Data Mining with R

2. 분류 (Classifications)

Contents

- 분류의 개념
- Decision Trees
- Neural Networks
- SVM (Singular Value Decomposition)

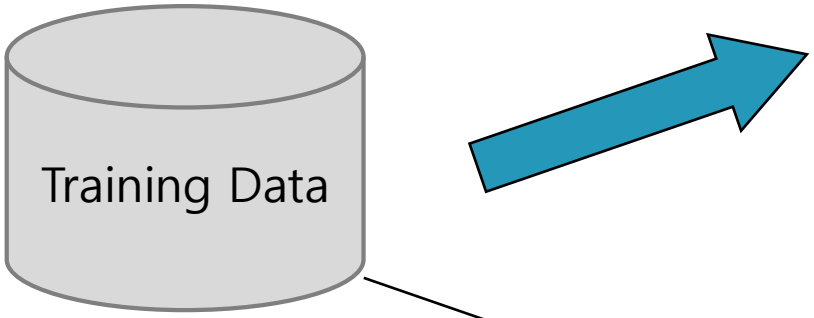
Classification vs. Prediction

- Classification
  - 범주형 데이터의 label 예측 (이산형, 명사형)
  - 학습 집합과 이에 맞는 범주 label을 이용하여 학습하고, 새로운 데이터에 적용한다.
- Prediction
  - 연속적인 값에 대한 함수/모델
- 주요 사례
  - 신용 평가
  - Target Marketing
  - 의료 진단
  - Fraud detection

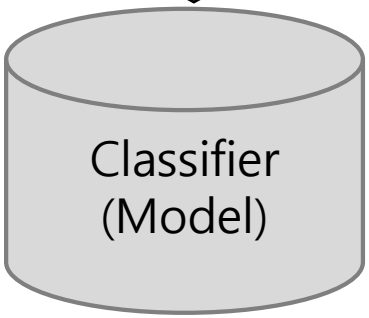
Two-Step Process

- Model construction
  - 모든 데이터는 주어진 범주에 속한다고 가정
  - 모델 생성에 사용하는 데이터는 학습 집합 (training set)
  - 사용하는 알고리즘에 따라 학습 모델은 분류 규칙, 결정 나무, 수식의 형태를 가짐
- Model usage
  - 미래 또는 미지의 데이터에 대한 분류
  - 모델의 정확도 추정
    - 테스트 집합 (test set)의 알려진 범주와 모델의 결과를 비교
    - 정확도 (accuracy rate)는 모델에 의해서 정확하게 분류된 테스트 집합의 비율
    - 테스트 집합은 학습 집합과 별개임. 과적합(over-fitting)이 발생할 수 있음)
  - 정확도가 사용할 만 하면, 새로운 데이터에 모델을 적용하여 데이터를 분류

Model construction



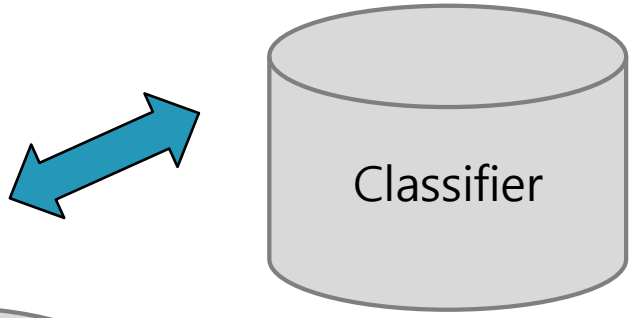
Classification Algorithms

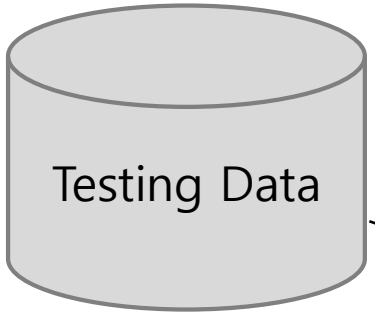


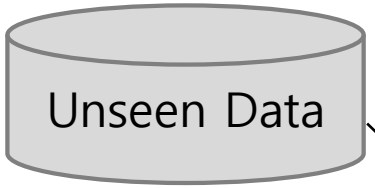
NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = 'professor'
OR years > 6
THEN tenured = 'yes'

Using the Model







(Jeff, Professor, 4)

NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Tenured?

Supervised vs. Unsupervised Learning

- 교사 학습 (Supervised learning, classification)
  - 학습 집합의 성격을 대변하는 범주가 주어짐
  - 새로운 데이터는 학습 집합에 기반하여 분류됨

- 비교사 학습 (Unsupervised learning, clustering)
  - 학습 집합의 범주는 없음
  - 데이터에 있는 클래스, 범주를 추정하는 것이 목적

Data 준비

- Data cleaning
  - Noise(이상치)를 줄이거나 missing value(결측값)을 처리하기 위한 데이터 전처리
- 관련성 분석 (feature selection)
  - 관련이 없거나 불필요한 요소 제거
- 데이터 변환
  - 데이터의 일반화 또는 정규화

Classification 방법 평가

- 정확도
  - 분류 정확도 : class의 label 예측
  - 예측 정확도 : 예측 속성의 값 추정
- 속도
  - 모델 생성 속도 (training time)
  - 모델 사용 속도 (classification / prediction time)
- Robustness
  - Noise와 결측치 처리
- Scalability
  - 처리할 수 있는 데이터 량
- Interpretability
  - 모델에 대한 이해도와 통찰력
- 기타
  - Goodness of rules (의사 결정 나무의 크기 등)

Contents

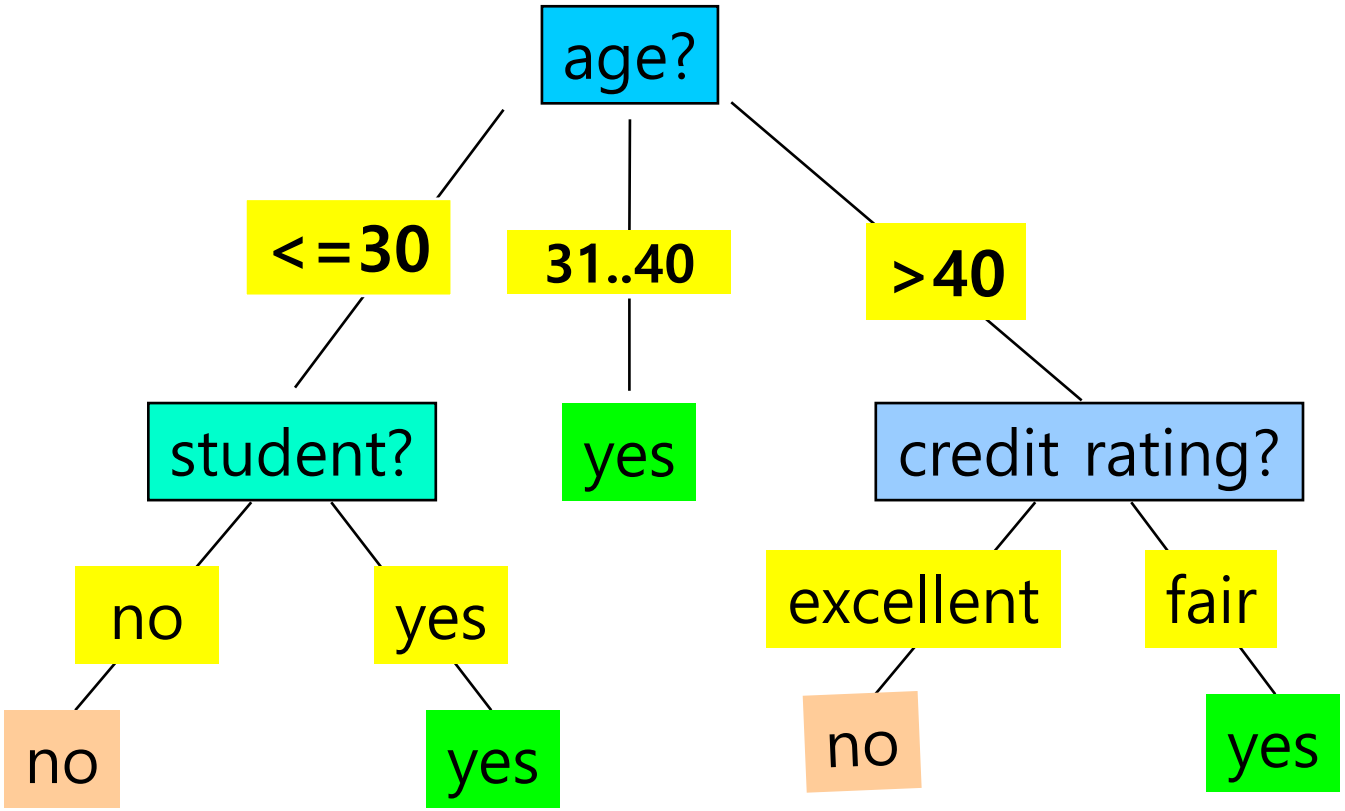
- 분류의 개념
- Decision Trees
- Neural Networks
- SVM (Singular Value Decomposition)

