모델 적용을 위한 전처리

박찬엽

2017년 7월 19일

목차

- 전처리를 위한 사전 지식
  - 자료형
  - 변수와 함수
- 모델을 위한 전처리
  - 문제 해결이란
  - 군집 알고리즘
  - 회귀 모델
  - 연관 분석

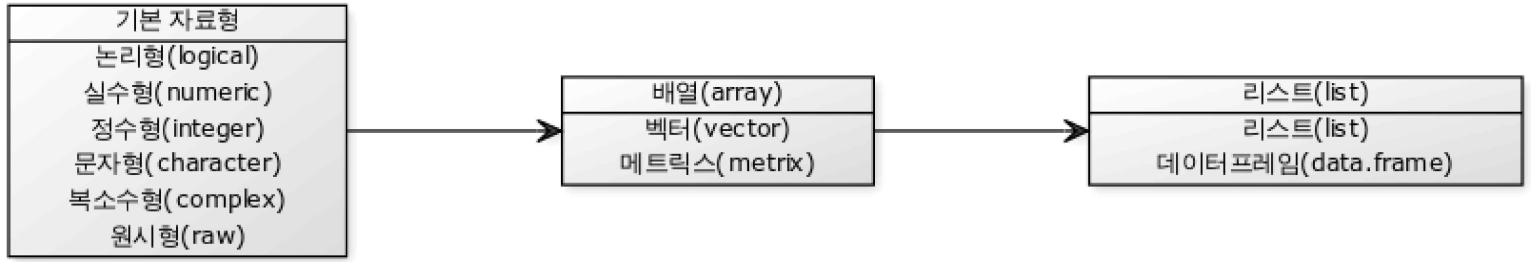
과제 확인

예시 코드

전처리를 위한 사전 지식

자료형

R은 기본 자료형과 벡터, 리스트, 서드파티 자료형을 가지고 있습니다.



기본 자료형

- 논리형: TRUE / FALSE
- 실수형: 숫자 자료형이며 소수점 이하값을 포함하는 형태
- 정수형: 숫자 자료형 소수점 이하가 없음
- 문자형: 글자 자료형이며 " 또는 ' 로 감싸서 표현
- 기타: 복소수나 원시형 등

논리형

R에서 논리형은 logical이라고 표현하고, 모두 대문자인 TRUE와 FALSE가 각각의 의미를 지닙니다. 약어로써 T와 F 또한 사용합니다.

```
TRUE; F
```

```
## [1] TRUE
```

```
## [1] FALSE
```

```
(x<-T)
```

```
## [1] TRUE
```

```
class(x)
```

```
## [1] "logical"
```

실수형

R에서 실수형은 numeric이라고 표현하고, 기본으로 동작하는 숫자 자료형입니다. num이라고 줄여서 작성하기도 합니다. 다른 언어에서는 float이나 double이라고도 합니다.

```
1; (x<-1)
```

```
## [1] 1
```

```
## [1] 1
```

```
class(x)
```

```
## [1] "numeric"
```

```
(y<-1.4)
```

```
## [1] 1.4
```

```
class(y)
```

```
## [1] "numeric"
```

정수형

R에서 정수형은 integer이라고 표현하고, 특별히 정수 숫자 뒤에 L을 함께 작성함으로써 정수형임을 표현합니다. int이라고 줄여서 작성하기도 합니다.

```
class(1)
```

```
## [1] "numeric"
```

```
class(1L)
```

```
## [1] "integer"
```

문자형

R에서 문자형은 character이라고 표현하고, "나 '로 감싸서 다른 객체와 구분하여 표현합니다. chr이나 char라고 줄여서 작성하기도 합니다.

logical글자와 같은 모양의 character를 확인하세요.

```
class(TRUE)
```

```
## [1] "logical"
```

```
class("TRUE")
```

```
## [1] "character"
```

함수와 같은 모양의 character를 확인하세요.

```
class(mean)
```

```
## [1] "function"
```

```
class("mean")
```

```
## [1] "character"
```

데이터 객체와 같은 모양의 character를 확인하세요.

```
x<-data.frame(1)
class(x)
```

```
## [1] "data.frame"
```

```
class("x")
```

```
## [1] "character"
```

배열

배열이란 기본 자료형의 집합입니다. 배열은 연산의 효율을 위하서 만들기 때문에 같은 기본 자료형을 쓰도록 강제하고 있습니다. 몇 차원이냐에 따라 특별한 이름이 있기도 합니다.

