Preparing data for bnlearn models

Aaron Smith

6/24/2020

Let’s use bnlearn::preprocess() to discretize some data, then build Bayesian networks to compare the data preparation techniques.

The bnlearn::preprocess() function gives us three ways to discretize numeric data.

* interval: constant subinterval width
* quantile: all subintervals have (approximately) the same number of observations
* Hartemink: use mutual information to determine cutoffs

# Load the data

We will use the gaussian.test data from bnlearn.

options(digits = 1)  
require(bnlearn)

## Loading required package: bnlearn

##   
## Attaching package: 'bnlearn'

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

data(gaussian.test)  
head(gaussian.test)

## A B C D E F G  
## 1 1.1 1.9 7 9 0.9 25 9  
## 2 -0.2 11.3 24 23 7.0 37 4  
## 3 1.9 3.0 11 11 3.8 22 2  
## 4 0.8 3.9 11 12 1.0 23 6  
## 5 0.5 4.5 10 13 4.1 25 5  
## 6 1.6 0.9 7 7 6.5 28 8

summary(gaussian.test)

## A B C D E   
## Min. :-2 Min. :-10 Min. :-16 Min. :-9 Min. :-4   
## 1st Qu.: 0 1st Qu.: 0 1st Qu.: 4 1st Qu.: 6 1st Qu.: 2   
## Median : 1 Median : 2 Median : 8 Median : 9 Median : 4   
## Mean : 1 Mean : 2 Mean : 8 Mean : 9 Mean : 3   
## 3rd Qu.: 2 3rd Qu.: 4 3rd Qu.: 12 3rd Qu.:12 3rd Qu.: 5   
## Max. : 5 Max. : 14 Max. : 32 Max. :27 Max. :11   
## F G   
## Min. :-1 Min. :-1   
## 1st Qu.:18 1st Qu.: 4   
## Median :22 Median : 5   
## Mean :22 Mean : 5   
## 3rd Qu.:26 3rd Qu.: 6   
## Max. :46 Max. :12

# Discretize the data

Let’s use the three discretize methods on the data set, and store the prepared data in a list.

M <- gaussian.test  
  
list\_M <- lapply(  
 X = c("interval","quantile","hartemink"),  
 FUN = function(method) discretize(  
 data = M,  
 method = method,  
 breaks = 4,  
 ordered = TRUE  
 )  
)  
names(list\_M) <- c("interval","quantile","hartemink")  
lapply(X = list\_M,FUN = summary)

## $interval  
## A B C   
## [-2.25,-0.473]: 332 [-10.1,-4.02]: 114 [-15.9,-3.75]: 156   
## (-0.473,1.3] :2773 (-4.02,2.06] :2428 (-3.75,8.32] :2403   
## (1.3,3.07] :1791 (2.06,8.14] :2353 (8.32,20.4] :2311   
## (3.07,4.85] : 104 (8.14,14.2] : 105 (20.4,32.5] : 130   
## D E F   
## [-9.08,-0.0385]: 117 [-3.57,0.205]: 225 [-1.22,10.6]: 157   
## (-0.0385,8.97] :2375 (0.205,3.97] :2749 (10.6,22.3] :2441   
## (8.97,18] :2385 (3.97,7.73] :1947 (22.3,34.1] :2265   
## (18,27] : 123 (7.73,11.5] : 79 (34.1,45.9] : 137   
## G   
## [-1.38,2.08]: 362   
## (2.08,5.52] :2653   
## (5.52,8.97] :1853   
## (8.97,12.4] : 132   
##   
## $quantile  
## A B C   
## [-2.25,0.324]:1250 [-10.1,-0.0175]:1250 [-15.8,3.72]:1250   
## (0.324,0.984]:1250 (-0.0175,2] :1250 (3.72,8.06] :1250   
## (0.984,1.69] :1250 (2,4.08] :1250 (8.06,12.4] :1250   
## (1.69,4.85] :1250 (4.08,14.2] :1250 (12.4,32.4] :1250   
## D E F G   
## [-9.04,5.98]:1250 [-3.56,2.1]:1250 [-1.17,17.9]:1250 [-1.37,3.74]:1250   
## (5.98,8.99] :1250 (2.1,3.51] :1250 (17.9,22] :1250 (3.74,5.03] :1250   
## (8.99,12.2] :1250 (3.51,4.87]:1250 (22,26.3] :1250 (5.03,6.34] :1250   
## (12.2,27] :1250 (4.87,11.5]:1250 (26.3,45.8] :1250 (6.34,12.4] :1250   
##   
## $hartemink  
## A B C   
## (-2.25,-0.864]: 150 (-10.1,0.595]:1550 (-15.8,3.09]:1100   
## (-0.864,0.937]:2250 (0.595,2.62] :1350 (3.09,7.71] :1300   
## (0.937,2.57] :2300 (2.62,4] : 800 (7.71,11.4] :1100   
## (2.57,4.85] : 300 (4,14.2] :1300 (11.4,32.4] :1500   
## D E F G   
## (-9.04,6.96]:1600 (-3.56,1.59]: 850 (-1.17,19] :1550 (-1.37,3.68]:1200   
## (6.96,9.43] :1100 (1.59,3.25] :1400 (19,24] :1550 (3.68,4.82] :1050   
## (9.43,12.2] :1050 (3.25,5.17] :1750 (24,27.6] : 950 (4.82,6.85] :1900   
## (12.2,27] :1250 (5.17,11.5] :1000 (27.6,45.8]: 950 (6.85,12.4] : 850