Decision Tree (의사 결정 나무)

- 학습 집합

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

‘buys_computer’에 대한 의사 결정 나무



특징

- 장점
  - 이해하고 해석하기 쉬움
  - 전문가에 의해서 새로운 통찰력을 얻을 수 있음
  - 새로운 시나리오를 추가 가능
  - 다양한 시나리오에 대해 최고, 최악, 기대값을 알 수 있음
- 단점
  - 분류되는 데이터의 크기가 다를 때, 분류 크기가 많은 것에 더 영향을 받음
  - 값이 많을 때는 계산이 복잡해 짐
  - 정확도가 상대적으로 떨어짐

iris Data

- 데이터 준비

```
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

- 학습 데이터와 테스트 데이터로 나눔

```
> set.seed(1234)
> ind <- sample(2, nrow(iris), replace = TRUE, prob = c(0.7, 0.3))
> train.data <- iris[ind == 1,]
> test.data <- iris[ind == 2,]
```

Build a ctree

- Control: MinSplit, MinBusket, MaxSurrogate, MaxDepth
- Target: Species
- Independent variable: others

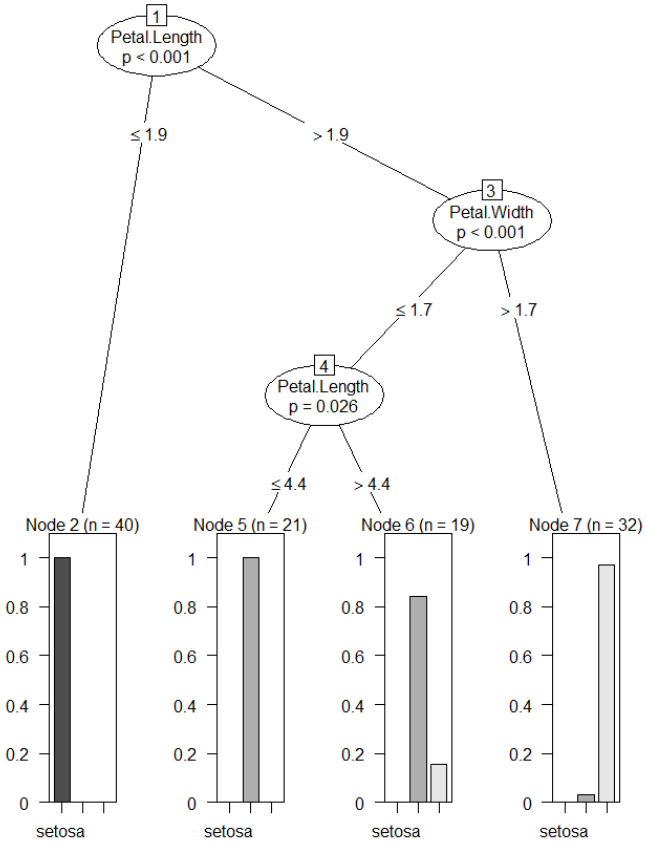
```
> library(party)
> myFormula <- Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width
> iris_ctree <- ctree(myFormula, data = train.data)
>
> table(predict(iris_ctree), train.data$Species)

            setosa versicolor virginica
  setosa        40          0         0
  versicolor     0         37         3
  virginica      0          1        31
```

Print ctree

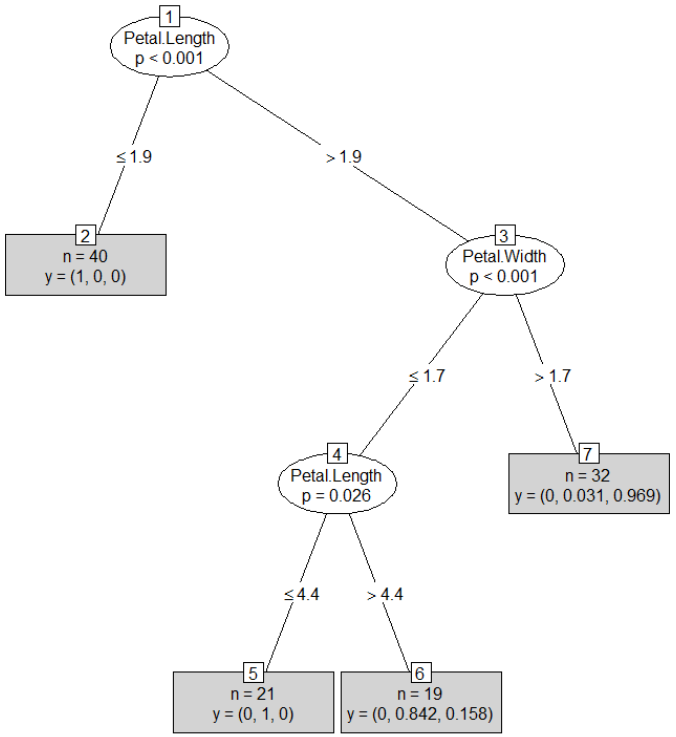
```
> print(iris_ctree)

         Conditional inference tree with 4 terminal nodes

Response:  Species
Inputs:  Sepal.Length, Sepal.Width, Petal.Length, Petal.Width
Number of observations:  112

1) Petal.Length <= 1.9; criterion = 1, statistic = 104.643
  2)*  weights = 40
1) Petal.Length > 1.9
  3) Petal.Width <= 1.7; criterion = 1, statistic = 48.939
    4) Petal.Length <= 4.4; criterion = 0.974, statistic = 7.397
      5)*  weights = 21
    4) Petal.Length > 4.4
      6)*  weights = 19
  3) Petal.Width > 1.7
    7)*  weights = 32
```

Plot ctree

