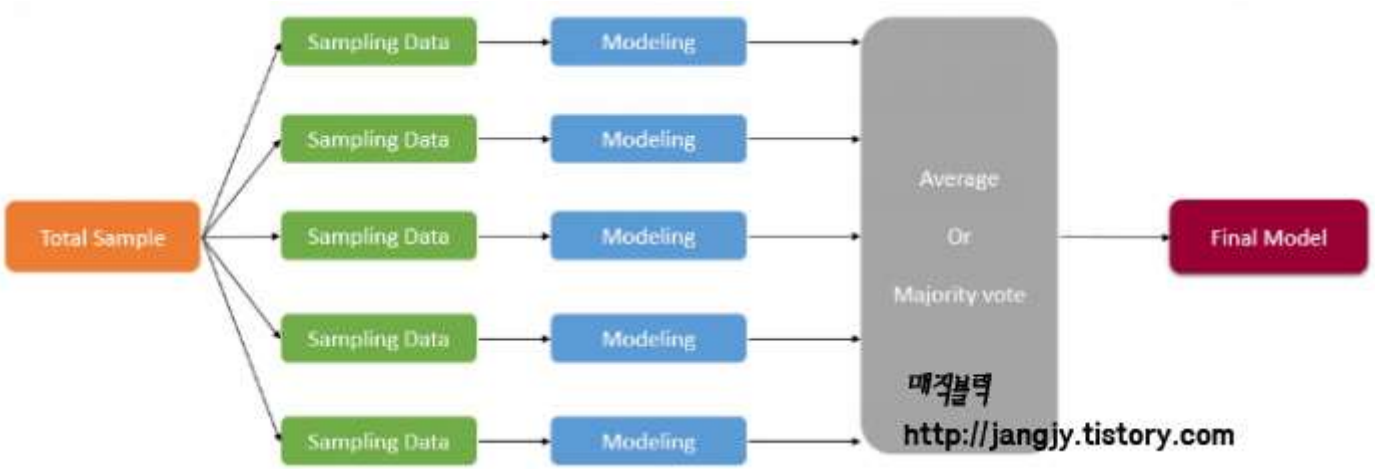
• 장단점

장점	단점
~ 대부분에 문제에 잘 수행되는 다용도 모델이다. ~ 노이즈 데이터와 결측시 데이터를 다룰 수 있다.(범주적 또는 연속적 속성) ~ 가장 중요한 속성만을 선택한다. ~ 극단적으로 큰 속성이나 예제를 가진 데이터를 다룰 수 있다.	~ 결정 트리와 달리 모델이 쉽게 해석되지 않는다. ~ 데이터에 대한 모델을 조절하는 많은 작업이 필요하다.

배깅 & 부스팅

- 배깅

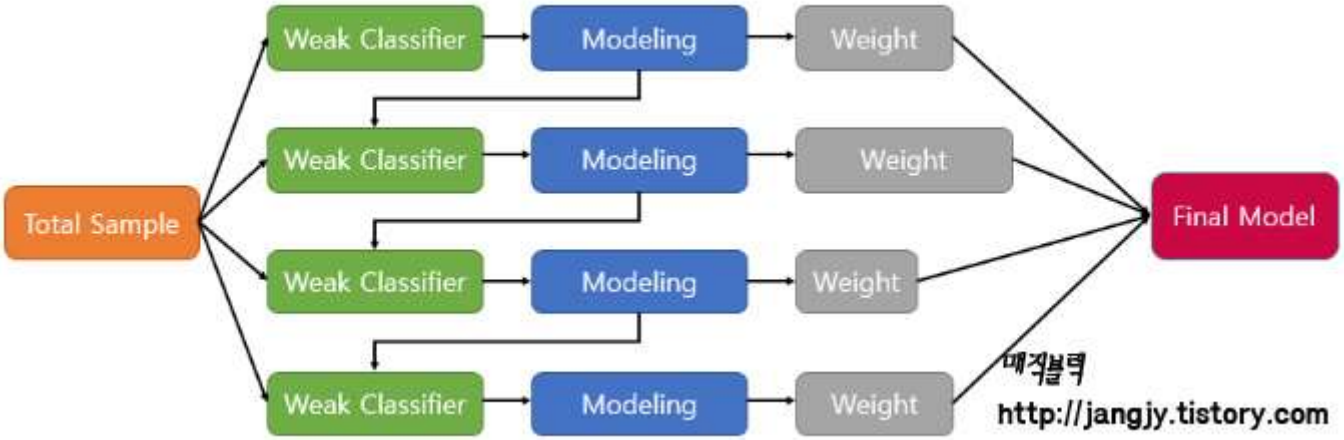
bootstrap agg**r**egating 의 줄임말로 주어진 데이터에 대해서 여러 개의 부트스트랩 자료를 생성하고 각 부트스트랩 자료를 모델링 한 후 **결합**하여 최종의 예측 모형을 산출하는 방법



배깅 & 부스팅

- 부스팅

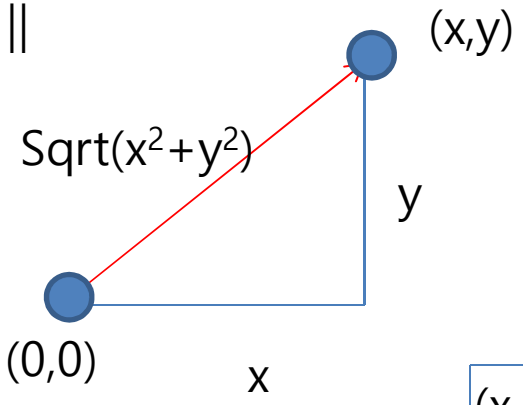
일반적으로 부스팅은 약검출기(weak classifier) 들을 여러개 모아 강검출기(strong classifier)를 생성하는 방법을 말한다.



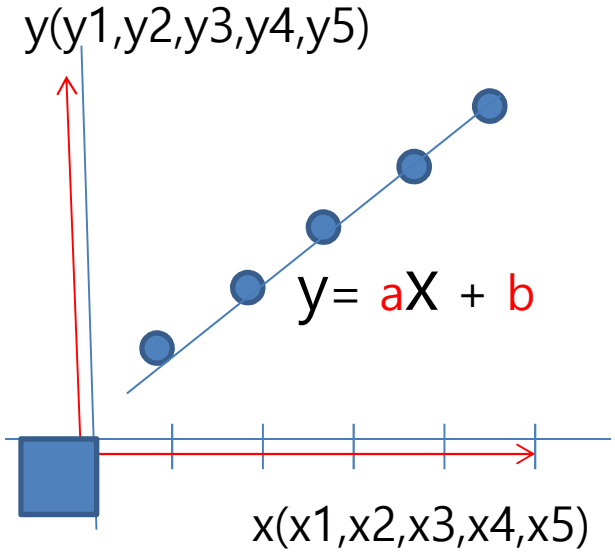
SVM

•벡터의 길이

|| x ||



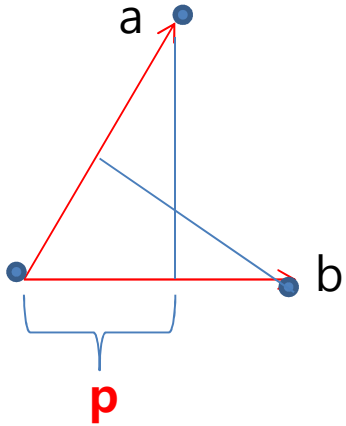
(x-0)2+(y-0)

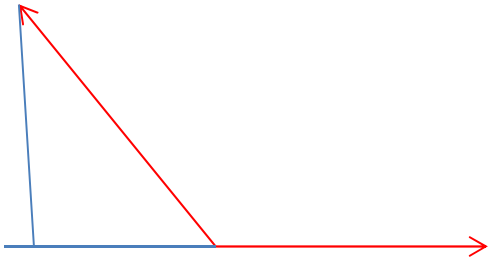


•벡터의 내적

X ● Y =(x1*y1) + (x2*y2) + ... =|x|*|y|*cos0

A와 B의 내적 = p * |b|





A와 B의 내적 = (-1) *p * |b|

•3차원 벡터의 내적

a의 성분을 (a_x, a_y, a_z)라 하고 b의 성분을 (b_x, b_y, b_z)라 하면 내적은 a·b= $a_x b_x + a_y b_y + a_z b_z$ 이다.