# Build the Bayesian networks

Now let’s build the models. Some of the algorithms generate partially ordered networks. To make the models more consistent, let’s use bnlearn::choose.direction() to order all graphs.

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

for(j in v\_algorithms) for(k in names(list\_M)) try({  
 list\_bnlearn[[j]][[k]] <- do.call(  
 what = j,  
 args = list(x = list\_M[[k]])  
 )  
 M\_arcs <- arcs(list\_bnlearn[[j]][[k]])  
 for(l in 1:nrow(M\_arcs)){  
 list\_bnlearn[[j]][[k]] <- set.arc(  
 x = list\_bnlearn[[j]][[k]],  
 from = M\_arcs[l,1],  
 to = M\_arcs[l,2],  
 check.cycles = FALSE,  
 check.illegal = FALSE  
 )  
 list\_bnlearn[[j]][[k]] <- choose.direction(  
 x = list\_bnlearn[[j]][[k]],  
 arc = M\_arcs[l,],  
 data = list\_M[[k]]  
 )  
 }  
},silent = TRUE)

# Scoring the networks

Now let’s use Bayesian information criterion to evaluate each model.

M\_score <- matrix(  
 data = NA,  
 nrow = length(v\_algorithms),  
 ncol = length(list\_M),  
)  
rownames(M\_score) <- v\_algorithms  
colnames(M\_score) <- names(list\_M)  
  
for(j in v\_algorithms) for(k in names(list\_M)) try({  
 M\_score[j,k] <- score(  
 x = list\_bnlearn[[j]][[k]],  
 data = list\_M[[k]],  
 type = "bic"  
 )  
})  
for(j in rownames(M\_score)) M\_score <- M\_score[,order(M\_score[j,])]  
for(j in colnames(M\_score)) M\_score <- M\_score[order(M\_score[,j]),]  
M\_score

## quantile hartemink interval  
## gs -48188 -45815 -37062  
## iamb -48188 -45815 -37062  
## inter.iamb -48188 -45815 -37062  
## iamb.fdr -48188 -45815 -37062  
## pc.stable -47549 -45345 -37062  
## aracne -40560 -37934 -26924  
## chow.liu -40560 -37934 -26924  
## mmpc -38420 -36079 -25489  
## hpc -37138 -34570 -25489  
## fast.iamb -37448 -34896 -25347  
## si.hiton.pc -37246 -34676 -25317  
## h2pc -37113 -34561 -24705  
## hc -35493 -33210 -24705  
## tabu -35493 -33210 -24705  
## rsmax2 -35493 -33210 -24705  
## mmhc -35493 -33210 -24705

graphviz.plot(  
 list\_bnlearn[[nrow(M\_score)]][[ncol(M\_score)]]  
)

## Loading required namespace: Rgraphviz