```
> plot(iris_ctree)
```




```
> plot(iris_ctree, type='simple')
```



Test

```
> testPred <- predict(iris_ctree, newdata = test.data)
> table(testPred, test.data$Species)

testPred     setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         12         2
  virginica       0          0        14
```

The bodyfat Dataset

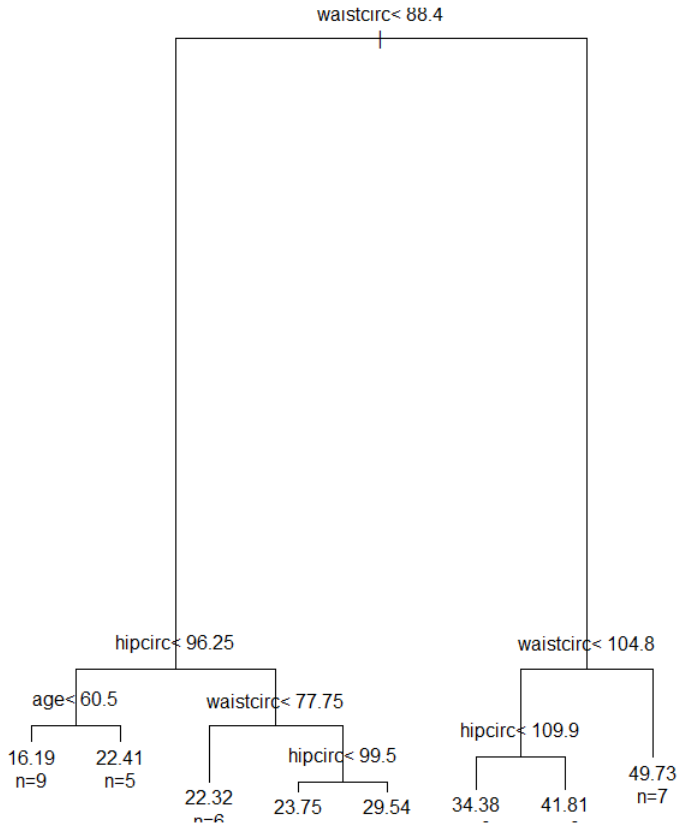
```
> data('bodyfat', package='TH.data')
> dim(bodyfat)
[1] 71 10
> head(bodyfat, 5)
   age DEXfat waistcirc hipcirc elbowbreadth kneebreadth anthro3a
47  57  41.68     100.0   112.0          7.1          9.4     4.42
48  65  43.29      99.5   116.5          6.5          8.9     4.63
49  59  35.41      96.0   108.5          6.2          8.9     4.12
50  58  22.79      72.0    96.5          6.1          9.2     4.03
51  60  36.42      89.5   100.5          7.1         10.0     4.24
   anthro3b anthro3c anthro4
47     4.95     4.50    6.13
48     5.01     4.48    6.37
49     4.74     4.60    5.82
50     4.48     3.91    5.66
51     4.68     4.15    5.91
```

Train a Decision Tree with rpart

```
> set.seed(1234)
> ind <- sample(2, nrow(bodyfat), replace=T, prob=c(0.7, 0.3))
> bodyfat.train <- bodyfat[ind==1,]
> bodyfat.test <- bodyfat[ind==2,]
>
> library(rpart)
> myFormula <- DEXfat ~ age + waistcirc + hipcirc + elbowbreadth + kneebreadth
> bodyfat_rpart <- rpart(myFormula, data = bodyfat.train, control=rpart.control(minsplit=10))
>
> print(bodyfat_rpart)
n= 56

node), split, n, deviance, yval
      * denotes terminal node

 1) root 56 7265.0290000 30.94589
   2) waistcirc< 88.4 31  960.5381000 22.55645
     4) hipcirc< 96.25 14  222.2648000 18.41143
       8) age< 60.5 9   66.8809600 16.19222 *
       9) age>=60.5 5   31.2769200 22.40600 *
     5) hipcirc>=96.25 17  299.6470000 25.97000
      10) waistcirc< 77.75 6   30.7345500 22.32500 *
      11) waistcirc>=77.75 11  145.7148000 27.95818
        22) hipcirc< 99.5 3    0.2568667 23.74667 *
        23) hipcirc>=99.5 8   72.2933500 29.53750 *
   3) waistcirc>=88.4 25 1417.1140000 41.34880
     6) waistcirc< 104.75 18  330.5792000 38.09111
      12) hipcirc< 109.9 9   68.9996200 34.37556 *
      13) hipcirc>=109.9 9   13.0832000 41.80667 *
     7) waistcirc>=104.75 7  404.3004000 49.72571 *
```

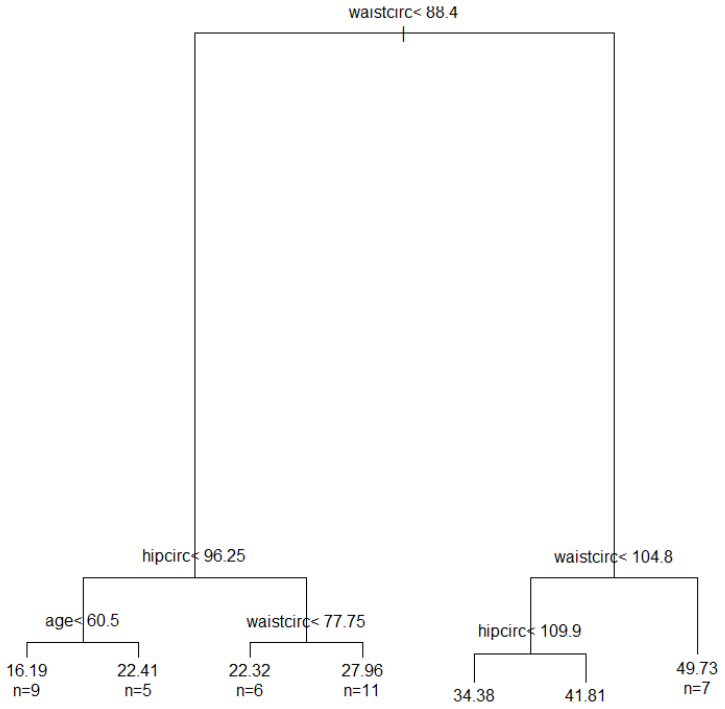
Plot rpart

```
> plot(bodyfat_rpart)
> text(bodyfat_rpart, use.n=T)
```



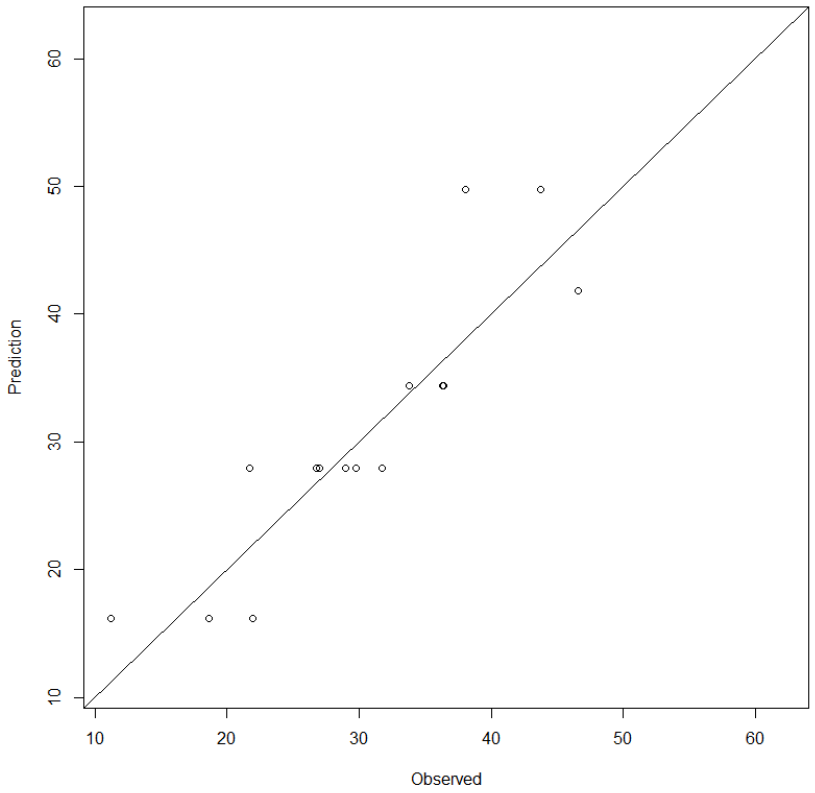
Select the Best Tree

