Case study: recommendation system with bnlearn

Aaron Smith

6/20/2020

Let’s build a recommendation system using R’s bnlearn package. Our goal is to create a model that identifies what a customer would be prone to adding to their shopping basket. At all times we want to get the customer to add one more item to their basket; once they add an item, we update our recommendation to get them to add another item.

# Conditional probability models

We can use the arules package in R to build rules for this task. Here we want to use a graphical model. There are advantages and disadvantages either way. My opinion is that graphical models have a more natural interface, and it is easier to identify meaningful rules.

We will use the Groceries data set from the arules package.

options(digits = 1)  
data(Groceries,package = "arules")  
Groceries

## Loading required package: arules

## Loading required package: Matrix

##   
## Attaching package: 'arules'

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

## transactions in sparse format with  
## 9835 transactions (rows) and  
## 169 items (columns)

The data is stored in a sparse matrix format. We will need to convert it to a data.frame() before passing it to the bnlearn functions.

M\_Groceries <- as.data.frame(as(  
 object = Groceries,  
 Class = "matrix"  
))  
colnames(M\_Groceries) <- make.names(colnames(M\_Groceries))  
M\_Groceries[1:6,1:6]

## frankfurter sausage liver.loaf ham meat finished.products  
## 1 FALSE FALSE FALSE FALSE FALSE FALSE  
## 2 FALSE FALSE FALSE FALSE FALSE FALSE  
## 3 FALSE FALSE FALSE FALSE FALSE FALSE  
## 4 FALSE FALSE FALSE FALSE FALSE FALSE  
## 5 FALSE FALSE FALSE FALSE FALSE FALSE  
## 6 FALSE FALSE FALSE FALSE FALSE FALSE

TRUE means the item is in the basket one or more times. FALSE means the item is not in the basket.

To make it easier to interpret what is going on as we build the model(s), lets sort the rows and columns by their row sums, column sums. The items that appear in the most baskets will be in the left most columns; the baskets with the most distinct items are at the top.

M\_Groceries <- M\_Groceries[  
 order(rowSums(M\_Groceries),decreasing = TRUE),  
 order(colSums(M\_Groceries),decreasing = TRUE)  
]

# Reduce the number of items in the model

* Conditional probability models take a long time to compute.
* Recommendations for rarely purchased items probably will be unsuccessful.
* Recommendations based on rarely purchased items will rarely get used.

Let’s trim down the items in our model to the ones that account for 80% of the distinct items in a basket.

v\_sums <- colSums(M\_Groceries)  
mean(cumsum(v\_sums)/sum(v\_sums) <= 0.8)

## [1] 0.3

sum(cumsum(v\_sums)/sum(v\_sums) <= 0.8)

## [1] 53

v\_names <- names(v\_sums[cumsum(v\_sums)/sum(v\_sums) <= 0.8])  
M\_Groceries <- M\_Groceries[,v\_names]

To make output more compact, I converted TRUE/FALSE to “1”/“0” factors. Notice that zero and one are categorical instead of numeric. The bnlearn package would have build multiple linear regression models (numeric target variables); we want binary target variables.

for(j in 1:ncol(M\_Groceries)){  
 M\_Groceries[,j] <- factor(as.numeric(M\_Groceries[,j]))  
}

# Building the model

Here are the possible algorithms in bnlearn that we can use to build the model. They are grouped by their method of identifying relationships.

Before writing this, I benchmarked each algorithm on a subset of the data. Each group is ordered by computation time.

* iamb.fdr() was the fastest constraint-based algorithm.
* hc() was the fastest score-based algorithm.
* h2pc() was the fastest hybrid algorithm.
* aracne() was the fastest local discovery algorithm (I have never had success with the local discovery algorithms).

My experience is that the constraint-based and score-based algorithms are the best ones for my tasks at work.

v\_algorithms <- c(  
 "iamb.fdr","fast.iamb","inter.iamb","gs","iamb","hpc","si.hiton.pc","mmpc","pc.stable",  
 "hc","tabu",  
 "h2pc","mmhc","rsmax2",  
 "aracne","chow.liu"  
)  
require(bnlearn)

## Loading required package: bnlearn

##   
## Attaching package: 'bnlearn'

## The following object is masked from 'package:arules':  
##   
## discretize

## The following object is masked from 'package:stats':  
##   
## sigma

For this presentation, I did not run this code chunk. It took too long to run. I would run this overnight if I needed the best model possible. Then use BIC(), AIC(), logLik(), or another network score to determine which model to use.