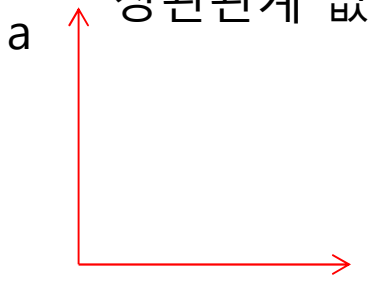
$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + a_3 b_3$$

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos\theta$$

$$\|\mathbf{a}\| = \sqrt{a_1^{\,2} + a_2^{\,2} + a_3^{\,2}}$$

•벡터의 내적

상관관계 없다



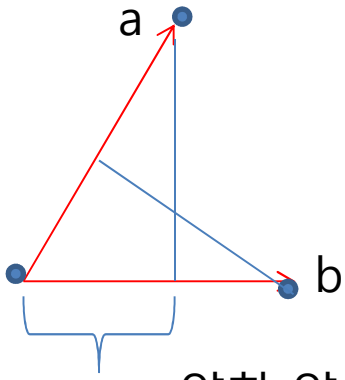
A와 B의 내적 = (-1) *p=0 * |b| =0

양의상관관계

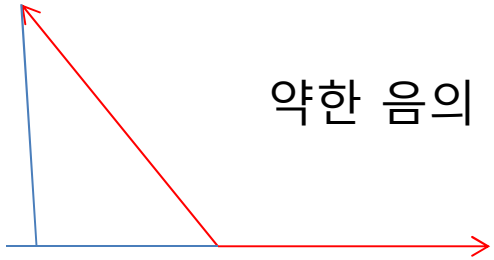
A와 B의 내적 = 1

음의상관관계

A와 B의 내적 = -1



약한 양의 상관관계



약한 음의 상관관계

상관관계 r (-1=<r<=1)

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos\theta$$

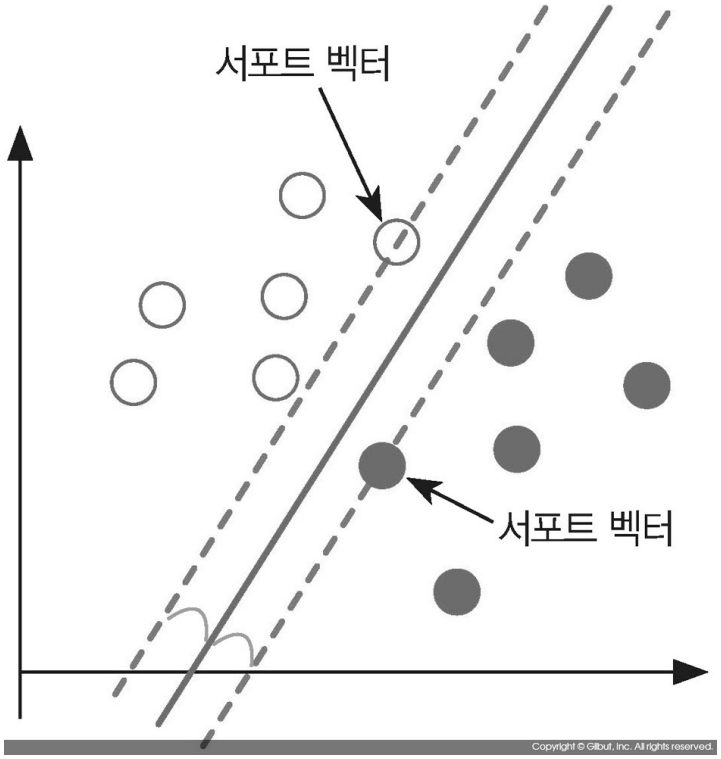
상관관계 r (-1=<r<=1) $= \frac{E[(X-m_1)(Y-n)]}{\sigma_x \sigma_y}$ = 내적 / 편차곱

= 코사인theta

SVM(Support Vector Machine) https://thebook.io/006723/

- **서포트 벡터 머신**

-서포트 벡터 머신(SVM)은 서로 다른 분류에 속한 데이터 간에 간격이 최대가 되는 선(또는 평면)을 찾아 이를 기준으로 데이터를 분류하는 모델이다.



SVM은 각 분류에 속하는 데이터로부터 같은 간격으로, 그리고 최대로 멀리 떨어진 선 또는 평면을 찾는다.

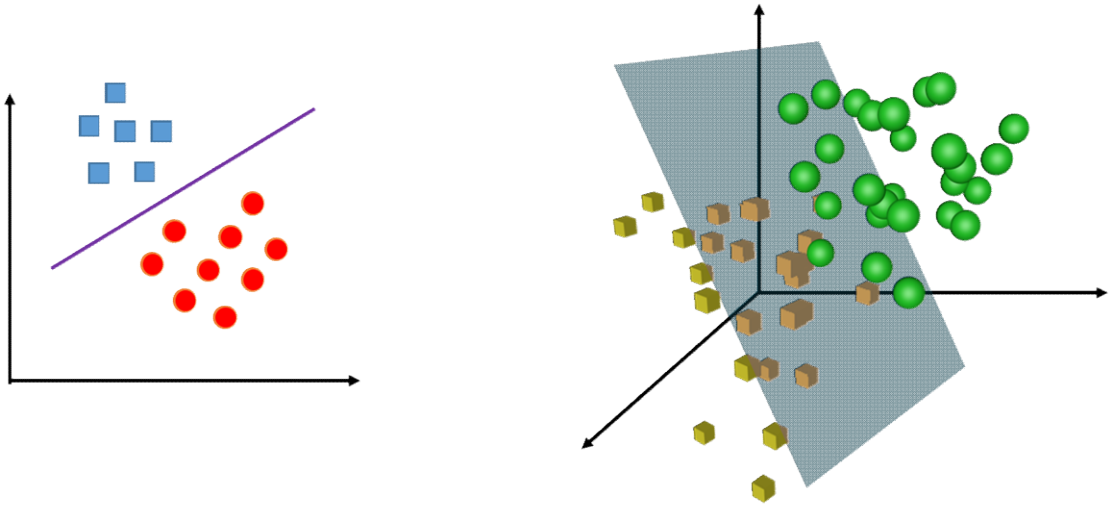
이러한 선 또는 평면을 최대 여백 초평면(Maximum Margin Hyperplane)이라고 하고, 이 평면이 분류를 나누는 기준이 된다.

SVM(Support Vector Machine)

- **서포트 벡터 머신**

-유사한 그룹끼리 분류하는 칸막이를 초평면이라고 하는데, 2차원 공간에 선(Line)으로 분리되거나 3차원 공간에서 평면으로 분리되는 것을 선형 분리 가능(Linearly Separable)이라고 한다.

-개념의 이해를 돕기 위해 선형 분리 가능한 것을 예로 들겠지만, SVM은 선형 분리 가능이 아닌 경우에도 적용이 가능하다.



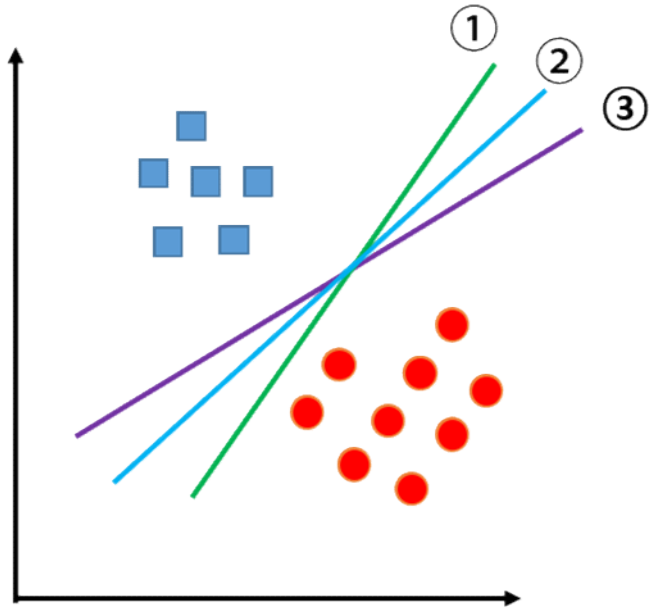
2차원 3차원

http://cinema4dr12.tistory.com/1015?category=593901

SVM(Support Vector Machine)

- **서포트 벡터 머신**

-2차원에서는 초평면은 선이 되며, 3차원에서는 평면이 된다.