```
> opt <- which.min(bodyfat_rpart$cptable[, 'xerror'])
> cp <- bodyfat_rpart$cptable[opt, 'CP']
>
> bodyfat_prune <- prune(bodyfat_rpart, cp=cp)
>
> plot(bodyfat_prune)
> text(bodyfat_prune, use.n=T)
```



Model Evaluation

```
> DEXfat_pred <- predict(bodyfat_prune, newdata = b
odyfat.test)
There were 12 warnings (use warnings() to see them)
> xlim <- range(bodyfat$DEXfat)
> plot(DEXfat_pred ~ DEXfat, data = bodyfat.test, x
lab = 'Observed', ylab = 'Prediction', ylim = xlim,
 xlim = xlim)
> abline(a = 0, b = 1)
```

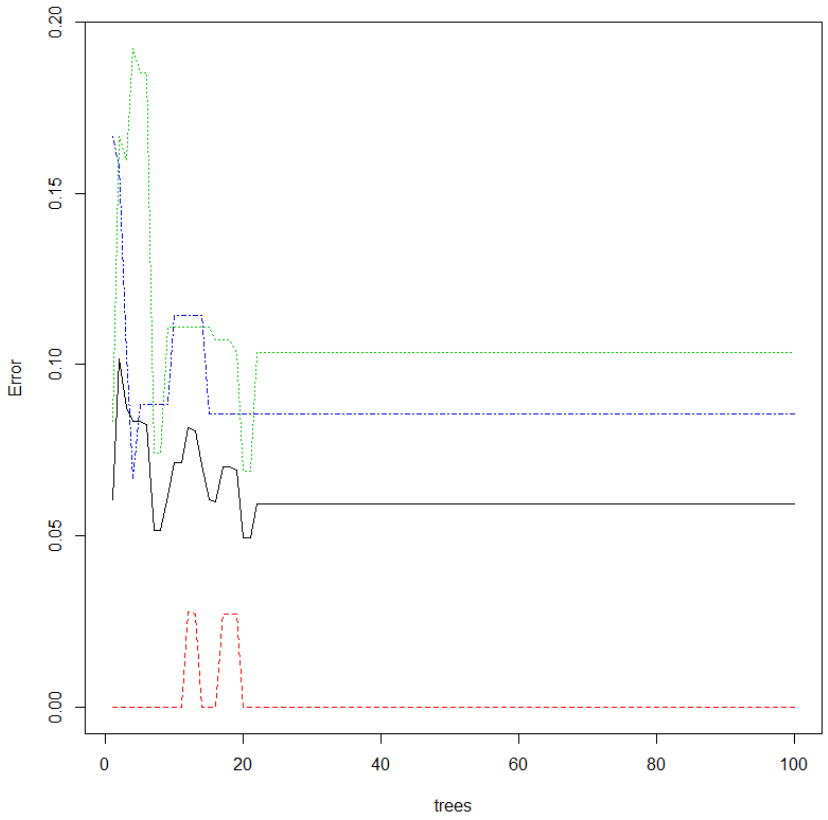


Train a Random Forest

```
> ind <- sample(2, nrow(iris), replace = TRUE, prob = c(0.7, 0.3))
> train.data <- iris[ind == 1,]
> test.data <- iris[ind == 2,]
>
> library(randomForest)
> rf <- randomForest(Species ~ ., data=train.data, ntree=100, proximity=T)


> table(predict(rf), train.data$Species)

             setosa versicolor virginica
  setosa         37          0         0
  versicolor      0         26         3
  virginica       0          3        32

> print(rf)

Call:
 randomForest(formula = Species ~ ., data = train.data, ntree = 100,
  proximity = T)
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 2

        OOB estimate of  error rate: 5.94%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         37          0         0  0.00000000
versicolor      0         26         3  0.10344828
virginica       0          3        32  0.08571429
```

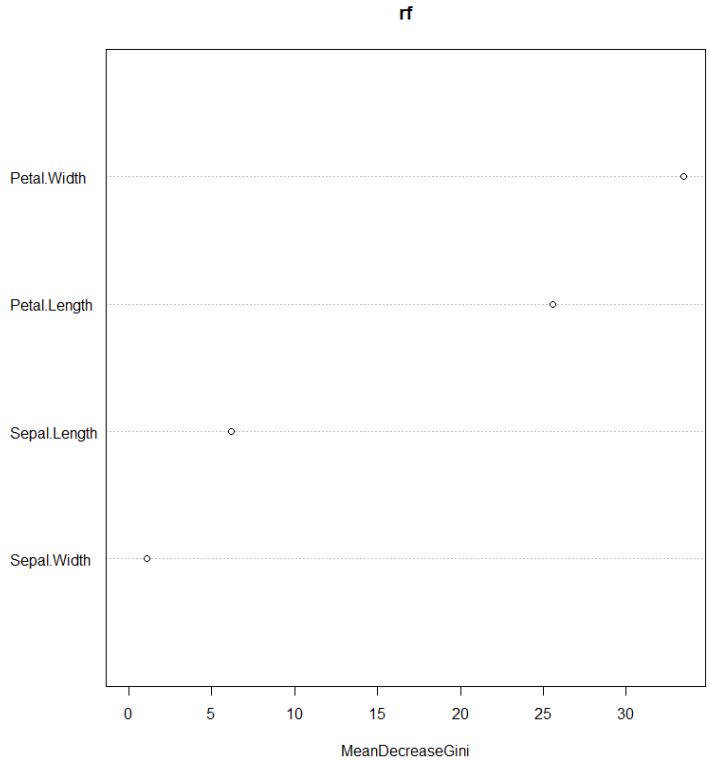
Error Rate of Random Forest

```
> plot(rf, main = '')
```



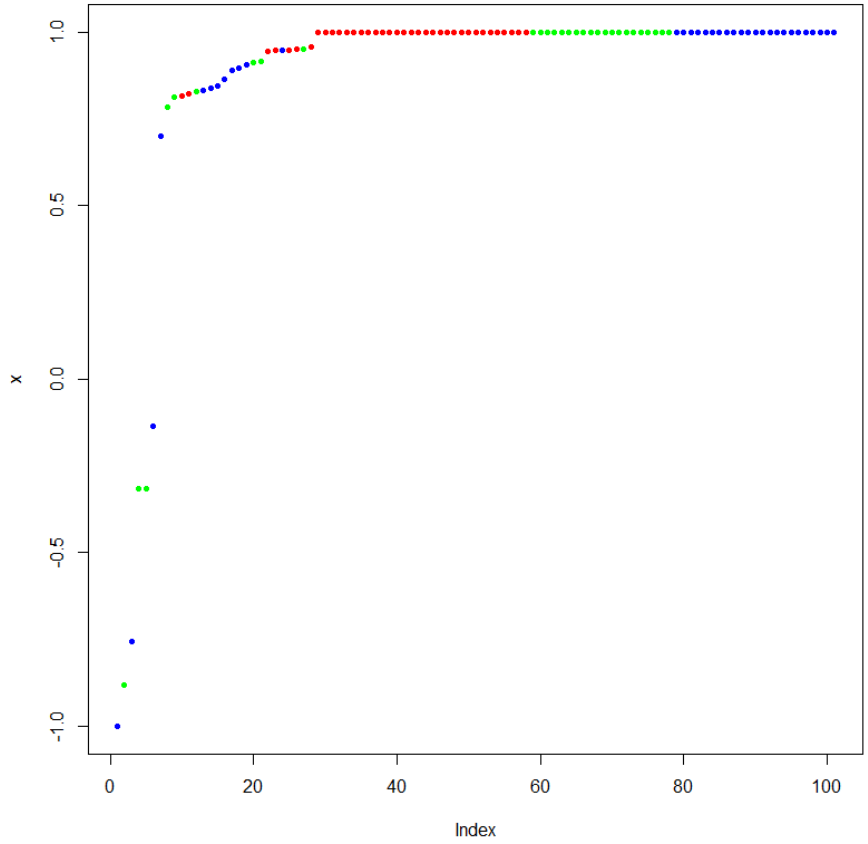
Variable Importance