## list\_bnlearn <- list()  
## for(j in v\_algorithms) try({  
## list\_bnlearn[[j]] <- do.call(  
## what = j,  
## args = list(x = M\_Groceries[,v\_names])  
## )  
## },silent = TRUE)

My experience is that hc() (hill-climber) gives solid results, and is fast. For this case study, let’s use hc() as our model.

list\_bnlearn <- list()  
for(j in "hc") try({  
 list\_bnlearn[[j]] <- do.call(  
 what = j,  
 args = list(x = M\_Groceries[,v\_names])  
 )  
},silent = TRUE)

Unfortunately, the bnlearn package has multiple model structures. To fully take advantage of the model, we need to convert the model using bn.fit()

bn\_Groceries <- bn.fit(  
 x = list\_bnlearn[["hc"]],  
 data = M\_Groceries[,v\_names]  
)

# Scoring some data

Let’s run the model on some baskets in the data, and see what the model recommends.

v\_subset <- sort(sample.int(  
 n = nrow(M\_Groceries),  
 size = 6  
))  
subset\_Groceries <- M\_Groceries[v\_subset,]  
predict\_Groceries <- matrix(  
 data = 0,  
 nrow = nrow(subset\_Groceries),  
 ncol = ncol(subset\_Groceries)  
)  
rownames(predict\_Groceries) <- rownames(subset\_Groceries)  
colnames(predict\_Groceries) <- colnames(subset\_Groceries)  
v\_index <- 1:ncol(subset\_Groceries)

The predict() function gives “0” or “1” as its standard output. We need to extract probabilities (expected value of the Bernoulli variable). We use attr() to get the estimated expected value.

for(j in 1:nrow(subset\_Groceries)){  
 for(k in v\_index[subset\_Groceries[j,] == "0"]){  
 predict\_Groceries[j,k] <- attr(  
 x = predict(  
 object = bn\_Groceries,  
 data = subset\_Groceries[j,-k],  
 node = v\_names[k],  
 prob = TRUE  
 ),  
 which = "prob"  
 )["1",1]  
 }  
}  
predict\_Groceries[,1:6]

## whole.milk other.vegetables rolls.buns soda yogurt bottled.water  
## 4121 0.2 0.3 0.0 0.0 0.1 0.15  
## 6784 0.3 0.0 0.2 0.0 0.2 0.15  
## 3337 0.2 0.2 0.0 0.0 0.1 0.00  
## 3538 0.3 0.0 0.2 0.2 0.2 0.09  
## 9569 0.0 0.2 0.2 0.2 0.2 0.00  
## 785 0.0 0.2 0.2 0.2 0.2 0.09

Let’s reshape the data so we can see the relationship between items in a basket, and recommended items.

list\_predict <- list()  
for(j in 1:nrow(subset\_Groceries)){  
 list\_predict[[j]] <- list()  
 list\_predict[[j]]$In\_Basket <- v\_names[subset\_Groceries[j,] == "1"]  
 list\_predict[[j]]$Promote <- sort(predict\_Groceries[j,predict\_Groceries[j,] > 0],decreasing = TRUE)  
}  
list\_predict