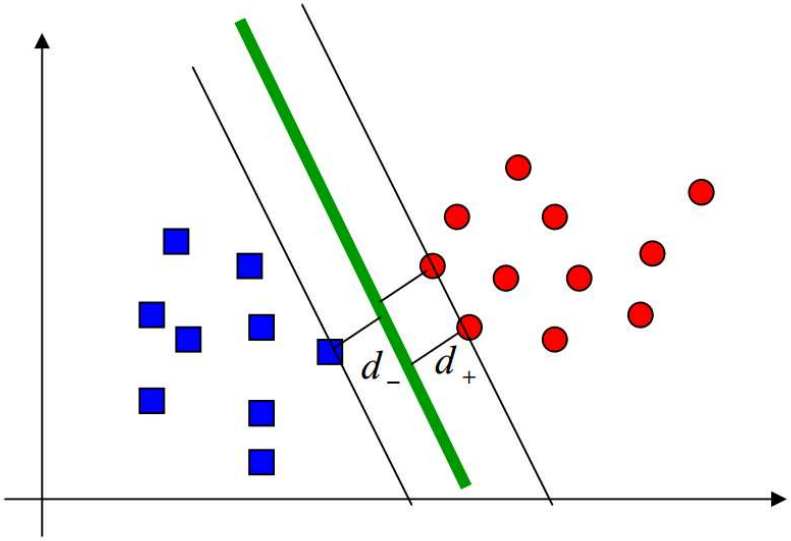
초평면 후보

최대 마진 초평면(Maximum Margin Hyperplane; MMH)을 찾는것이 SVM의 핵심

SVM(Support Vector Machine)

- **서포트 벡터 머신**

-Support Vectors는 MMH에 가장 근접한 각 그룹의 개체들이다. 각 그룹은 최소 1개의 Support Vector(s)를 갖는다. Support Vectors만으로 MMH를 정의할 수 있으며 이것이 SVM의 핵심이 되는 특징이다. 즉, Support Vectors는 각 개체가 아무리 많은 특징들을 갖고 있더라도 분류 모델을 저장하는 매우 간편한 방법이다.



최대마진 초평면의 개념

SVM(Support Vector Machine)

- **초평면에 대한 수학적 설명**

-초평면을 수학적으로 표현하면,

$$w^T x + w_0 = 0$$ (WT는 2,3,..n차원 중의 해당하는 하나의 벡터)

을 만족하는 **x**의 집합이며, *n*-차원 공간에서

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

특히 w_0를 편향(Bias)라고 하며 이는 마치 2차원 평면 내 직선의 방정식에서 *y*-절편과도 같다.

초평면은 두 그룹을 나누는 기준이 되며, 초평면에 의해 나뉘어지는 두 그룹을 다음과 같이 정의할 수 있다

분류가 1에 속하는 경우

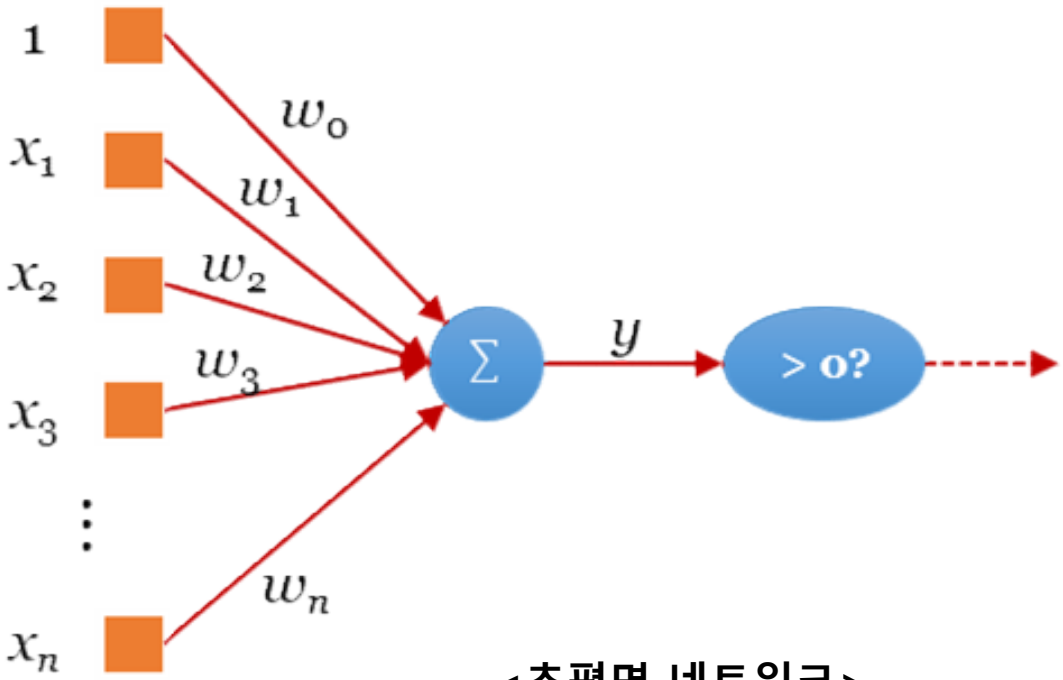
[CLASS (+1)] $w^T x + w_0 > 0$

[CLASS (-1)] $w^T x + w_0 < 0$

SVM(Support Vector Machine)

- **초평면에 대한 수학적 설명**

선형 초평면일 경우 아래 그림과 같이 Data Flow를 도식화 할 수 있다.



<초평면 네트워크>

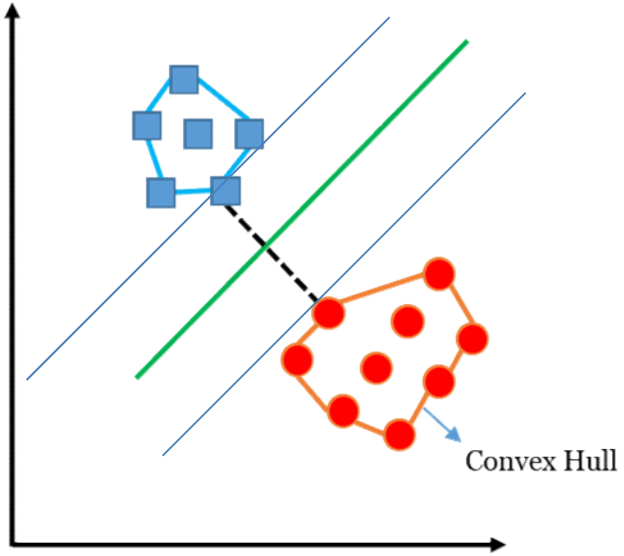
활성 함수를 Sigmoid 함수를 사용할 수 있으며 이 함수의 미분 연산을 통하여 Gradient 방법을 사용하여 가중치를 학습시킬 수 있다.

SVM(Support Vector Machine)

• **선형분리가능 데이터**

-그룹을 선형 분리 가능하다고 가정할 경우, 최대 마진을 찾는 것은 수월하다. 이 경우, MMH는 두 그룹의 경계(Boundary)가 가장 멀리 떨어지도록 하며, 이 경계를 Convex Hull이라고 부른다.

-이 때, MMH는 두 Convex Hull 사이의 가장 짧은 선에 대한 수직 이등분선이며, 최대 마진을 찾는 알고리즘은 2차 형식에 대한 최적화 기법(Quadratic Optimisation)을 사용한다.



기하학적인 마진 ρ는

$$\rho_{\mathbf{w},\, w0}(\mathbf{x},\, y) = y(\mathbf{w}^T\mathbf{x}_i + w_0)/\|\mathbf{w}\|$$

점 **x**의 초평면으로부터의 거리를 의미하며, 여기서 **w**는 초평면의 법선 벡터이다.

