Module4L4

Neha Parulekar

February 28, 2016

## Additional packages needed

To run the code in M04\_Lesson\_04.Rmd you may need additional packages.

* If necessary install these packages.

install.packages("arules");  
install.packages("arulesViz");  
install.packages("Matrix");

require(arules)

## Loading required package: arules

## Loading required package: Matrix

##   
## Attaching package: 'arules'

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

require(arulesViz)

## Loading required package: arulesViz

## Loading required package: grid

require(Matrix)

**Loading the data** I choose chess data from the link given.

# Checking the working directory and loading the data  
setwd("c:/Users/Neha/Desktop")  
ChessData <- read.table(file = "chess.dat" ,sep=",")  
head(ChessData)

## V1  
## 1 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74   
## 2 1 3 5 7 9 12 13 15 17 19 21 23 25 27 29 31 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74   
## 3 1 3 5 7 9 12 13 16 17 19 21 23 25 27 29 31 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74   
## 4 1 3 5 7 9 11 13 15 17 20 21 23 25 27 29 31 34 36 38 40 42 44 47 48 50 52 54 56 58 60 62 64 66 68 70 72 74   
## 5 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 34 36 38 40 42 44 46 48 51 52 54 56 58 60 62 64 66 68 70 72 74   
## 6 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 34 36 38 40 42 44 46 48 51 52 54 56 58 60 63 64 66 68 70 72 74

summary(ChessData)

## V1   
## 1 3 5 7 10 11 13 15 17 19 22 23 25 27 29 32 34 36 39 40 43 45 46 48 50 52 54 56 58 60 62 64 67 69 71 73 74 : 1   
## 1 3 5 7 10 11 13 16 18 20 22 23 26 27 29 32 34 36 39 40 42 45 46 48 50 52 54 56 58 60 62 64 67 69 71 73 74 : 1   
## 1 3 5 7 10 11 14 15 17 19 21 23 25 27 29 31 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 71 72 74 : 1   
## 1 3 5 7 10 11 14 15 17 19 21 23 25 27 29 31 34 36 38 40 42 44 46 48 50 52 55 56 58 60 62 64 66 68 71 72 75 : 1   
## 1 3 5 7 10 11 14 15 17 19 21 23 25 27 29 31 34 36 38 40 42 44 46 48 51 52 54 56 58 60 62 64 66 68 71 72 74 : 1   
## 1 3 5 7 10 11 14 15 17 19 21 23 25 27 29 31 34 36 38 40 42 44 46 48 51 52 55 56 58 60 62 64 66 68 71 72 75 : 1   
## (Other) :3190

# # default settings result in zero rules learned  
ChessData.d <- apriori(ChessData, parameter = list(support = 0.1, confidence = 0.8, minlen = 1))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport support minlen maxlen  
## 0.8 0.1 1 none FALSE TRUE 0.1 1 10  
## target ext  
## rules FALSE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 319   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[3196 item(s), 3196 transaction(s)] done [0.00s].  
## sorting and recoding items ... [0 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 done [0.00s].  
## writing ... [0 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

summary(ChessData.d)

## set of 0 rules

# set better support and confidence levels to learn more rules  
  
# Tried different sets of rules but still get the output value as 0.  
ChessData.d <- apriori(ChessData, parameter = list(support = 0.02, confidence = 0.9, minlen = 2))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport support minlen maxlen  
## 0.9 0.1 1 none FALSE TRUE 0.02 2 10  
## target ext  
## rules FALSE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 63   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[3196 item(s), 3196 transaction(s)] done [0.00s].  
## sorting and recoding items ... [0 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 done [0.00s].  
## writing ... [0 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

summary(ChessData.d)

## set of 0 rules

ChessData.d <- apriori(ChessData, parameter = list(support = 0.01, confidence = 0.5, minlen = 4))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport support minlen maxlen  
## 0.5 0.1 1 none FALSE TRUE 0.01 4 10  
## target ext  
## rules FALSE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 31   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[3196 item(s), 3196 transaction(s)] done [0.00s].  
## sorting and recoding items ... [0 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 done [0.00s].  
## writing ... [0 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

summary(ChessData.d)

## set of 0 rules

ChessData.d <- apriori(ChessData, parameter = list(support = 0.05, confidence = 0.5, minlen = 1))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport support minlen maxlen  
## 0.5 0.1 1 none FALSE TRUE 0.05 1 10  
## target ext  
## rules FALSE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 159   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[3196 item(s), 3196 transaction(s)] done [0.00s].  
## sorting and recoding items ... [0 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 done [0.00s].  
## writing ... [0 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

summary(ChessData.d)

## set of 0 rules

## **Association Rules with Apriori Algorithms**

Assoiation rules with Apriori Algorithms is generally used for mining frequent itemsets.The Apriori algorithm employs level-wise search for frequent itemsets. The implementation of Apriori used includes some improvements (e.g., a prefix tree and item sorting).

### **Usage**

apriori(data, parameter = NULL, appearance = NULL, control = NULL)

### Arguments

**Data** any data structure which can be coerced into transactions (e.g., a binary matrix or data.frame).

**Parameter** object of class APparameter or named list. The default behavior is to mine rules with support 0.1, confidence 0.8, and maxlen 10.

**Appearance** object of class APappearance or named list. With this argument item appearance can be restricted (implements rule templates). By default all items can appear unrestricted.

**Control** object of class APcontrol or named list. Controls the algorithmic performance of the mining algorithm (item sorting, etc.)

### **Values**

Returns an object of class rules or itemsets.

#### Reference : <http://www.inside-r.org/packages/cran/arules/docs/apriori>