Data Mining 2012
Assignment 2: Graphical Models

Instructions

This assignment should be made in groups of two or three students. Solutions should be handed in ultimately Friday, November 2. Send your solution by e-mail to `ad@cs.uu.nl`. Your solution consists of

1. An R workspace containing the functions you have written.
2. A flat ascii file containing the documented program code, and
3. A PDF file containing the answers to Part 2.

Always put name and student number on your work.

Part 1: Programming

Write a function to learn an undirected graphical model from data. To fit a single model, use the function `loglin` for fitting hierarchical log-linear models. You have to pass as arguments to `loglin` a table with the observed counts, and a list containing the margins to be fitted. The margins to be fitted can simply be indicated by giving the appropriate dimension numbers, for example to indicate that the margin containing the first, third and sixth variable should be fitted, simply include the vector `c(1,3,6)` in the list of margins that you pass to `loglin`. You don't have to use variable names or value names. Note that in graphical models, the margins to be fitted correspond to the cliques in the graph. Read the documentation of `loglin` for the relevant details.

The function to be written should be named `gm.search`. The first argument is the table of observed counts (to be passed to `loglin`), and the second argument is the initial graph from which the search process starts. In the call to `gm.search` the initial graph is represented as a matrix called `graph.init`, where `graph.init[i,j]=1` if there is an edge between node i and node j, and `graph.init[i,j]=0` otherwise. Obviously, this matrix is symmetric.

The algorithm will perform a local search starting at `graph.init`, where the neighbors of the current model are those models that can be obtained by

1. adding an edge to the current model, or

2. removing an edge from the current model.

The third argument of `gm.search` is the boolean `forward` indicating whether adding edges is allowed, and the fourth argument is the boolean `backward` indicating whether removing edges is allowed.

Models are scored by AIC or BIC (the user should have a choice), where

$$\mathrm{BIC}(M) = \mathrm{dev}(M) + \ln N \times \dim(M).$$

The fifth argument is the string `score` which has the value "aic" if AIC scoring is wanted, and "bic" if BIC scoring is wanted.

The function `gm.search` should return a list containing the following named components:

1. `$model`: A list containing the cliques of the final model. Each clique is a vector containing the numbers of the variables in the clique.

2. `$score`: The AIC or BIC score of the final model.

3. `$call`: the call to the function `gm.search` that produced this result. Use the function `match.call` for this.

Here's an example call and the correct result. I included some print statements in the function to provide some information about the search process. The file `coronary.txt` can be downloaded from the course web page.

```
# Read in the data (assuming you saved it in "U:/Data Mining/coronary.txt").

> coronary.dat <- read.table("U:/Data Mining/coronary.txt",header=T)

# Perform a forward-backward search starting from the empty graph.
# Use BIC to score the models.

> coronary.graphmod <- gm.search(table(coronary.dat),matrix(0,6,6),
                                 forward=T,backward=T,score="bic")
Added: 2 - 3 (score= 457.58 )
Added: 2 - 5 (score= 415.52 )
Added: 1 - 2 (score= 380.46 )
Added: 1 - 5 (score= 351.56 )
Added: 1 - 3 (score= 331.61 )
Added: 4 - 5 (score= 326.32 )
Added: 2 - 6 (score= 321.71 )
Added: 1 - 4 (score= 320.2 )
Added: 2 - 4 (score= 315.15 )
```

```
Removed: 4 - 5 (score= 311.79 )

# Print the resulting object.

> coronary.graphmod
$cliques
$cliques[[1]]
[1] 1 2 3

$cliques[[2]]
[1] 1 2 4

$cliques[[3]]
[1] 1 2 5

$cliques[[4]]
[1] 2 6


$score
[1] 311.7889

$call
gm.search(data = table(coronary.dat), graph.init = matrix(0,6,6),
    forward = T, backward = T, score = "bic")
```

A simple hill-climbing search can get stuck in (bad) local minima. A straightforward way to try to mitigate this problem is to use random restarts from different initial models and keeping the best solution found. Write a function `gm.restart` with arguments `nstart` (the number of restarts to be performed) and `prob`, which specifies the probability of an edge between any pair of nodes (to be used in generating the random graph). The third argument `seed` allows the user to specify a random seed so the results can be reproduced. The random seed can be set with the function `set.seed`.

Part 2: Data Analysis

In the second part you are going to analyze a data set with the function you have written. The data set contains data from a medical study on potential predictors for the completion of a vaccine regimen (see the course web page for the data set and documentation). Before you start the analysis, remove the first column `Age` (the second column contains a discretized version of this attribute), and the fourth column `Shots` (contains almost the same information as `Completed`). This should give you a data set with 8 columns and 1413 rows.

Give answers to the following questions in your report:

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

(b) Using the independence graph you found under (a), what can you say about the pairwise conditional independence between insurance type and whether or not someone completes the vaccine regimen? If we are interested in predicting whether or not someone completes the vaccine regimen, which variables appear to be sufficient for that purpose?

(c) Test the model you found under (a) against the saturated model. Use $\alpha = 0.05$. What is the conclusion of the test? (note that this is not a statistical test in the traditional sense, since we used the data to generate the "hypothesis").

(d) Perform a forward-backward search on this data set, starting from the empty graph. Use the AIC scoring function. List the cliques of the resulting model and draw the independence graph. How does it compare to the model you found under (a)?

(e) Perform a forward-backward search on this data set, starting from the complete graph (saturated model). Use the AIC scoring function. Does it produce a better-scoring model than the one you found under (d)?

(f) 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.

Handing in the assignment

The R workspace (file with extension `.Rdata`) that you hand in should be tested to work on the Windows machines in the computer labs of the University. This file will be used to test the functions you have written.

Also put the functions you have written in a flat ascii file. Before you give the code of a function, provide the following information (start each line containing this information with the symbol #):

1. Name of the function.
2. Names and types of input arguments.
3. The result returned by the function.
4. A short description of what it does.

The two main functions should be called `gm.search` and `gm.restart`, and should be the first two functions in the file you send me. Any other functions you have written that are needed should be listed below that. Your report on the analysis of the data set (containing the answers to part II) should be handed in in PDF format.

Remarks

- As we mentioned, in graphical models, the margins to be fitted correspond to the cliques in the graph. Hence, you need to implement an algorithm for finding the cliques of a graph. The Bron-Kerbosch algorithm with pivot selection is quite easy to implement for example.
- Note that we cannot handle data sets with too many variables, because the `table` function will start to complain that we try to create a table with too many cells. Since the data has to be passed to `loglin` as a table, we cannot avoid this problem, unless we write our own IPF implementation.

Grading

The following considerations are taken into account to determine the grade for this assignment:

- Does the program work, and does it return the correct result?
- Efficiency and elegance of the implementation.
- Quality of the report.