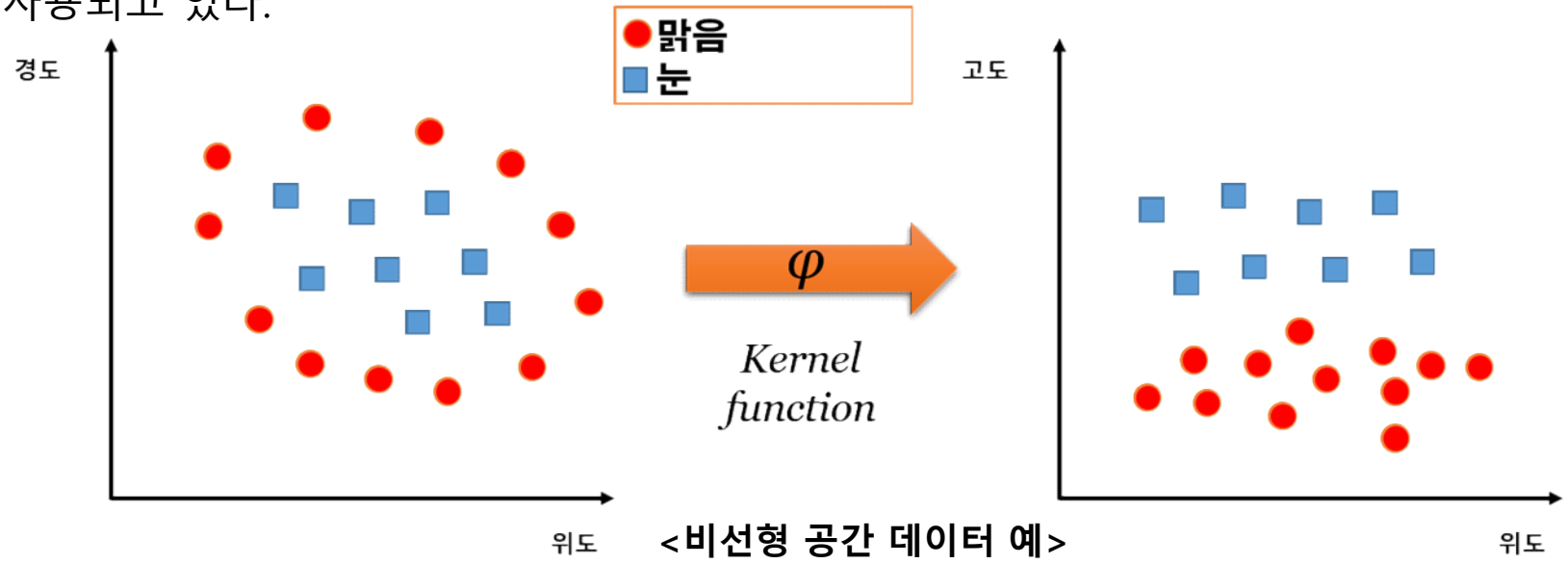
$$d_+ + d_- = 2/\|\mathbf{w}\|$$

$d_+ + d_- = 2/\|\mathbf{w}\|$ 을 최대화하면 되고, 이를 최대화한다는 것은 $\|\mathbf{w}\|$을 최소화하는 것과 동일하다.

SVM(Support Vector Machine)

• **선형분리 불가능 데이터**

-아래 그림에서 왼쪽 그림은 위도 및 경도에 따른 날씨 분포인데, 위도와 경도에 따른 날씨와의 뚜렷한 상관관계를 발견하기 어렵다. 그런데 오른쪽 그림과 같이 고도에 따른 날씨를 측정결과를 확인해보니 **고도-위도** 상에서는 데이터가 선형 분리가능한 상태임을 알 수 있다. 왼쪽 상태에서 오른쪽 상태로 변환하는(새로운 데이터 공간으로 맵핑(Mapping)) 함수를 커널 함수라고 하며, 데이터 공간 변환을 위해 다양한 커널 함수가 사용되고 있다.



<비선형 공간 데이터 예>

SVM(Support Vector Machine)

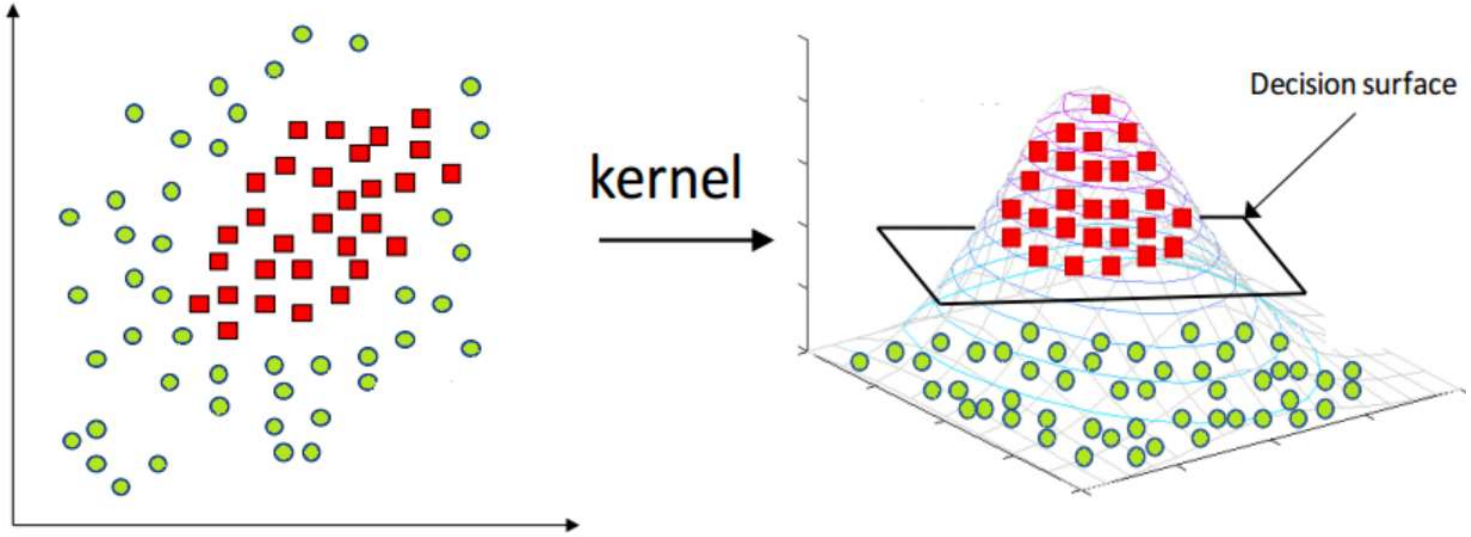
• 선형분리 불가능 데이터

-왼쪽 그림은 위에서 아래로 내려다보는 시각으로 표현된 것이며, **오른쪽 그림은 지면에서 관찰된 데이터인데, 이처럼 데이터를 보는 시각에 따라 분리 가능한 상태로 변환할 수 있다.** 이런 식으로 SVM은 비선형 커널을 이용하여 데이터를 분리할 수 있는 상태로 변환하기 위해 데이터에 새로운 차원을 추가한다. 즉, 커널 트릭은 관찰된 데이터의 특징 사이에 수학적 관계를 표현하는 새로운 특징을 구성하는 과정이라고 생각할 수 있다. 예를 들어, 고도라는 특징은 위도와 경도의 상호작용에 의해 수학적으로 표현된 것이라고 할 수 있는데, 위도-경도로 표현되는 것을 등고선에 비유할 수 있고 중심으로 갈수록 고도가 높은 것으로 해석할 수 있다는 것이다. 이렇게 하면 SVM이 원래의 데이터에서는 명시적으로 표현되지 않은 특징들을 학습할 수 있도록 한다.