- 1D array: 벡터(vector)
- 2D array: 트릭스(matrix)

벡터

벡터는 1차원 배열과 같습니다. 보통 c()함수로 만들어 집니다. 숫자의 연속인 형태의 벡터는 from:to 양식으로 만들기도 합니다. 데이터가 1개인 경우도 벡터이기 때문에 자료형을 확인하기 위해서 class나 is.vector를 사용할 때 의식하고 있어야 합니다. [] 를 이용하여 일부만 가져올 수 있습니다.

```
class(c(1,2,3,4))
```

```
## [1] "numeric"
```

```
class(1:4)
```

```
## [1] "integer"
```

```
is.vector(1:4)
```

```
## [1] TRUE
```

```
is.vector(1)
```

```
## [1] TRUE
```

매트릭스

매트릭스는 2차원 배열로 사람이 가장 익숙한 자료형이며 matrix()함수로 만들 수 있습니다. 1차원의 벡터를 2차원으로 제한한 형태라고 할 수 있습니다. 그렇기 때문에 방향에 따라 벡터의 subset문법도 사용할 수 있습니다. 차원은 ,로 구분합니다.

```
(x<-matrix(1:6, nrow=2))
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
x[1,2]
```

```
## [1] 3
```

```
x[3]
```

```
## [1] 3
```

벡터와 메트릭스는 각 위치의 문법을 표시해주기도 합니다.

```
x<-matrix(1:6, nrow=2)
x[1,]
```

```
## [1] 1 3 5
```

```
x[,3]
```

```
## [1] 5 6
```

리스트

리스트는 여러 테이블을 가진 하나의 데이터베이스와 비슷합니다. 여러 독립적인 벡터들을 모아놓은 자료형을 리스트라고 합니다. 독립적이기 때문에 리스트 내에서 자료형이나 데이터의 길이가 같을 필요가 없습니다.

```
(x<-list(1:6, c(T,F),"play",list(c("A","b"))))
```

```
## [[1]]
## [1] 1 2 3 4 5 6
##
## [[2]]
## [1]  TRUE FALSE
##
## [[3]]
## [1] "play"
##
## [[4]]
## [[4]][[1]]
## [1] "A" "b"
```

리스트 안에 리스트를 가질 수도 있습니다.

```
x<-list(1:6, c(T,F),"play",list(c("A","b")))
str(x)
```

```
## List of 4
##  $ : int [1:6] 1 2 3 4 5 6
##  $ : logi [1:2] TRUE FALSE
##  $ : chr "play"
##  $ :List of 1
##   ..$ : chr [1:2] "A" "b"
```

리스트는 대괄호를 두 개([[,]]) 사용함으로써 벡터와 구분하여 subset을 합니다. 각 리스트의 이름이 있으면 데이터$리스트의이름 형태로 사용할 수도 있습니다.

```
x<-list(num=1:6, c(T,F),"play",list(c("A","b")))
x[[1]]
```

```
## [1] 1 2 3 4 5 6
```

```
x$num
```

```
## [1] 1 2 3 4 5 6
```

데이터프레임

데이터프레임은 앞서 확인한 리스트가 너무 자유도가 높아 사용하기 불편한 점을 개선하기 위해 기능을 제한한 리스트라고 할 수 있습니다. 리스트내의 각 리스트함를 구성하는 벡터로 컬럼을 구성하고 길이가 같아야 합니다.

- 각 컬럼은 벡터로 구성
    - 다른 자료형은 사용하지 못함
- 컬럼의 길이가 모두 같아야 함
- subset은 메트릭스와 같이 [를 1개만 사용
- 리스트와 같이 $을 사용하나 [를 2개를 사용하지 않음

data.frame으로 데이터를 만듭니다.