```
> importance(rf)
             MeanDecreaseGini
Sepal.Length         6.161573
Sepal.Width          1.093108
Petal.Length        25.572946
Petal.Width         33.447076
>
> varImpPlot(rf)
```



Margin of Predictions

```
> irisPred <- predict(rf, newdata = test.data)
> table(irisPred, test.data$Species)

irisPred     setosa versicolor virginica
  setosa         13          0         0
  versicolor      0         20         3
  virginica       0          1        12
>
> plot(margin(rf, test.data$Species))
```



Contents

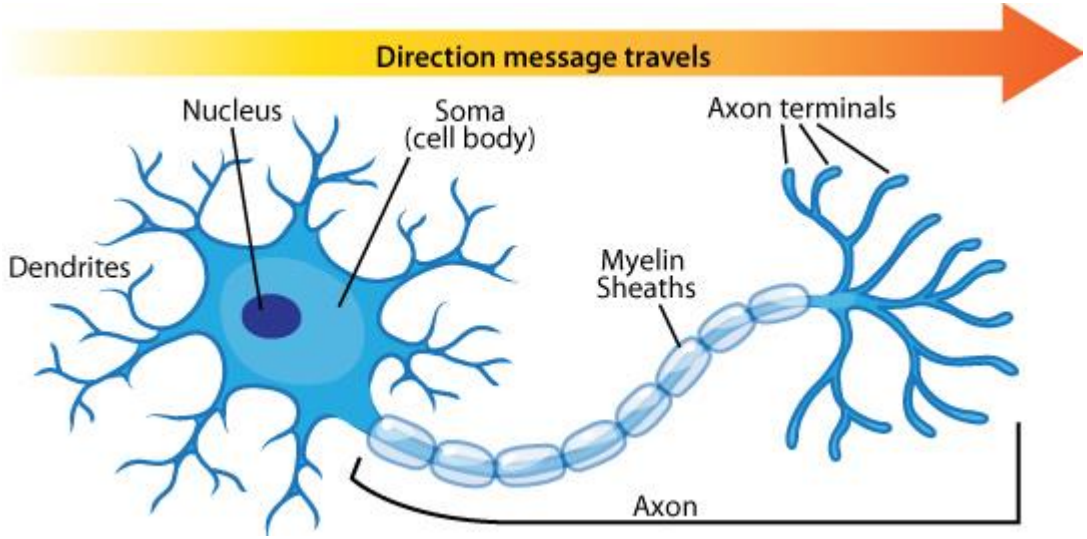
- 분류의 개념
- Decision Trees
- Neural Networks
- SVM (Singular Value Decomposition)

Neural Networks

- 신경회로망
  - 인간의 두뇌 작용을 신경세포(neuron, nerve cell)들간의 연결 관계로 모형
  - 연결주의(connectionism) 혹은 신연결주의(neo-connectionism)라 불림
  - 신경세포의 구조 및 기능을 단순화 하여 수학적 모형(model)으로 표시하고 이런 소자들을 상호 연결하여 신경회로망을 구성
  - 사람의 뇌의 동작을 흉내내어 만든 프로그램이나 데이터 구조 시스템
  - 동물의 뇌에서 일어나는 지능적 정보처리 과정을 단순히 모방함으로써 지능적 행태를 재현하려는 시도
  - 음성 인식, 영상 인식, 감성 인식, 추론 학습 기능 등의 문제 해결
  - 뇌의 구조와 기능을 이해함으로써 뇌가 수행하는 연산 기능의 원리로부터 새로운 개념을 추출하여 구현한 시스템
  - 학습(learning)과 재생(recall)라는 두 단계의 과정
    - 학습(learning): 패턴 부류에 따라 신경망의 연결가중치 조정
    - 재생(recall): 학습된 가중치와 입력벡터와의 거리 계산하여 가장 가까운 클래스로 분류
  - 병렬분산처리(Parallel Distributed Processing: PDP)의 구조와 기능
  - 기능이나 구조에 적합한 특수형 하드웨어(neuro-chip, neural processor) 제작 가능성

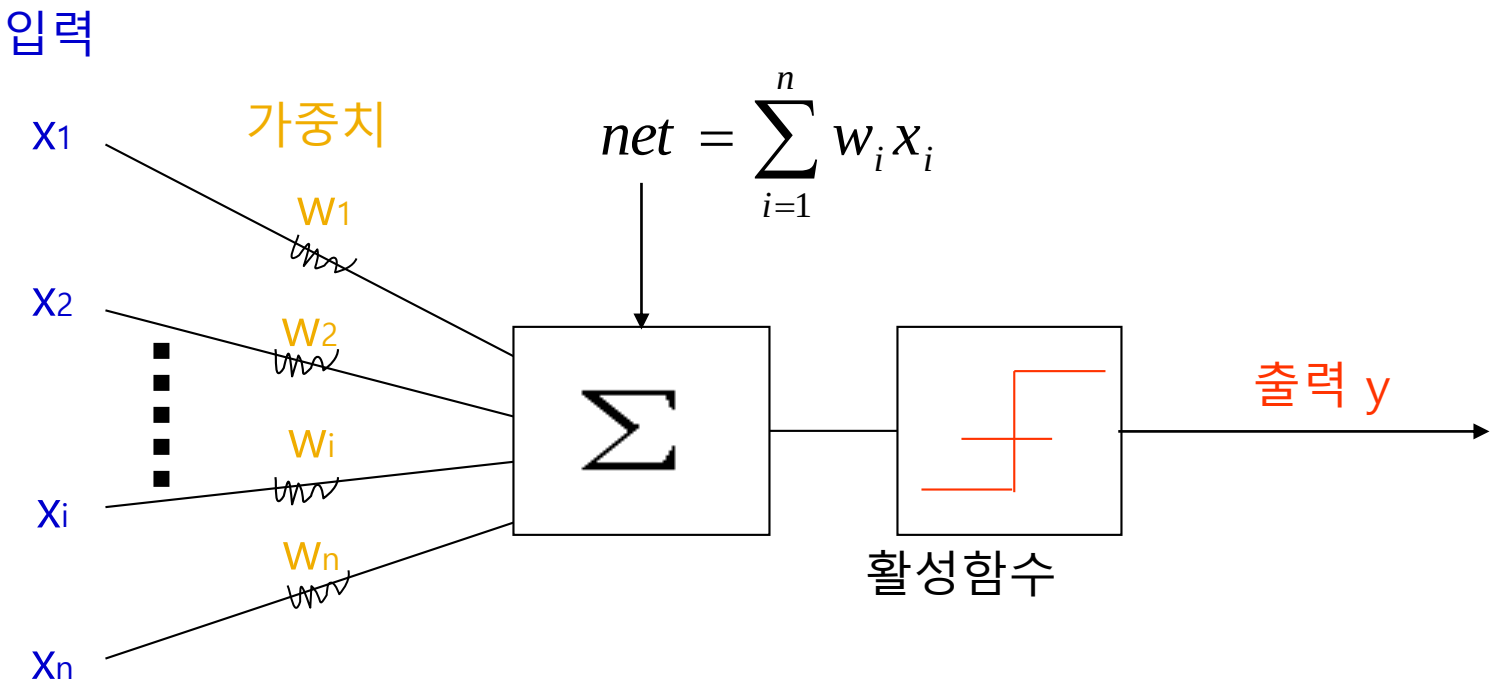
Neural Networks

- 신경세포의 구성 요소
  - 수상돌기(dendrite)
    - 다른 신경세포의 전기화학적 신호를 신경망칩을 통해 받는 기능
  - 신경연접(synapse)
    - 받아들인 자극을 강도에 따라서 활성화(excite) 또는 억제(inhibit)
  - 세포체(cell body, soma)
    - 활성적(active) 입력신호와 억제적(inhibitory) 입력신호들을 합함
  - 축색돌기(axon)
    - 세포체의 점화에 의해 발생되는 전기화학적 에너지를 다른 신경세포까지 연결 전달
  - 신경세포 간의 정보교환은 모두 신경연접를 통하여 행하여지며 신경 정보의 전달은 편 지향적



Neural Networks

- 인공 신경회로망의 처리소자(Processing Unit: PU) 구조