SVM(Support Vector Machine)

- **커널 트릭(Kernel Trick)**

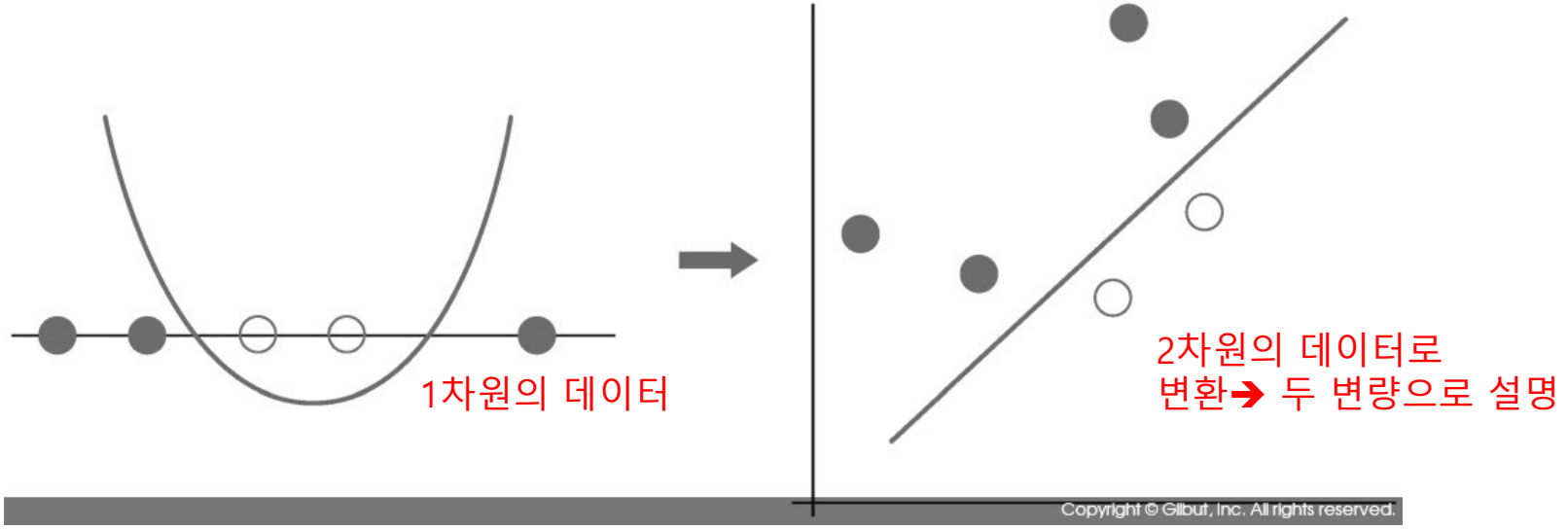
- 이 문제는 커널 트릭Kernel Trick이라는 기법으로 해결한다. 커널 트릭의 기본 아이디어는 **주어진 데이터를 적절한 고차원으로 옮긴 뒤 변환된 차원에서 서포트 벡터 머신을 사용해 초평면을 찾는 것**이다.



SVM(Support Vector Machine)

- **커널 트릭(Kernel Trick)**

- 이 문제는 커널 트릭Kernel Trick이라는 기법으로 해결한다. 커널 트릭의 기본 아이디어는 **주어진 데이터를 적절한 고차원으로 옮긴 뒤 변환된 차원에서 서포트 벡터 머신을 사용해 초평면을 찾는 것**이다.



SVM(Support Vector Machine)

- **커널 함수**

- **K(x, y) = M(x) * M(y)**를 만족하는 매핑함수가 존재할 때 K를 커널함수라고 부른다.

-에서 M(x),M(y)는 각각 H공간의 벡터이다.

예를 들어, a=(0,0), b=(1,1) 이라는 2차원 점들이 있다고 하자. 이를 3차원 공간으로 매핑시키기 위해서, **M(x) = (x1^2, 루트2 * x1*x2, x2^2)** 라는 매핑함수를 정의했다고 하자.

그러면 a=(0,0) --> a=(0,0,0)
b=(0,0,0) --> b=(1, 루트2, 1) 라는 3차원 공간 좌표로 매핑 된다.

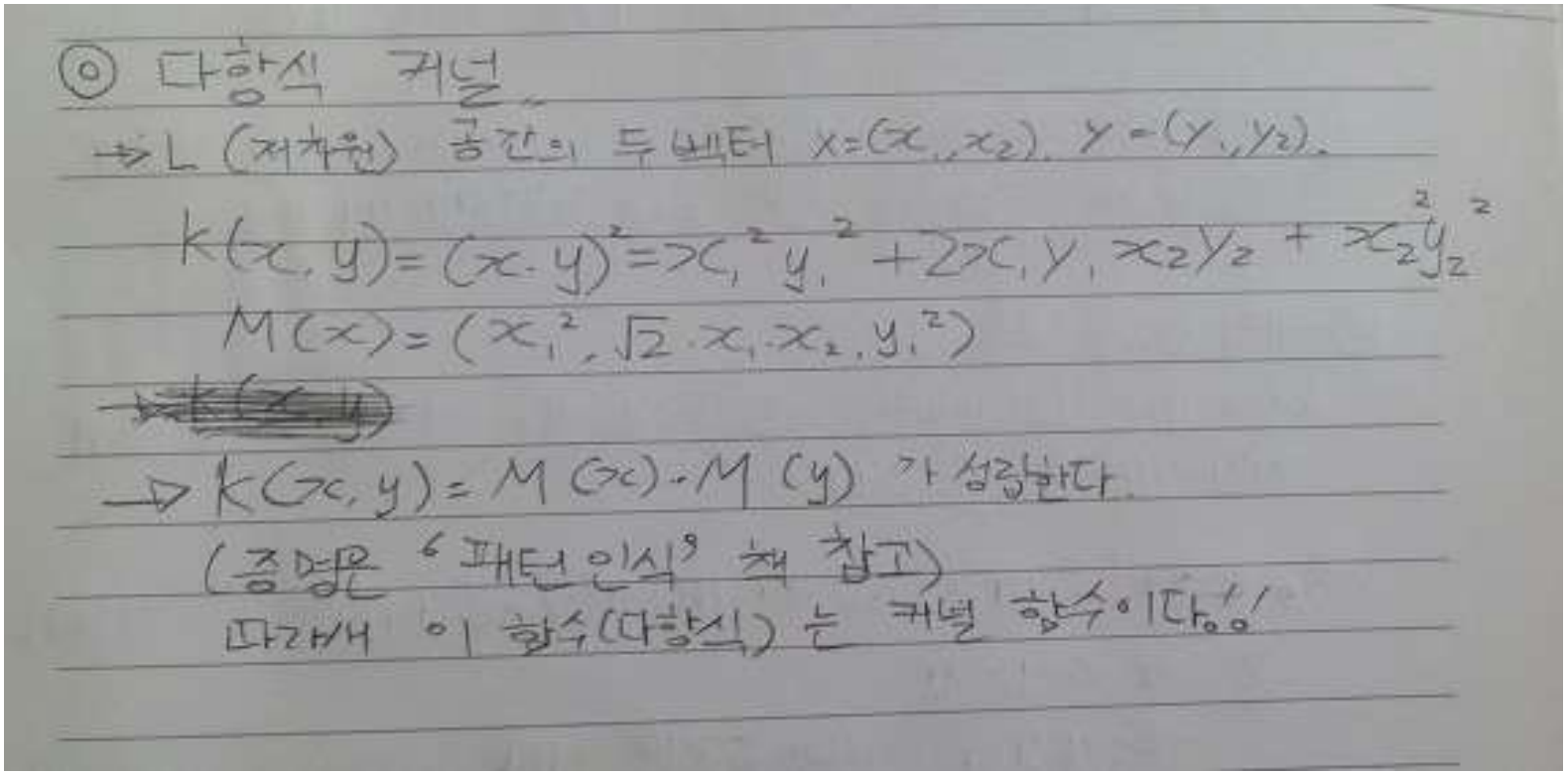
여기서 M(x): L -> H (저차원에서 고차원으로 매핑시켜 주는 매핑 함수) 가 된다.

SVM(Support Vector Machine)

- **커널 함수**

- **K(x, y) = M(x) * M(y)**를 만족하는 매핑함수가 존재할 때 K를 커널함수라고 부른다.

-에서 M(x),M(y)는 각각 H공간의 벡터이다.



SVM(Support Vector Machine)

• **커널 함수**

서포트 벡터 머신 모델의 핵심 수식은 **벡터 간 내적 계산이다**.

그러므로 커널은 다음과 같은 특징을 가진다.

① x와 y가 동일한 벡터일 때 가장 크고

② 두 벡터간의 거리가 멀어질 수록 작아진다.