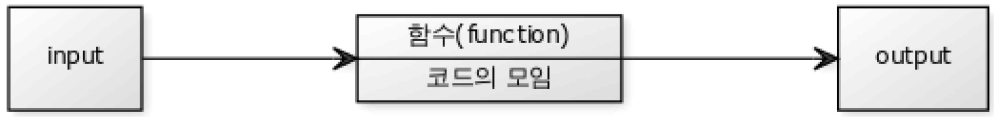
```
(x<-data.frame(A=1:3,b=letters[1:3],c=c(T,F,T),d="play"))
```

```
##   A b     c    d
## 1 1 a  TRUE play
## 2 2 b FALSE play
## 3 3 c  TRUE play
```

변수와 함수

R은 전부 객체로 메모리에 저장합니다. 객체로 저장하여 활용하는 대표적인 예가 변수와 함수인데 변수는 데이터 혹은 데이터의 이름을 뜻합니다.

함수는 특별한 형태의 변수인데, 코드의 묶음을 데이터로 보고 특별한 문법 ()을 사용함으로써 내부의 코드를 실행하는 형태로 사용할 수 있습니다.



함수

함수는 ( ) 안에 사용하는 인자를 정의하고 정의된 대로 사용해야 합니다. 괄호 안에는 필수와 필수가 아닌 인자들이 있고 필수가 아닌 인자들은 기본값을 가집니다.

필수인자가 없는 함수도 있고, 많은 것도 있으며 함수를 만든 사람마다 규칙이 다르기도 합니다.

- ?함수이름 : 함수의 설명을 보여줍니다.
- 함수이름 : 함수의 코드 구성을 직접 출력합니다.
- 함수이름( ) : 함수의 기능을 동작하게 합니다.