Neural Networks의 특징

- 예제를 통한 학습 가능성
  - 많은 신경망은 예제들로 이루어진 훈련세트(training set)로 학습 가능
  - 지도학습(supervised learning)의 시나리오를 통해 입력 패턴과 원하는 목표 출력과의 차이를 줄여나감
  - 입력값들이 주어지면 비슷한 것들이 모아지는 비지도학습(unsupervised learning)도 있음
  - 보통(비학습) 컴퓨터 프로그램과 차이점
    - -> 프로그램에 의해 미리 정해진 순서를 따라 수행

Neural Networks의 특징

- 결함 허용성
  - 분산 저장 방법의 결과로 정보의 중복된(redundancy) 저장
  - 시스템이 일부 파손되더라도 대부분 작동 가능한 파손내구성(fault tolerant)
- 일반화
  - 신경망은 분류 작업을 수행하는데 있어서 뛰어난 성능을 발휘
  - 이전의 예들로부터 집단의 특징들을 일반화
    - -> 예가 많을 수록 좋은 일반화 가능
- 연상기억(Associative Memory:AM)
  - 패턴의 일부나 약간 다른 패턴으로부터 전체를 얻을 수 있는 연상기억
  - 궤환(feed back)회로의 존재로 서로 영향을 주며 해에 수렴하는 동적(dynamic) 작동원리

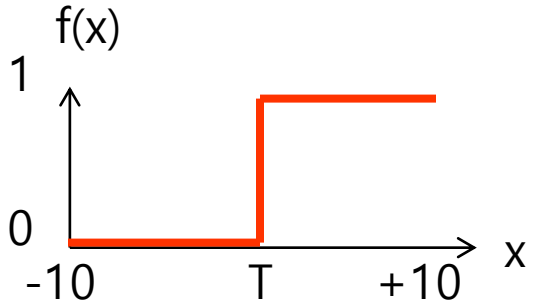
Neural Networks의 요소

- 신경회로망의 구조
  - 처리 소자들간의 연결 및 작용 뱡향 등을 정의
- 노드 특성
  - 각각의 노드에서의 처리 특성
- 학습 규칙
  - 신경회로망이 원하는 결과를 만들어 내도록, 가중치 조절 과정을 통하여 입출력 관계를 근사하게, 하는 학습 알고리즘
- 신경회로망의 구성
  - 입력층: 외부의 입력을 받아들이는 층
  - 출력층: 신경회로망의 처리 결과를 외부로 내보내는 층
  - 가중치: 각 처리소자간의 연결성을 표현하는 연결가중치 (connection weight)

Neural Networks의 요소

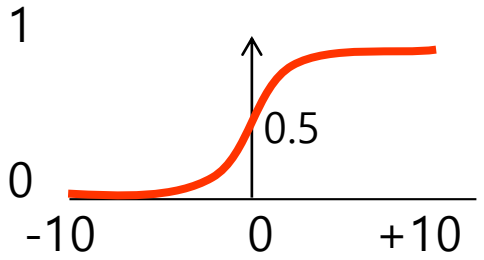
- 신경회로망의 구성
  - 활성(activation) 함수: 가중치합을 입력으로 하여 (보통) 비선형함수를 사용하여 결과값을 출력값으로 사용
    - 스텝함수: 입력이 임계치(threshold)보다 작으면 0을 출력, 임계치보다 크면 1을 출력하는 함수

$$F(x) = \begin{cases} 1 & \text{if } x > T \\ 0 & \text{otherwise} \end{cases}$$

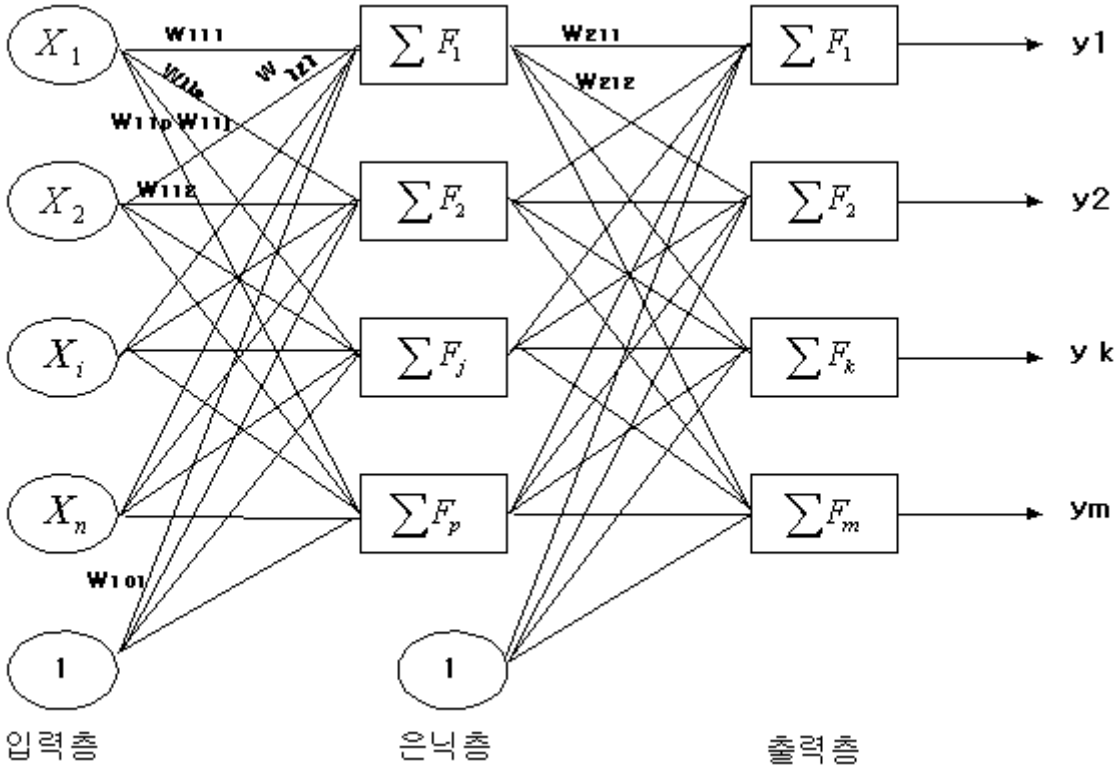


    - 시그모이드 함수: 넓은 범위의 입력에 대한 적절한 이득제어를 제공

$$F(X) = \frac{1}{1+e^{-x}}$$



Multi Layer Perceptron



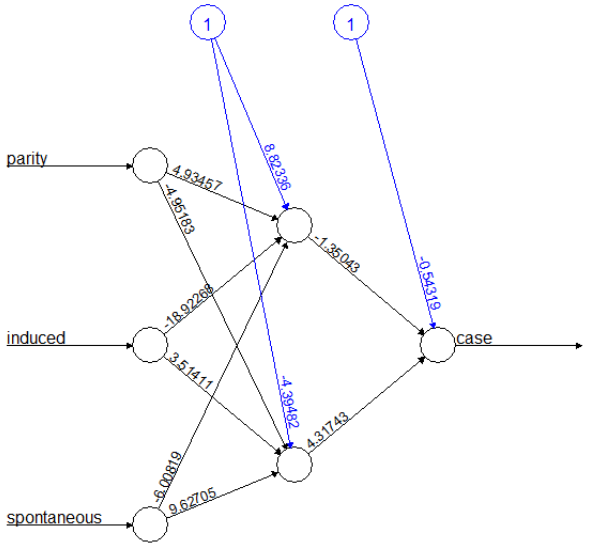
Neural Networks의 장단점

- 장점
  - 다양한 문제에 적용 가능: 인식, 분류, 예측, 시각화, 제어
  - 우수한 일반화 성능
  - 잡음 처리 능력 우수
  - 범주 및 연속형 변수 처리 가능
  - 다양한 S/W 패키지 사용 가능
  - 복잡한 문제에서도 쓸만한 결과를 도출
- 단점
  - 모든 입력값과 출력값의 디지털화 필요
  - 신경망의 학습 결과를 설명하는 것이 어려움
  - 입력 변수의 수에 비례하여 필요한 자료의 수 증가
  - 지역적 해에 수렴하는 학습 경향

Train by neuralnet

- infert dataset: 자연 유산과 인공 유산 후 불임

