Implementation of a Graphical Model search algorithm in R - retake

João Paulo Pizani Flor, Tomaš Kadlec
Department of Information and Computing Sciences,
Utrecht University - The Netherlands
e-mail: {j.p.pizaniflor, t.kadlec}@students.uu.nl

Thursday 2nd January, 2014

1 Introduction

This report describes the solution to the second practical assignment of the master course "Data Mining" at Utrecht University, taught in the 1st period of the academic year 2013/2014.

2 Organization of the distributed package

The handed-in solution is organized as follows:

R workspace The R workspace image containing both the *code* and the *datasets* used for analysis is under the package root directory with name "`RData.RData`"

R Source The source code of the functions implemented is located in the package root with the name "`graphModel.R`". In a running R session, the code can be loaded by issuing the command `source("graphModel.R")`.

Datasets The datasets available for usage are available under the "`datasets`" subdirectory, in table and CSV format.

3 Remarks and improvements

Error corrected in the retake period A small error was causing all the problems in the first version of the handed-in code, and it was corrected. Namely, the *dimension* of a model was being calculated incorrectly, therefore also the score. Now we calculate the dimension of a model (number of *unique* parameters) as the number of cells in the contingency table (parameters of the *saturated model*) **minus** the degrees of freedom (value `df` returned by the `loglin` function).

Remarks from the original work (regular period) We improved the "standard" solution to the assignment in the following ways:

1. We use our own clique-finding function. It is based on the Bron-Kerbosch algorithm, but we had written the code before the version by the lecturer was released. Our implementation is based on recursion.

2. We implemented *Iterated Local Search* (ILS). Apart from the requested "restarts" function (usually called *Repeated* or *Multistart* Local Search), we implemented a function which performs *hill climbing on the space of local minima*. To implement this, we define two *perturbation* functions: one small and one large. The *small perturbation* is used for the "inner" local search, to achieve a local minimum. This function only adds or removes an edge in a graph i.e. generates an immediate neighbor in the search space of all possible graphs. In our code this function is called *gm.perturbateEdge*. Then, in the ILS function, given a local minimum, we perturbate it using the *large perturbation*, find "neighbours" and then start a local search for each of these neighbours. The large perturbation function in our code is called *gm.perturbateForILS*. It takes as a parameters the *number of edges* to be perturbated (defaults to 3) and chooses n edges at random in the graph to perturbate.

4 Data Analysis

This section contains the answers to part 2 of the assignment, which involved using the implemented algorithm to perform the analysis of a dataset about cancer treatment and mortality.

The dataset contains data about 5.735 critically ill patients receiving care in an Intensive Care Unit for 1 of 9 disease categories. The goal of the study was to examine the association between use of right heart catheterization (RHC) in the first 24 hours of treatment and subsequent survival.

The analyzed dataset consists of 10 categorical variables, distributed as follows:

cat1 Disease category (9 possible values)

death Did the patient die within 180 days

swang1 Was RHC performed in the first 24 hours

gender Male/Female

race Black/White/Other

ninsclas Type of medical insurance (6 possible values)

income Patient income (4 possible values)

ca Cancer status (yes/no/metastatic)

age Patient age (5 possible values)

meanbp1 Mean blood pressure (2 possible values)

4.1 Answers to the data analysis questions

Question a: How many different graphical models are there for this dataset?

R: For n vertices, there are $2^{n(n-1)/2}$ possible simple graphs with those vertices. Therefore, considering that the RHC dataset has 10 variables, graphical models for it will have 10 vertices, and there are $2^{10(10-1)/2} = 2^{45}$ possible graphical models with 10 vertices.

Question b: How many cells does the table of counts for this data set have? How many parameters does the saturated model have?

R: The dataset has 10 variables, with respectively 9, 2, 2, 2, 3, 6, 4, 3, 5 and 2 possible values each. Therefore, the contingency table has $9*2*2*2*3*6*4*3*5*2 = 155520$ cells. The saturated model has one parameter for each cell in the contingency table, therefore also 155520 parameters.

Question c: Perform a forward-backward search on this data set, starting from the empty graph (independence model). Use the BIC scoring function. List the cliques of the resulting model and draw the independence graph.

R: The search resulted in a model with a score of 15841.66, with the following cliques: $\{1, 3, 10\}$, $\{1, 8\}$, $\{1, 9\}$, $\{2, 8\}$, $\{2, 9\}$, $\{2, 10\}$, $\{3, 6\}$, $\{4, 6\}$, $\{5, 6\}$, $\{5, 10\}$, $\{6, 7\}$, $\{6, 8\}$, $\{6, 9\}$.

The resulting graph is drawn below on figure 4.1.

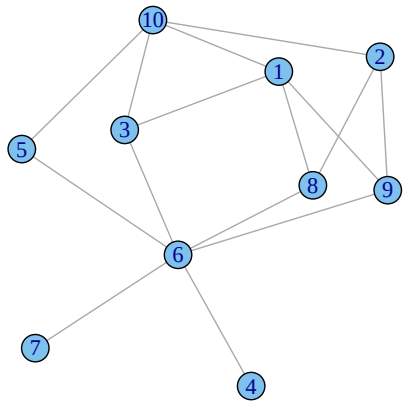

Figure 1: Model resulting from starting with the empty graph and using BIC score

Question d: Using the independence graph you found under question c, what can you say about the pairwise conditional independence between income and gender? If we are interested in predicting whether or not someone survives, looking at the independence graph, which variables appear to be sufficient for that purpose?