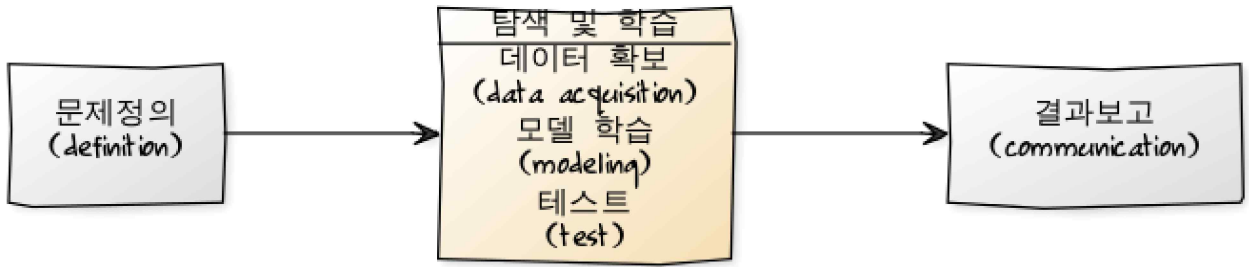
모델을 위한 전처리

문제 해결이란

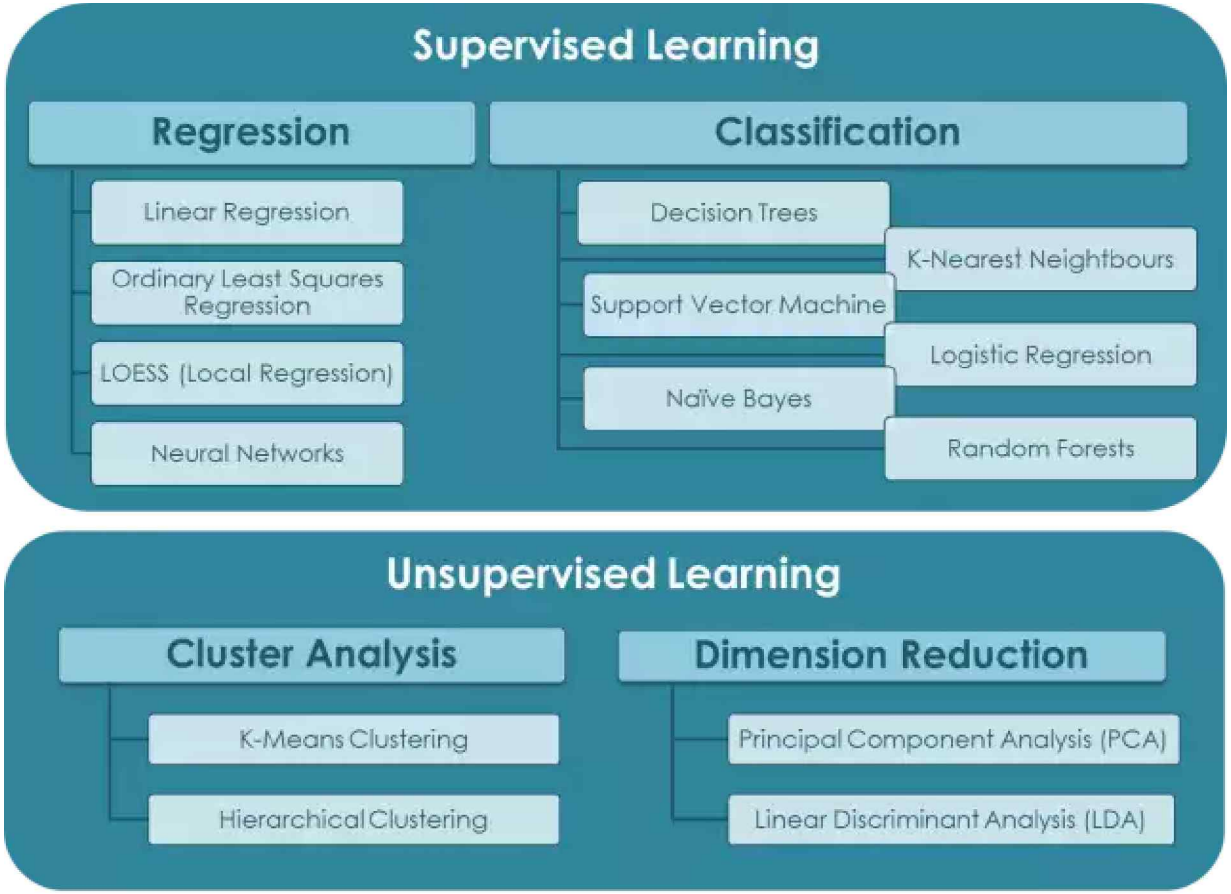
통계 분석이나 기계학습은 문제를 정의하고, 데이터를 기반으로 해결책을 찾고, 결과를 공유하는 과정을 거칩니다.

해결책을 찾는 과정에서 적절한 데이터를 구하고, 문제에 적합한 알고리즘을 선정하거나 개발하여 모델로써 학습하고, 학습한 결과를 평가함으로써 결과를 검증하는 3단계를 계속적으로 수행합니다.

보통은 데이터를 구하고 정제하는데 많은 시간을 사용합니다.



머신 러닝의 종류


Supervised Learning
Regression
Classification
Linear Regression
Ordinary Least Squares Regression
LOESS (Local Regression)
Neural Networks
Decision Trees
K-Nearest Neightbours
Support Vector Machine
Logistic Regression
Naïve Bayes
Random Forests
Unsupervised Learning
Cluster Analysis
Dimension Reduction
K-Means Clustering
Hierarchical Clustering
Principal Component Analysis (PCA)
Linear Discriminant Analysis (LDA)

평가하기

평가를 위해서는 평가를 위한 데이터를 따로 준비해야 합니다. 하지만 데이터를 확보하는데 한계가 있기 때문에 전체 데이터를 일정 부분 남겨두는 방식을 많이 취합니다.

보통 학습 데이터:테스트 데이터의 비율을 7:3 정도로 사용하는 편입니다.