```
> data(infert, package='datasets')
>
> ind <- sample(2, nrow(infert), replace = T, prob = c(0.7,
0.3))
> train.infert <- infert[ind == 1,]
> test.infert <- infert[ind == 2,]
>
> library(neuralnet)
>
> net.infert <- neuralnet(case~parity+induced+spontaneous, t
rain.infert, hidden = 2, err.fct="ce", linear.output=FALSE,
likelihood=TRUE)
> print(net.infert)
Call: neuralnet(formula = case ~ parity + induced + spontane
ous, data = train.infert,     hidden = 2, err.fct = "ce", li
near.output = FALSE, likelihood = TRUE)

1 repetition was calculated.

        Error         AIC         BIC Reached Threshold Steps
1 86.29320355 194.5864071 228.884339    0.009981068515  2903

> plot(net.infert)
```



Error: 86.293204 Steps: 2903

Train by nnet

```
> library(nnet)
> infert.nn <- nnet(case ~ parity+induced+spontaneous, data=train.infert, size = 2)
# weights:  11
initial  value 43.952629
iter  10 value 29.458231
iter  20 value 29.334072
iter  30 value 29.320350
iter  40 value 29.309292
iter  50 value 29.307787
iter  60 value 29.307701
final  value 29.307696
converged
>
> inf.train.result <- predict(infert.nn, train.infert)
> inf.train.result <- ifelse(inf.train.result > 0.5, 1, 0)
> table(train.infert$case, inf.train.result)
   inf.train.result
     0  1
  0 93 14
  1 27 33
> inf.test.result <- predict(infert.nn, test.infert)
> inf.test.result <- ifelse(inf.test.result > 0.5, 1, 0)
> table(test.infert$case, inf.test.result)
   inf.test.result
     0  1
  0 50  8
  1 10 13
```

Train by nnet

```
> iris.targets <- class.ind(iris$Species)
> iris.new <- cbind(iris[,1:4], iris.targets)
> ind <- sample(2, nrow(iris.new), replace = T, prob
= c(0.7, 0.3))
> train.iris <- iris.new[ind == 1,]
> test.iris <- iris.new[ind == 2,]
> library(nnet)
> iris.nn <- nnet(train.iris[,1:4], train.iris[5:7],
size = 2, rang = 0.1, decay = 5e-4, maxit = 200)
# weights:  19
initial  value 80.984537
iter  10 value 37.996421
iter  20 value 36.130579
iter  30 value 35.902662
iter  40 value 35.878724
iter  50 value 35.828208
iter  60 value 30.068461
iter  70 value 26.367740
iter  80 value 25.858494
iter  90 value 25.725721
iter 100 value 25.690243
iter 110 value 25.591265
iter 120 value 25.576196
iter 130 value 25.575950
final  value 25.575949
converged
```

```
> test.cl <- function(true, pred) {
+ true <- max.col(true)
+ cres <- max.col(pred)
+ table(true, cres)
+ }
>
> test.cl(train.iris[,5:7], predict(iris.nn, train.ir
is[,1:4]))
    cres
true  1  2  3
   1 36  0  0
   2  0 36  3
   3  0  0 33
>
>
> test.cl(test.iris[,5:7], predict(iris.nn, test.iris
[,1:4]))
    cres
true  1  2  3
   1 14  0  0
   2  0 11  0
   3  0  0 17
```

Train by nnet

```
> train.iris2 <- iris[ind == 1,]
> test.iris2 <- iris[ind == 2,]
> iris.nn2 <- nnet(Species ~ ., data = train.iris2, size = 2, rang = 0.1, decay = 5e-4, max
it = 200)
# weights:  19
initial  value 118.439787
iter  10 value 61.006492
iter  20 value 12.738655
iter  30 value 12.125985
iter  40 value 7.602146
iter  50 value 7.218335
iter  60 value 6.977453
iter  70 value 6.426639
iter  80 value 6.387428
iter  90 value 6.382811
iter 100 value 6.377544
iter 110 value 6.376917
iter 120 value 6.376646
iter 120 value 6.376646
final  value 6.376646
converged
> table(train.iris2$Species, predict(iris.nn2, train.iris2, type = "class"))

             setosa versicolor virginica
  setosa         36          0         0
  versicolor      0         38         1
  virginica       0          0        33
> table(test.iris2$Species, predict(iris.nn2, test.iris2, type = "class"))