## [[1]]  
## [[1]]$In\_Basket  
## [1] "rolls.buns" "soda" "sausage"   
## [4] "pastry" "canned.beer" "pip.fruit"   
## [7] "fruit.vegetable.juice" "salty.snack" "hygiene.articles"   
## [10] "candy"   
##   
## [[1]]$Promote  
## other.vegetables shopping.bags whole.milk   
## 0.287 0.234 0.203   
## bottled.water newspapers frankfurter   
## 0.147 0.134 0.104   
## yogurt white.bread specialty.chocolate   
## 0.100 0.082 0.078   
## long.life.bakery.product bottled.beer chocolate   
## 0.078 0.073 0.071   
## root.vegetables tropical.fruit misc..beverages   
## 0.068 0.061 0.055   
## ham coffee brown.bread   
## 0.053 0.052 0.051   
## citrus.fruit meat napkins   
## 0.049 0.044 0.042   
## pork margarine frozen.vegetables   
## 0.039 0.038 0.038   
## domestic.eggs whipped.sour.cream UHT.milk   
## 0.038 0.035 0.035   
## waffles dessert beef   
## 0.032 0.029 0.029   
## butter chicken curd   
## 0.029 0.028 0.027   
## cream.cheese. specialty.bar berries   
## 0.026 0.025 0.024   
## frozen.meals sugar oil   
## 0.023 0.022 0.020   
## butter.milk hamburger.meat onions   
## 0.019 0.018 0.017   
## beverages   
## 0.001   
##   
##   
## [[2]]  
## [[2]]$In\_Basket  
## [1] "other.vegetables" "soda" "tropical.fruit" "sausage"   
## [5] "pastry" "frankfurter" "salty.snack" "hamburger.meat"   
##   
## [[2]]$Promote  
## whole.milk citrus.fruit pip.fruit   
## 0.34 0.25 0.25   
## shopping.bags root.vegetables rolls.buns   
## 0.23 0.23 0.18   
## yogurt bottled.water whipped.sour.cream   
## 0.18 0.15 0.12   
## fruit.vegetable.juice pork frozen.vegetables   
## 0.11 0.10 0.09   
## margarine meat domestic.eggs   
## 0.09 0.09 0.08   
## chicken bottled.beer beef   
## 0.08 0.07 0.07   
## chocolate newspapers butter   
## 0.07 0.07 0.06   
## onions cream.cheese. ham   
## 0.06 0.06 0.05   
## coffee frozen.meals brown.bread   
## 0.05 0.05 0.05   
## candy canned.beer napkins   
## 0.05 0.05 0.04   
## oil butter.milk UHT.milk   
## 0.04 0.04 0.03   
## waffles long.life.bakery.product white.bread   
## 0.03 0.03 0.03   
## dessert specialty.chocolate curd   
## 0.03 0.03 0.03   
## misc..beverages beverages specialty.bar   
## 0.03 0.03 0.03   
## hygiene.articles berries sugar   
## 0.03 0.02 0.02   
##   
##   
## [[3]]  
## [[3]]$In\_Basket  
## [1] "rolls.buns" "soda" "bottled.water"   
## [4] "pip.fruit" "specialty.chocolate"  
##   
## [[3]]$Promote  
## whole.milk other.vegetables sausage   
## 0.20 0.18 0.17   
## bottled.beer shopping.bags canned.beer   
## 0.14 0.13 0.11   
## fruit.vegetable.juice frankfurter yogurt   
## 0.10 0.10 0.10   
## UHT.milk pastry chocolate   
## 0.08 0.07 0.07   
## root.vegetables newspapers tropical.fruit   
## 0.07 0.07 0.06   
## coffee brown.bread citrus.fruit   
## 0.05 0.05 0.05   
## napkins pork margarine   
## 0.04 0.04 0.04   
## frozen.vegetables domestic.eggs whipped.sour.cream   
## 0.04 0.04 0.03   
## waffles salty.snack long.life.bakery.product   
## 0.03 0.03 0.03   
## white.bread dessert beef   
## 0.03 0.03 0.03   
## butter chicken curd   
## 0.03 0.03 0.03   
## cream.cheese. misc..beverages beverages   
## 0.03 0.03 0.03   
## specialty.bar hygiene.articles berries   
## 0.03 0.03 0.02   
## candy frozen.meals sugar   
## 0.02 0.02 0.02   
## oil ham butter.milk   
## 0.02 0.02 0.02   
## hamburger.meat onions meat   
## 0.02 0.02 0.02   
##   
##   