평가가 적절히 이루어 지려면 학습 데이터 중에 테스트 데이터가 겹치는 것이 있지 않는 것이 좋습니다.

군집 알고리즘

군집 알고리즘에서 가장 유명한 것이 k-means 입니다. 거리가 가까운 것을 기준으로 k 개의 군집을 만들어 줍니다. 거리를 기준으로 하기 때문에 변수들이 모두 같은 단위이거나 해야 합니다.

함수는 kmeans를 사용하며 수행한 결과가 바로 결과물입니다.

```
km <- kmeans(data, centers = n, iter.max = m)
km$cluster
```

모델 학습

```
km <- kmeans(data, centers = k, iter.max = m)
```

- kmeans : k-means 알고리즘을 수행하는 함수입니다.
- data : k-means 알고리즘을 수행할 대상 데이터입니다.
- centers : k를 뜻하는 군집의 갯수입니다. 중심을 몇 개로 할 것인지 정해야 하며 필수값입니다.
- iter.max: k-means는 반복 계산을 통해 결과를 산출하기 때문에 적당히 반복 계산이 되면 멈추는 것이 효율적입니다. 최대 반복 횟수를 지정합니다. 기본값으로 10이 되어 있습니다.

데이터

k-means를 사용하는 데이터는 군집을 거리를 기반으로 하기 때문에 거리에 대해 같은 의미를 지니는 변수만을 바탕으로 군집을 시도해야 합니다.

위에 조건에 부합하는 변수만을 사용하는 것은 너무 어렵기 때문에 다른 것을 함께 사용할 때 조정을 해야 할 필요성이 있습니다.

예를들어 키와 몸무게로 사람들을 군집화를 시도하면 키가 몸무게에 비해 차이가 클 것이기 때문에 키의 차이가 군집을 결정하는데 더 영향을 많이 미칩니다.

이런 부분을 조정하기 위한 방법이 표준화(scale)입니다.

표준화

R은 좋은 표준화 함수를 제공하고 있습니다. scale은 숫자형의 벡터를 받아서 중심을 0으로 바꾸는 center와 분포를 표준정규 분포에 맞추는 scale 옵션을 가지고 있습니다.

```
scale(data, center=T, scale=T)
```

- scale : 표준화를 진행하기 위한 함수 이름입니다.
- data : 표준화 대상이 되는 데이터입니다. 숫자형만 들어올 수 있습니다.
- center: 중심을 0으로 조절할 것인지를 결정합니다. 기본 값은 TRUE이고 FALSE를 하면 숫자 벡터의 평균을 중심으로 유지합니다.
- scale : 분포를 표준정규 분포로 조절할 것인지를 결정합니다. 기본 값은 TRUE이고 FALSE를 하면 숫자 벡터의 분산을 그대로 유지합니다.

실습

iris 데이터는 군집화의 사례로 가장 유명한 데이터입니다. 꽃입의 길이와 폭, 꽃술의 길이와 폭으로 iris의 종류를 구별합니다.

```
data(iris)
train<-iris[,-5]
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 ...
```

실습

1. train 데이터의 자료형을 파악하세요.
2. k-means 함수에 군집 3개, 반복 횟수 100회로 결과를 만드세요.
3. scale 함수를 이용해서 표준화를 진행한 후 위의 결과를 만드세요.
4. 두 결과를 비교해 보세요.

군집과 분류

군집이란 정답이 없는 나누기를 의미하고, 분류는 이미 정해진 기준에 맞게 잘 선택하는 것을 뜻합니다. 그러다 보니 군집과 분류의 가장 큰 차이점은 정답을 모델이 활용하는지 안하는지라고 볼 수 있습니다.

iris데이터에는 정답이 있지만, 답을 확인할 때만 활용할 뿐, 학습에 사용하지 않는 k-means 알고리즘을 사용했습니다. 정답은 군집을 성능을 평가할 때 사용할 수 있을 것입니다.