             setosa versicolor virginica
  setosa         14          0         0
  versicolor      0         11         0
  virginica       0          0        17
```

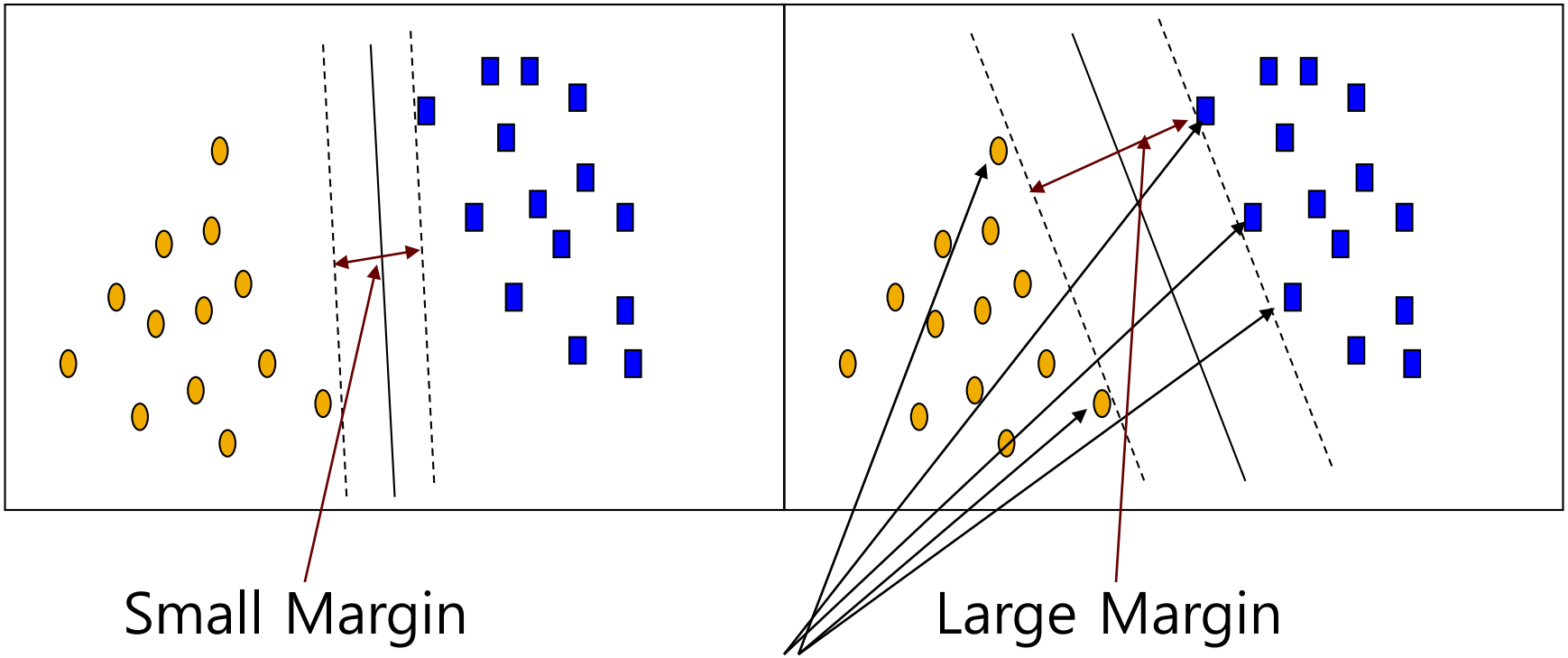
Contents

- 분류의 개념
- Decision Trees
- Neural Networks
- SVM (Singular Value Decomposition)

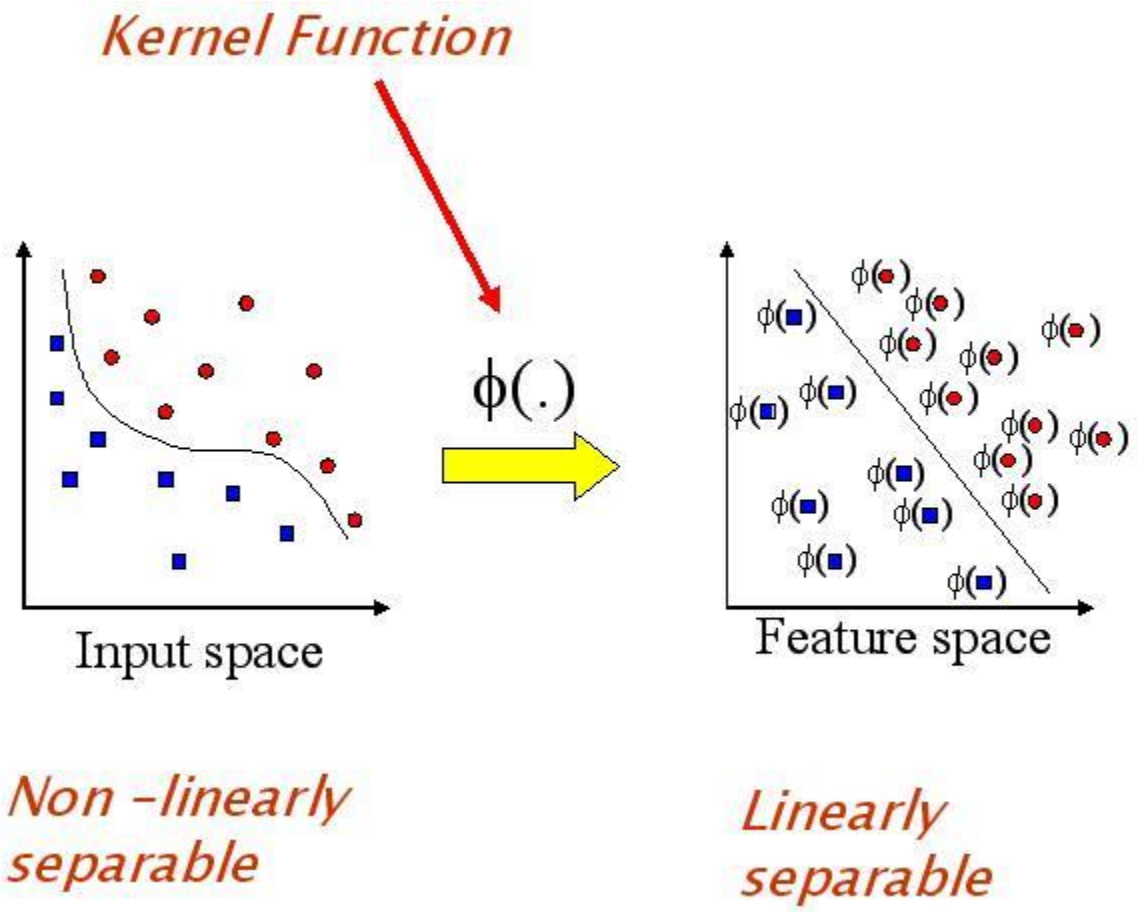
SVM

- A new classification method for both linear and nonlinear data
- It uses a nonlinear mapping to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating hyperplane (i.e., "decision boundary")
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane
- SVM **finds this hyperplane** using support vectors ("essential" training tuples) and margins (defined by the support vectors)

SVM



SVM


Kernel Function
ϕ(.)
Input space
Feature space
Non -linearly separable
Linearly separable

Train by SVM

```
> ind <- sample(2, nrow(iris), replace = T, prob = c(0.7, 0.3))
> train.iris <- iris[ind == 1,]
> test.iris <- iris[ind == 2,]
>
> model <- svm(Species ~ ., data = train.iris)
>
> print(model)

Call:
svm(formula = Species ~ ., data = train.iris)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.25

Number of Support Vectors:  44

>
> summary(model)

Call:
svm(formula = Species ~ ., data = train.iris)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.25

Number of Support Vectors:  44

 ( 7 19 18 )


Number of Classes:  3

Levels:
 setosa versicolor virginica
```

```
> plot(cmdscale(dist(train.iris[,-5])),
+      col = as.integer(train.iris[,5]),
+      pch = c("o","+")[1:150 %in% model$index + 1])
> pred <- predict(model, train.iris[,1:4])
> table(pred, train.iris[,5])

pred         setosa versicolor virginica
  setosa         32          0         0
  versicolor      0         36         3
  virginica       0          2        37
>
> pred2 <- predict(model, test.iris[,1:4])
> table(pred2, test.iris[,5])

pred2        setosa versicolor virginica
  setosa         18          0         0
  versicolor      0         12         0
  virginica       0          0        10
```