R: According to the graph found in question (c), the variables `income` and `gender` are independent *given* the type of insurance. If we are interested in predicting the `death` variable, the local Markov property tell us that we only need to know the values of variable in the boundary of the `death` variable. That is, to predict death we need to know the values of variables 10 (`mean blood pressure`), 8 (`cancer status`) and 9 (`age`).

Question e: Perform a forward-backward search on this data set, starting from the complete graph (saturated model). Use the BIC scoring function. List the cliques of the

resulting model and draw the independence graph. How does it compare to the model you found under (c)?

R: These search parameters returned a model with score 15850.53, and the cliques:
$\{1,3,10\}$, $\{1,8\}$, $\{2,7\}$, $\{2,8\}$, $\{2,9\}$, $\{2,10\}$, $\{3,4\}$, $\{3,9\}$, $\{4,6\}$, $\{5,7\}$, $\{5,9\}$, $\{5,10\}$, $\{6,7\}$, $\{6,8\}$, $\{6,9\}$.
The graph of this model is drawn below on figure 4.1.

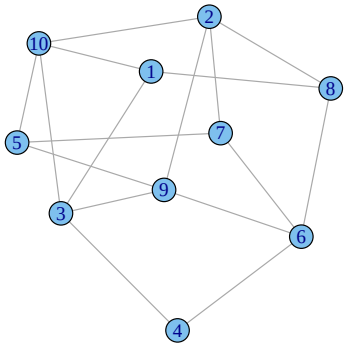

Figure 2: Graph of the model resulting from starting with the complete graph and using BIC score

Question f: Perform a forward-backward search with AIC scoring on this data set, starting from the complete graph and empty graph. Give the cliques of the models you find, and the score of the models. Are they the same? Compare the complexity of the models you found with BIC to those you found with AIC. Can you explain the difference?

R: Both searches using AIC scoring resulted in the same model with score 14278.21 and containing following 14 cliques: $\{1,2,8\}$, $\{1,2,9\}$, $\{1,2,10\}$, $\{1,3,9\}$, $\{1,3,10\}$, $\{1,4,8\}$, $\{1,4,10\}$, $\{2,7\}$, $\{3,6,9\}$, $\{4,5,6\}$, $\{4,5,10\}$, $\{4,6,8\}$, $\{5,6,7\}$, $\{5,6,9\}$.
Graph of this model is shown on figure 4.1.
The difference between AIC and BIC lies in the sizes of contained cliques. The models obtained with BIC scoring have majority of cliques of size 2 and only one of size 3 (staring on empty graph) or two (when complete graph was the starting point) whereas when AIC scoring was used the ratio was opposite - it contained one *1*-clique and 13 *3*-cliques This can be explained by the fact that the BIC scoring function gives a larger *penalty* for complex models. The factor for the penalty in BIC is $\ln n$ (where n is the number of observations), while for AIC it is the constant 2. In this dataset, $n = 5735$, and $\ln 5735 > 2$.

Question g: Use random restarts to see whether you can find better scoring models than you have found so far (both for AIC and for BIC scoring). Report your findings.

R: We have performed 10 searches from distinct starting graphs in the searched space with both scoring schemes (AIC and BIC). We chose equal probability for an edge to exists

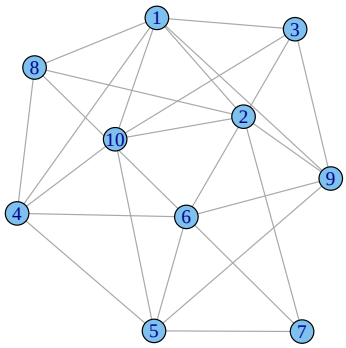

Figure 3: Graph of the model obrained with AIC scoring starting from both empty and complete graphs

between any given pair of vertices in an initial graph i.e. *prob* parameter was 0.5. Interestingly enough both runs have provided models with slightly higher score than models which were found starting on complete or empty graphs. The returned model has score 14347.75 using AIC and 16000.28 using BIC scoring. The model obtained with AIC scoring contains following cliques:

$\{\{1,2,8\},\{1,2,10\},\{1,3,10\},\{1,4,6\},\{1,4,8\},\{1,4,10\},\{2,7,8\},\{2,8,9\},\{3,7\},\{3,9\},\{4,6,9\},\{4,8,9\},\{$

The graph of this model is shown below on figure 4.1.

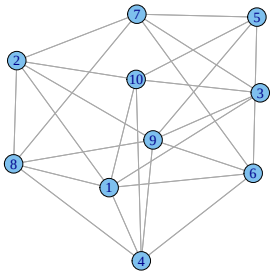

Figure 4: Model found by random restarts with AIC scoring

The model found with BIC scoring and random restarts contains 18 cliques:

$\{\{1,8\},\{1,9\},\{1,10\},\{2,7\},\{2,8\},\{2,9\},\{2,10\},\{3,4\},\{3,8\},\{3,9\},\{3,10\},\{4,6\},\{5,7\},\{5,9\},\{5,10\}\{$

This model is shown on figure 4.1.

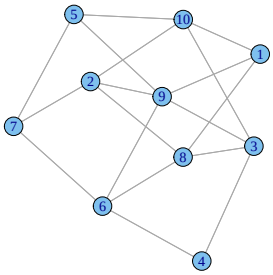

Figure 5: Model found by random restarts with BIC scoring

Tests showed that structure of the resulting model is heavily dependent on the initial parameter *prob*. With lower value the algorithm starts with sparser initial graphs and the resulting model was also sparse in comparision with results obtained with higher values of *prob*. Unfortunately searches starting on denser graphs tend to be very computationally demanding which have preventend us from using higher number of random restarts.