회귀 모델

회귀 모델을 사용하는 함수는 대표적으로 lm이 있습니다. 이외에 glm이나 rlm 등이 있습니다. 데이터가 준비되었다면, 모델을 세우고 평가하는 코드가 짧은 편입니다.

```
fit <- lm(y ~ x1 + x2 + x3, data=mydata)
summary(fit)
predict(fit, testData, interval="predict")
```

모델 학습

```
fit <- lm(y ~ x1 + x2 + x3, data=mydata)
```

- lm: linear model의 약어이며 모델을 학습하는 함수입니다.
- fit: 결과를 저장한 데이터 객체로 학습한 모델의 정보를 가지고 있습니다.
  - 결과라는 것이 학습한 모델을 뜻하는 것이지 예측 결과를 의미하는 것이 아닙니다.
- ~ : 첫번째 인자인 식을 작성하는 방식으로 ~을 사용합니다.
- data: 사용하는 데이터를 지정합니다. 지정하면 앞에 식을 작성할 때 컬럼 이름만 으로 작성하는 등의 문법적 혜택이 있습니다.

```
fit <- lm(y ~ . , data=mydata)
```

- . : mydata의 컬럼 중에 y를 제외하고 모두 x로 사용하고 싶으면 남은 변수 모두라는 의미로 사용합니다.

데이터

일반적인 회귀 모델은 입력 변수들을 바탕으로 연속형 숫자가 결과로 나오는 모델입니다. 그렇기 때문에 Y(=종속변수=결과=정답=라벨=아웃풋=...)에 해당하는 변수가 numeric이어야 합니다.

X(=독립변수=데이터=요인=특성=인풋-=...)에 해당하는 변수 또한 numeric이어야 합니다. 하지만 dummy 변수라는 방법으로 명목형 데이터를 사용할 수 있습니다.

더미 변수란

더미 변수란 0과 1을 값으로 가지는 변수를 의미합니다. 회귀 모델에서 명목형 데이터를 X로 사용하는 방법은 하나의 명목형 데이터를 여러 더미 변수로 변경하여 사용하는 것입니다.

회귀식에서 더미 변수의 의미는 0일 때 해당 변수가 적용되지 않음, 1일때 적용됨을 뜻합니다.

R에서는 명목형 데이터를 character로 두지 않고 factor를 만듬으로써 쉽게 적용할 수 있습니다.

실습

```
library(MASS)
data(birthwt)
str(birthwt)
```

```
## 'data.frame':    189 obs. of  10 variables:
##  $ low  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ age  : int  19 33 20 21 18 21 22 17 29 26 ...
##  $ lwt  : int  182 155 105 108 107 124 118 103 123 113 ...
##  $ race : int  2 3 1 1 1 3 1 3 1 1 ...
##  $ smoke: int  0 0 1 1 1 0 0 0 1 1 ...
##  $ ptl  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ht   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ui   : int  1 0 0 1 1 0 0 0 0 0 ...
##  $ ftv  : int  0 3 1 2 0 0 1 1 1 0 ...
##  $ bwt  : int  2523 2551 2557 2594 2600 2622 2637 2637 2663 2665 ...
```

데이터 설명

birthwt: 1986년 미국의 Messachusetts 주 Springfield에 있는 Baystate Medical Center에서 수집한 189명에 대한 출산 관련 정보 codebook

- low : 2.5.kg보다 작게 태어났으면 1, 아니면 0
- age : 출산 당시 어머니의 나이
- lwt : 마지막 생리 기간 중 어머니의 몸무게(pounds)
- race : 어머니의 인종. 백인 1, 흑인 2, 그외 3
- smoke: 임신중 흡연 여부
- ptl : 과거 조산 횟수
- ht : 고혈압의 발병 이력
- ui : uterine irritability 여부
- ftv : 첫번째 임신기간중 의사 방문 횟수
- bwt : 아기의 몸무게(grams)

실습

1. 데이터를 보고 각 컬럼의 자료형을 파악하세요.
2. 변수 설명과 의미에 따라 더미 변수로 사용해야 하는 것을 파악하세요.
3. fit<-lm(수식, data=birthwt) 에서 수식을 완성하세요
4. summary(fit)을 확인해 보세요.