③ 즉, 두 표본 데이터 간의 유사도(similarity)를 측정하는 기준으로 볼 수도 있다.

$$\frac{\text{내적}}{\text{편차곱}} = \cos \text{세타}$$

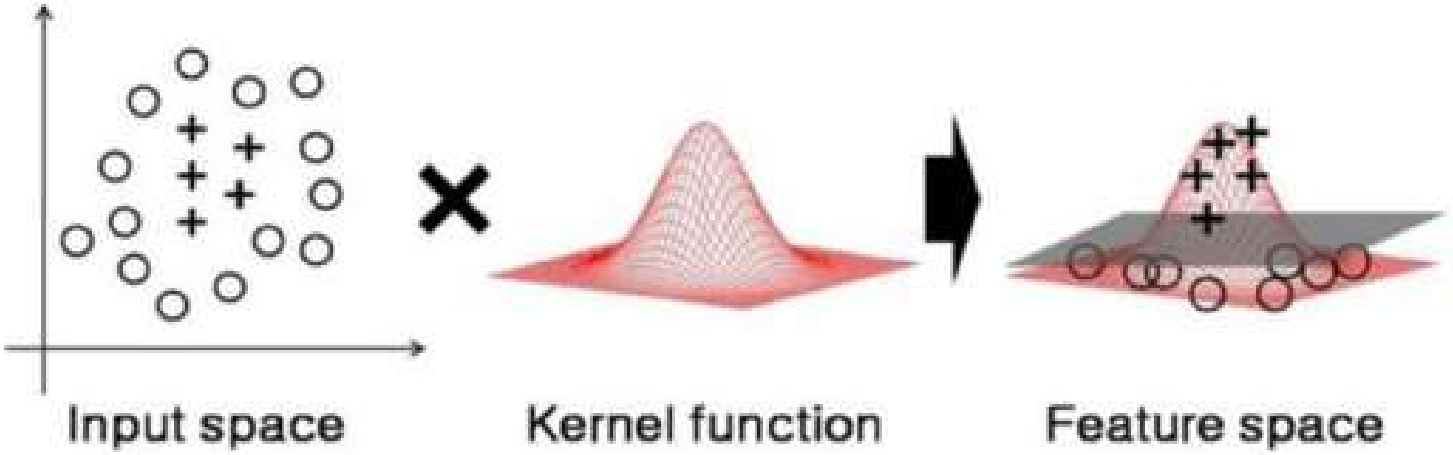
두 표본간의 유사도를 측정하는 기준으로 커널을 볼 수 있으며, X를 기저함수 변환 후 변환된 독립 변수 벡터를 내적(inner product) 한 값을 하나의 함수로 나타낸 함수가 커널 함수이다.

SVM(Support Vector Machine)

• **커널 함수**

커널 트릭에서는 실제로 데이터를 고차원으로 변환하는 대신 고차원에서 벡터 간 내적 계산을 했을 때와 같은 값을 반환하는 함수들을 사용한다.

이 함수들을 사용하면 마치 데이터를 고차원으로 옮긴 듯한 효과를 일으키면서도 데이터를 고차원으로 옮기는 데 따른 계산 비용 증가는 피할 수 있다. 이러한 함수들을 커널 함수(kernel Function)라고 부른다.



SVM(Support Vector Machine)

• **커널 함수**

-커널 함수의 대표적인 예에는 다항 커널(Polynomial Kernel)과 가우시안 커널(Gaussian Kernel(레이디얼 베이스 함수 커널Radial Basis Function Kernel)) 이 있다. D차원 다항식에 대한 다항 커널은 다음과 같다.

1) D차원 다항식에 대한 다항 커널은 다음과 같다.

$$K(x, y) = (x^{\mathsf{T}} y + c)^d$$

다항 커널은 입력의 모든 차원의 조합인 공간에서 내적을 계산한 것과 같은 결과를 반환한다.

SVM(Support Vector Machine)

- **커널 함수**

$$K(x, y) = (x^{\mathsf{T}} y + c)^d$$

2차원 입력 벡터 $x=(x_1,x_2)^T$, $y=(y_1,y_2)^T$에 d=2인 다항 커널을 적용하면

$$K\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}\right) = ((x_1\ x_2)\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} + c)^2$$

$$= (x_1 y_1 + x_2 y_2 + c)^2$$

$$= x_1^2 y_1^2 + x_2^2 y_2^2 + c^2 + 2(x_1 y_1 x_2 y_2 + x_2 y_2 c + c x_1 y_1)$$

$$= (x_1 x_1 \ \ x_1 x_2 \ \ x_2 x_1 \ \ x_2 x_2 \ \ \sqrt{2c}x_1 \ \ \sqrt{2c}x_2 \ \ c)^T (y_1 y_1 \ \ y_1 y_2 \ \ y_2 y_1 \ \ y_2 y_2 \ \ \sqrt{2c}y_1 \ \ \sqrt{2c}y_2 \ \ c)$$

SVM(Support Vector Machine)

- **커널 함수**

2) 가우시안 커널

무한 자원으로 데이터를 옮긴 뒤 그 곳에서 내적을 계산한 것과 같은 결과를 반환한다.

- **가우시안 커널식**

gamma

$$K(x, y) = \exp(-\gamma \|x - y\|^2)$$

SVM(Support Vector Machine)

- **커널 함수**

선형 커널 : 데이터를 전혀 변환하지 않음

$$K\left(\vec{x}_i, \vec{x}_j\right) = \vec{x}_i \cdot \vec{x}_j$$

시그모이드 커널 : 활성함수를 사용한 신경망과 유사

$$K\left(\vec{x}_i, \vec{x}_j\right) = \tanh\left(k\,\vec{x}_i \cdot \vec{x}_j - \delta\right)$$

가우시안 RBF 커널 : 다양한 데이터의 형태에 잘 적용되기 때문에 많은 학습 태스크에 이용

$$K\left(\vec{x}_i, \vec{x}_j\right) = e^{\frac{-\left\|\vec{x}_i - \vec{x}_j\right\|^2}{2\sigma^2}} \leftarrow \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

〈정규분포 확률밀도함수〉

SVM(Support Vector Machine)

- **SVM의 장점과 단점**

장점	단점
• 범주나 수치 예측 문제에 대해 사용할 수 있다. • 노이즈 데이터에 영향을 크게 받지 않고 잘 과적합화되지 않는다. • 특히 잘 지원되는 일부 SVM 알고리즘 때문에 신경망보다 사용하기 쉽다. • 높은 정확도와 높은 프로필로 데이터 마이닝 경쟁에서 우승해 인기를 끌었다.	• 최적의 모델을 찾기 위해 커널과 모델에서 매개변수의 여러 가지 조합 테스트가 필요하다. • 특히 입력 데이터셋이 예제 개수와 속성의 수가 많다면 훈련이 느릴 수 있다. • 해석하기가 불가능하지 않지만, 어렵고 복잡한 블랙박스가 된다.

SVM(Support Vector Machine)

- **SVM Package in R**

SVM 모델을 위한 패키지에는 e1071, kernlab 등이 있다. e1071은 효율적인 SVM 구현체로 잘 알려진 libsvm을 R에서 사용할 수 있도록 한 패키지며, kernlab은 커널 기반의 기계 학습 알고리즘을 R에서 구현한 것으로 사용자가 C++ 코드의 수정 없이 기능을 손쉽게 확장할 수 있다.

```
#install.packages("kernlab") #가우시안 커널을 기본으로 사용한다.
library(kernlab)

m=ksvm(Species~.,data=iris)
m

head(predict(m,newdata=iris))
```

SVM(Support Vector Machine)

- **SVM Package in R**

#커널은 특별한 변환 없이 내적을 계산하라는 뜻이다.

ksvm(Species~.,data=iris,kernel="vanilladot")

#ploydot은 다항커널을 사용하는 것이고, degree=3은 3차원을 뜻하는 것이다.

m=ksvm(Species~.,data=iris,kernel="polydot",kpar=list(degree=3))

SVM(Support Vector Machine)

• **SVM 파라미터 튜닝**

파라미터의 값을 정하는 방법에는

① 교차 검증을 활용한 방법

② SVM 패키지가 제공하는 파라미터 튜닝 기능을 활용하는 방법

#E1071에서는 tune() 함수를 사용해 모델을 튜닝 할 수 있다.