## [[4]]  
## [[4]]$In\_Basket  
## [1] "other.vegetables" "newspapers" "white.bread"   
##   
## [[4]]$Promote  
## whole.milk root.vegetables rolls.buns   
## 0.34 0.23 0.18   
## yogurt soda tropical.fruit   
## 0.18 0.17 0.14   
## ham citrus.fruit whipped.sour.cream   
## 0.13 0.12 0.12   
## pork bottled.water frozen.vegetables   
## 0.10 0.09 0.09   
## margarine shopping.bags domestic.eggs   
## 0.09 0.08 0.08   
## chicken pip.fruit sausage   
## 0.08 0.08 0.08   
## bottled.beer beef pastry   
## 0.07 0.07 0.07   
## butter hamburger.meat onions   
## 0.06 0.06 0.06   
## cream.cheese. fruit.vegetable.juice coffee   
## 0.06 0.05 0.05   
## brown.bread canned.beer meat   
## 0.05 0.05 0.05   
## frankfurter napkins oil   
## 0.04 0.04 0.04   
## butter.milk chocolate UHT.milk   
## 0.04 0.04 0.03   
## waffles salty.snack long.life.bakery.product   
## 0.03 0.03 0.03   
## dessert specialty.chocolate curd   
## 0.03 0.03 0.03   
## misc..beverages beverages specialty.bar   
## 0.03 0.03 0.03   
## hygiene.articles berries candy   
## 0.03 0.02 0.02   
## frozen.meals sugar   
## 0.02 0.02   
##   
##   
## [[5]]  
## [[5]]$In\_Basket  
## [1] "whole.milk" "bottled.water" "bottled.beer"   
##   
## [[5]]$Promote  
## yogurt rolls.buns other.vegetables   
## 0.19 0.18 0.18   
## soda pastry tropical.fruit   
## 0.17 0.13 0.12   
## newspapers fruit.vegetable.juice brown.bread   
## 0.11 0.10 0.10   
## domestic.eggs whipped.sour.cream shopping.bags   
## 0.10 0.09 0.08   
## pip.fruit margarine butter   
## 0.08 0.08 0.08   
## sausage curd napkins   
## 0.08 0.07 0.07   
## root.vegetables white.bread sugar   
## 0.07 0.06 0.05   
## coffee citrus.fruit canned.beer   
## 0.05 0.05 0.05   
## hamburger.meat frankfurter chocolate   
## 0.05 0.04 0.04   
## pork frozen.vegetables waffles   
## 0.04 0.04 0.03   
## salty.snack long.life.bakery.product UHT.milk   
## 0.03 0.03 0.03   
## dessert beef specialty.chocolate   
## 0.03 0.03 0.03   
## chicken cream.cheese. misc..beverages   
## 0.03 0.03 0.03   
## beverages specialty.bar hygiene.articles   
## 0.03 0.03 0.03   
## berries candy frozen.meals   
## 0.02 0.02 0.02   
## oil ham butter.milk   
## 0.02 0.02 0.02   
## onions meat   
## 0.02 0.02   
##   
##   
## [[6]]  
## [[6]]$In\_Basket  
## [1] "whole.milk" "dessert"   
##   
## [[6]]$Promote  
## yogurt rolls.buns other.vegetables   
## 0.19 0.18 0.18   
## soda pastry tropical.fruit   
## 0.17 0.13 0.12   
## newspapers brown.bread domestic.eggs   
## 0.11 0.10 0.10   
## bottled.water whipped.sour.cream shopping.bags   
## 0.09 0.09 0.08   
## pip.fruit margarine butter   
## 0.08 0.08 0.08   
## sausage bottled.beer curd   
## 0.08 0.07 0.07   
## napkins root.vegetables white.bread   
## 0.07 0.07 0.06   
## sugar fruit.vegetable.juice coffee   
## 0.05 0.05 0.05   
## citrus.fruit hamburger.meat frankfurter   
## 0.05 0.05 0.04   
## chocolate pork frozen.vegetables   
## 0.04 0.04 0.04   
## waffles salty.snack canned.beer   
## 0.03 0.03 0.03   
## long.life.bakery.product beef specialty.chocolate   
## 0.03 0.03 0.03   
## chicken cream.cheese. misc..beverages   
## 0.03 0.03 0.03   
## beverages specialty.bar hygiene.articles   
## 0.03 0.03 0.03   
## berries candy frozen.meals   
## 0.02 0.02 0.02   
## oil ham butter.milk   
## 0.02 0.02 0.02   
## onions meat UHT.milk   
## 0.02 0.02 0.01