예측 이라면

```
set.seed(1234)
tar   <- sample(1:nrow(birthwt),round(nrow(birthwt)*0.7))
train <- birthwt[tar, ]
test  <- birthwt[-tar, ]
fit<-lm(수식, data=train)
predict(fit, test)
```

- predict: 학습에 결과로 만들어진 수식을 새로운 데이터에 적용해서 결과를 만드는 함수입니다.
- fit : 기 학습된 결과물을 저장한 객체입니다.
- test : 테스트로 사용할 새로운 데이터입니다.

연관 분석

연관 분석은 장바구니 분석이라고도 하며 동시에 랜덤 갯수의 아이템이 같이 소비된 데이터를 바탕으로 어떤 아이템들이 같이 소비되었는지를 분석하는 방법입니다.

연관 분석에서는 3가지 개념을 소개하고 있습니다.

- 지지도(support) : 전체 데이터에서 아이템이 나타난 확율. 빈도의 의미이기 때문에 지지도가 높은 제품을 우선 분석해보는데 기준이 됩니다.
- 신뢰도(confidence): 기준 아이템이 나타났을 때 대상 아이템이 나타난 조건부 확율. 신뢰도가 높을 수록 유용한 규칙이라는 것을 알 수 있습니다.
- 향상도(lift) : 신뢰도 / 지지도 로써 1보다 클수록 우연적이지 않다는 뜻이 됩니다.

대표적인 비지도 학습이며 위의 3가지 개념을 바탕으로 계산한 결과를 사람이 판단하는 방식입니다. 빈도에 의존하기 때문에 데이터가 많을 수록 실제하는 패턴을 찾기가 좋아집니다.

apriori 알고리즘

```
rule <- apriori(data = transactionsData,
                parameter = list(support = n,
                                 confidence = m,
                                 minlen = l))
```

- apriori : 연관 분석을 실행하는 함수의 이름입니다.
- data : 분석 대상 데이터로 transactions라는 자료형을 입력으로 받습니다.
- parameter: supoort, confidence등의 기준값을 제시하는 인자로, 입력한 값보다 큰 결과들만 계산합니다.
- rule : 계산된 결과로 연관 규칙에 대한 정보를 담고 있습니다.

자료형 소개

```
if (!require(arules)) install.packages("arules")
```

```
## Loading required package: arules
```

```
## Warning: package 'arules' was built under R version 3.4.1
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
library(arules)
data(Groceries)
str(Groceries)
```

연관분석에서 사용하는 transactions 자료형은 특별히 arules 패키지에서 사용하기 위해 고안되었습니다.

```
summary(Groceries)
```

```
## transactions as itemMatrix in sparse format with
##  9835 rows (elements/itemsets/transactions) and
##  169 columns (items) and a density of 0.02609146
##
## most frequent items:
##       whole milk other vegetables       rolls/buns             soda
##             2513             1903             1809             1715
##           yogurt          (Other)
##             1372            34055
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
## 2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55
##   16   17   18   19   20   21   22   23   24   26   27   28   29   32
##   46   29   14   14    9   11    4    6    1    1    1    1    3    1
##
```

그렇기 때문에 가지고 있는 데이터를 transactions 자료형으로 변경하는 것이 중요합니다.

```
dvdread<-read.csv("./data/dvdtrans.csv")
head(dvdread)
dvdtran<-read.transactions("./data/dvdtrans.csv")
inspect(dvdtran[1:10,])
```

실습

1. dvdtrans 데이터를 본래 transactions 자료형에 맞게 입력해 보세요. 아래 데이터처럼 나와야 됩니다.

```
load("./data/dvdTranSample.RData")
inspect(dvdTranSample)
```

1. Groceries 데이터를 사용해서 apriori 함수로 연관규칙을 만들어 보세요. support는 0.01, confidence는 0.2, minle은 2로 작성합니다.

1. ./data/tran.csv 데이터를 transactions 자료형에 맞게 입력해보세요.

과제