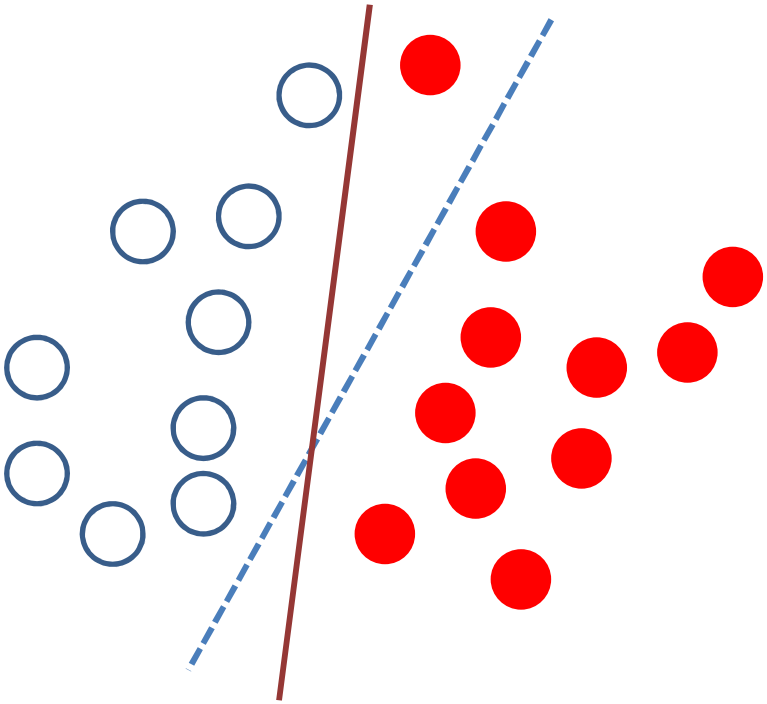
> help(tune)

SVM(Support Vector Machine)

• **SVM 파라미터 튜닝**

-가우시안 커널의 gamma(r)와 cost 파라미터

-SVM에서 cost(흔히 C로 지칭)는 과적합을 막는 정도를 지정하는 파라미터이다. 두 개의 분류를 정확히 가운데로 나누는 선이 있으면 좋겠지만 때에 따라 한 두 개 데이터만 특이 값을 갖고 있는 경우가 있을 수 있다.



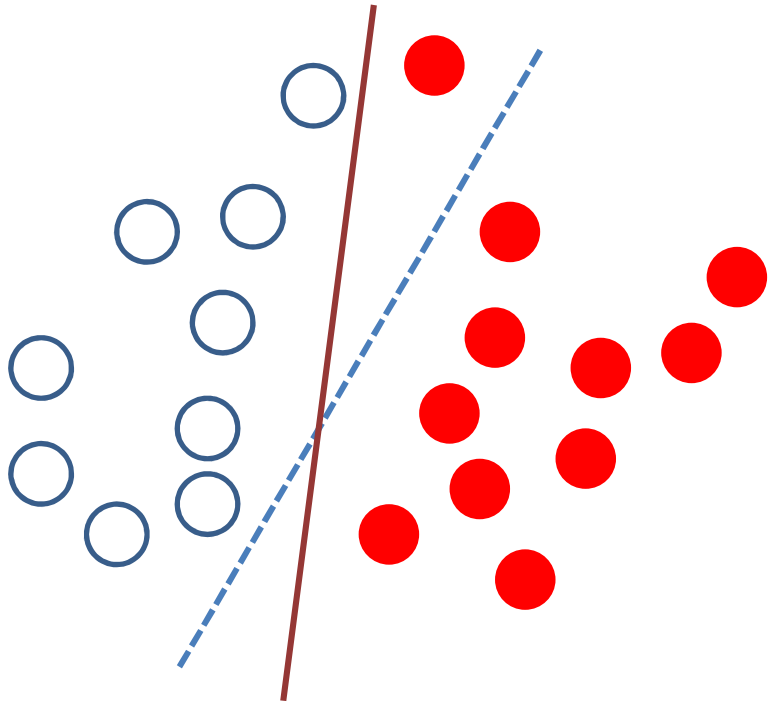
직선은 혼자만 특이한 위치에 있는 검은 점까지 고려한 선이며 점선은 해당 점을 제외한 뒤의 선이다. 직선은 데이터를 정확히 반영하지만 데이터의 노이즈에 집착한 과적합 가능성이 보인다. 반면 점선은 한 개의 점은 잘못 분류했지만 전체적인 데이터의 모양을 더 잘 나타낸다.

cost는 점선의 경우처럼 데이터를 잘못 분류하는 선을 긋게 될 경우 얼마만큼의 비용(cost) 를 지불할 것인지를 지정한다.

SVM(Support Vector Machine)

• **SVM 파라미터 튜닝**

-가우시안 커널의 gamma(r)와 cost 파라미터



cost는 점선의 경우처럼 데이터를 잘못 분류하는 선을 긋게 될 경우 얼마만큼의 비용(cost) 를 지불할 것인지를 지정한다.

SVM은
1) 데이터를 한 가운데로 얼마나 잘 나누는지와
2) 잘못 구분한 점으로 인한 비용의 합을 최소화하는 선을 찾는다.

결과적으로 SVM은 Cost를 사용해 과적합 정도를 조절하게 된다.

SVM(Support Vector Machine)

- **SVM 파라미터 튜닝**

- 가우시안 커널에서 gamma와 cost 파라미터를 찾는 예

install.packages("e1071") #tune() 함수를 담고 있음

library(e1071)

#그리드탐색을 사용한 파라미터 튜닝을 실행한다.
#그리드 탐색: 인자로 주어진 모든 가능한 경우에 대해 테스트해보는 방식
tune.svm(Species~.,data=iris,gamma=2^(-1:1),cost=2^(2:4))

```
> tune.svm(Species~.,data=iris,gamma=2^(-1:1),cost=2^(2:4))

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
   0.5    4

- best performance: 0.04
```

tune()의 반환 값은 객체며, 객체의 속성은 attributes()로 살펴볼 수 있다.

SVM(Support Vector Machine)

- **SVM 파라미터 튜닝**

- 가우시안 커널에서 gamma와 cost 파라미터를 찾는 예

➢**attributes(result)**

➢**result$best.parameters #최적 파라미터**

➢**result$best.parameters[“gamma”] #최적 파라미터 중 gamma**

➢**result$best.parameters[“cost”] #최적 파라미터 중 cost**

KNN : K-최근접 이웃(K-Nearest Neighbor) 기법

- **K-최근접 이웃(K-Nearest Neighbor) 기법**

-특정 데이터 좌표점과 다른 나머지 데이터 좌표점 들간의 거리에 기반을 두어 가장 가까운 K개점들의 목적변수(혹은 반응변수) 값들의 다수결로 분류하는 기법

-게으른 학습(Lazy Learning)

KNN : K-최근접 이웃(K-Nearest Neighbor) 기법

• **K-최근접 이웃(K-Nearest Neighbor) 기법**

K-최근접 이웃 기법은 목표변수의 범주를 알지 못하는 데이터 세트의 분류를 위해 해당데이터 세트와 가장 유사한 주변 데이터 세트의 범주로 지정하는 방식으로 분류예측을 하는 기법이다. 이러한 K-최근접 이웃 기법에서는 해당 데이터 점과 주변 데이터 세트 간의 '유사성'을 어떻게 측정할 것인가와 최종적으로 목표변수의 범주를 분류할 때 주변 데이터 세트 몇 개를 기준으로 판단할 것인가에 대한 기준이 필요하다.

① 유사성 측정 방법

일반적으로 두 점간의 **유클리디안 제곱 거리의 역수를** 취하거나 피어슨 상관계수를 이용하여 유사성을 계산한다. 점들이 이산형 변수일 경우 **자카드 계수**(Jaccard coefficient)를 사용한다.

KNN : K-최근접 이웃(K-Nearest Neighbor) 기법

• K-최근접 이웃(K-Nearest Neighbor) 기법

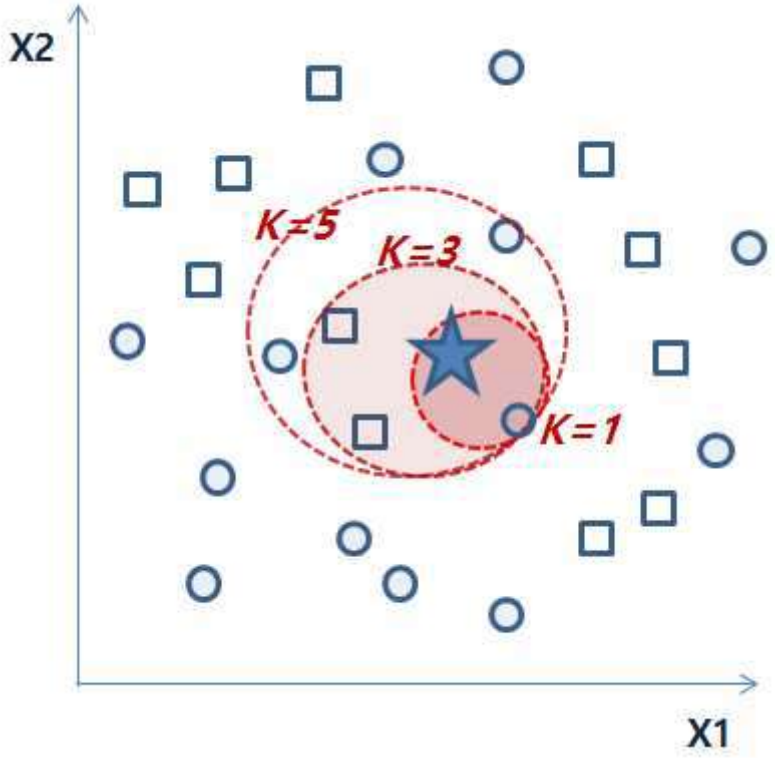
② 목표변수 분류 기준

K-최근접 이웃에서의 'K'는 해당 데이터 점과 주변 데이터 세트 간의 유사성을 측정한 후, 해당 데이터 점의 목표변수 분류를 위해 참조할 주변 데이터 점들의 개수를 의미한다. 예를 들어 어떤 고객이 좋아하는 영화와 유사하면서 기존에 시청하지 않았던 영화에 대한 추천을 생각해볼 수 있다. 이럴 때 해당 고객이 좋아하는 영화와 가장 유사한 영화 1개만 고려한다면 '1-근접이웃'이라고 말할 수 있고, 유사한 영화 3개 를 고려한다면 '3-근접이웃'이라고 말할 수 있다. 이런 식으로 해당 데이터 점과 유사한 k개의 주변 데이터 점을 살펴보고, 해당 데이터 점의 분류범주를 고려하여 다수결의 원칙에 따라 새로운 범주를 결정하는 방식이 K-근접이웃 기법이다.

KNN : K-최근접 이웃(K-Nearest Neighbor) 기법

•K-최근접 이웃(K-Nearest Neighbor) 기법

② 목표변수 분류 기준



'☆'으로 표시된 점이 새롭게 분류해야 할 데이터 값이 고 나머지'□'와 '◯'로 표시된 점들은 주변에 존재하는 다른 데이터 점이다. 여기서 K=1로 설정하면, '☆'와 가장 가까운 데이터 점은'◯'이므로'☆'의 목표변수는 '◯'로 분류될 것이다. 반면 K=3으로 설정하면'☆'와 가장 가까운 3개의 점을 고려하게 되므로'□'가 2개,'◯'가 1개이므로 이때는'☆'점은'□'로 분류되게 된다. 한편 K=5로 설정한다면,'☆'주변에'◯'가 더 많으므로, 다수결의 원칙에 의하여 또다시 '◯'로 분류될 것이다. 여기서 우리가 알 수 있는 것은 K-최근접 이웃 기법에서는 K 값을 어떻게 정하느냐에 따라 목표변수의 범주 예측 결과가 크게 달라질 수 있다는 점이다. 따라서 K-최근접 이웃 기법에서는 적절한 K 값을 정하는 것이 중요하다.

K-최근접 이웃 기법에서의 K 값 변화에 따른 목표변수 범주 변화

KNN : K-최근접 이웃(K-Nearest Neighbor) 기법

• **K-최근접 이웃(K-Nearest Neighbor) 기법**

③ K-최근접 이웃 기법의 장/단점

장 점	단 점
• 알고리즘 이해가 쉽고 직관적이다. • 사전 모형 설정 및 모수 추정이 필요 없다. • 데이터 세트의 확률분포 등에 대한 가정이 필요 없다. • 빠른 훈련(학습) 시간	• 모형이 없으므로, 변수들간의 관계 등 가설검증이나 구조 등에 대한 분석을 통해 통찰력을 얻기 어렵다. (특정한 가설이나 모형 없이 주어진 데이터를 통해 범주의 분류결과만 판단) • K에 대한 명확한 기준 없으며 시행착오적 접근 필요 • 느린 분류(예측) 소요시간 (새로운 데이터가 주어질 때마다 모든 데이터와의 유사성 계산해야 하므로 그만큼 시간 소요. 이런 특성으로 인해 게으른 학습(Lazy Learning)으로 불림) • 많은 메모리 소요(대용량 데이터일 때 불리함)

나이브 베이즈 기법

•나이브 베이즈(Naive Bayes) 기법

나이브 베이즈 기법은 목표변수의 범주를 학습시키기 위해 통계학에서 사용되는 베이즈 정리를 사용한다. 즉, 나이브 베이즈 기법은 베이즈 확률 추정에 기반을 둔 확률모형이다.

① 베이즈 정리(Bayes'theorem)

베이즈 정리는 특정 사건이 발생할 확률을 다음과 같은 형태로 표현한 이론을 말한다.

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

즉, 사건 B가 일어났을 때 사건 A가 일어날 확률(사후확률)은 사건 A가 일어날 확률 (사전확률)과 사건 A가 일어났을 때 사건 B가 일어날 조건부 확률 P(B|A)의 곱을 사건 B가 일어날 확률 P(B)로 나누어 알아낼 수 있다는 뜻이다

나이브 베이즈 기법

•나이브 베이즈(Naive Bayes) 기법

② 나이브 베이즈 알고리즘

예를 들어 받은 이메일이 스팸일 확률을 구하는 문제를 고려해 보자.
실제 데이터를 관측하기 전에 경험이나 주관적 확신 등에 의거하여, '내가 오늘 받은 이메일 중 스팸 메일이 포함되어 있을 가능성은 대략 10%일 것 같다.'고 가정할 수 있다. 이렇게 추가적인 증거 없이 (Native)가능성을 추정하는 것을 '사전확률'이라고 할 수 있다. 이제 실제 데이터를 관측했다고 가정해보자. 오늘 도착한 이메일 중 스팸 메일 들에 '반짝할인'이라는 단어가 들어가 있는 조건부 확률 P(반짝할인|스팸) 및 주변확률 P(반짝할인)의 확률을 계산할 수 있다. 이 경우 조건부확률 P(반짝할인|스팸)를 0.4,주변확률 P(반짝할인)를 0.05라고 가정하면, 본 예시에서 구하고자 하는 사후확률 P(스팸|반짝할인)는 (0.4*0.1)/0.05 = 0.8로 계산할 수 있다. 즉, 받은 이메일에서 '반짝할인'이라는 단어가 들어가 있으면, 해당 이메일이 '스팸'일 확률은 80%라고 할 수있다. 이는 데이터 세트를 관측하기 전의 사전확률 10%보다 8배 높아진 것이므로,'반짝할인'이라는 단어가 들어가 있는 이메일은 스팸 메일함으로 분류되어야 한다는 '강한 확신'을 줄 수 있다고 볼 수 있다.

나이브 베이즈 기법

•나이브 베이즈(Naive Bayes) 기법

③ 나이브 베이즈 기법의 장/단점

장 점	단 점
• 개념과 이론이 단순하고 계산이 빠르다. • 노이즈 및 결측치가 있어도 잘 수행된다. • 고차원의 데이터 세트에 적합하다. • 알고리즘의 단순성에 비해서 복잡한 분류 알고리즘보다 예측결과가 더 효과적인 경우가 많다.	• 모든 속성이 동등하게 중요하고 독립적이라는 결함 가정에 의존한다. • 독립변수들이 범주 형태가 아닌 수치 형태일 경우 정확성이 떨어진다. • 범주 분류문제에는 적합하나, 예측된 범주의 확률값을 활용해야 할 경우에는 적합하지 않다.

④ 나이브 베이즈 기법의 활용 분야

실제로 스팸메일 필터에 많이 사용되고 있으며, 텍스트 데이터 등에 근거한 문서 분류에 많이 사용된다. 그 외에도 빠르고 효율적인 분류가 필요한 침입자 검출 및 이상행동 검출, 컴퓨터 네트워크 진단, 질병 진찰, 이탈 예측, 민원 예측 등의 분야에도 활용되고 있다.