Multidimensional Bayesian regression in R

Fraser I. Lewis

Abstract

This vignette introduces the **abn** package of R for identifying and exploring dependencies in multi-dimensional data through the application of additive Bayesian network models. Among the models currently implemented is a multidimensional analogue of logistic regression. Laplace approximations are used to estimate marginal likelihoods for model selection and also to compute marginal posterior densities.

Keywords: R, Bayesian Networks, additive models, structure discovery.

1. Introduction

Bayesian network (BN) modeling (**? ?**; **? ?**; **? ?**; **? ?**) is a form of graphical modeling which attempts to separate out indirect from direct association in complex multivariate data, a process typically referred to as structure discovery (**? ?**). Unlike other widely used multivariate approaches where dimensionality is reduced through exploiting linear combinations of random variables, such as in principal component analysis, graphical modeling does not involve any such dimension reduction. Bayesian networks have been developed for analyzing multinomial, multivariate Gaussian or conditionally Gaussian networks (a mix categorical and Gaussian variables). A number of libraries for fitting such BNs are available from CRAN. These types of BN have been constructed to ensure conjugacy, that is, enable posterior distributions for the model parameters and marginal likelihood to be calculated analytically. The purpose of **abn** is to provide a library of functions for more flexible BNs which do not rely on conjugacy, which opens up an extremely rich modeling framework but at some considerable additional computational cost.

This first release of **abn** includes functionality for fitting non-conjugate BN models which are multi-dimensional analogues of logistic regression, with the caveat that all the variables in the data are binary. Also included are traditional conjugate BN models for multinomial data and a range of heuristic search options for determining locally optimal models.

The general objective in BN modeling/structure discovery is to perform a model search on the data to identify a locally optimal model, recall that BN models have a vast search space and it is generally impossible to determine a globally optimal model. We consider two different approaches for identifying a "best" locally optimal BN model: firstly by conducting a series of heuristic searches and then selecting the best model found (**? ?**); and secondly by building a summary network using results across many different searches (**? ?**; **? ?**). There are obvious pros and cons to either approach and both are common in the literature and provide a good first exploration of the data. For a general non-technical review of BN modeling applied in biology see **? ?**. A case study in applying BN models to epidemiological data using the

conjugate BN functionality in **abn** can be found in **? ?**.

2. Searching and model fitting

The package comes with several simulated data sets for use with the examples in the help files. The data set we use here comprises of ten random variables where each of these is a factor and is binary.

```
> library(abn)
> data(sim10varadd);
> head(sim10varadd);
```

```
  X1 X2 X3 X4 X5 X6 X7 X8 X9 X10
1  0  1  1  1  1  0  1  0  1   1
2  1  1  1  0  0  1  1  1  1   1
3  0  0  1  0  0  1  1  1  1   1
4  0  0  0  0  0  0  0  1  1   1
5  0  1  0  1  0  1  1  1  0   1
6  0  0  1  0  0  0  1  1  1   1
```

We now conduct a single heuristic search using a stepwise hill-climbing algorithm where the search commences from random initial network model. To avoid excessive computation on overly complex models we impose a limit on the maximum number of parents (arcs go from parent to child) which each node can have, this should be large enough so that it does not affect the model search but small enough to sensibly limit the model search space. It is also possible to further restrict the search space by forbidding certain arcs between variables, this is given as a matrix with one row for each child node and the columns are parents. This must be provided as a numeric matrix where a 1 is for the arc to be banned - the search will not consider any models containing this arc. In the following analyzes we do not ban any arcs.

```
> banned<-matrix(     c(0,0,0,0,0,0,0,0,0,0, ##
+                         0,0,0,0,0,0,0,0,0,0, ##
+                         0,0,0,0,0,0,0,0,0,0, ##
+                         0,0,0,0,0,0,0,0,0,0, ##
+                         0,0,0,0,0,0,0,0,0,0, ##
+                         0,0,0,0,0,0,0,0,0,0, ##
+                         0,0,0,0,0,0,0,0,0,0, ##
+                         0,0,0,0,0,0,0,0,0,0, ##
+                         0,0,0,0,0,0,0,0,0,0, ##
+                         0,0,0,0,0,0,0,0,0,0  ##
+                         ),byrow=TRUE,ncol=10);# 10 x 10 matrix
> rownames(banned)<-names(sim10varadd);# must have rownames set
> colnames(banned)<-names(sim10varadd);# must have colnames set
> set.seed(10000);# only random part is the initial choice of network
> myres<-searchabn(data.df=sim10varadd,banned.m=banned,
+                  hyper.params=list(
```

```
+                   mean=c(0,0,0,0,0,0,0,0,0,0,0),
+                   var=c(1000,1000,1000,1000,1000,
+                   1000,1000,1000,1000,1000,1000)),
+                   max.parents=3,init.permuts=10);

initial network: (log) network score = -6457.506054
search iteration...1 new score=-6442.953014
search iteration...2 new score=-6437.488063
search iteration...3 new score=-6432.029875
search iteration...4 new score=-6401.874728
search iteration...5 new score=-6396.717416
search iteration...6 new score=-6391.616027
search iteration...7 new score=-6386.598619
search iteration...8 new score=-6381.680833
search iteration...9 new score=-6376.853649
search iteration...10 new score=-6372.701646
search iteration...11 new score=-6369.532669
search iteration...12 new score=-6366.762068
search iteration...13 new score=-6366.762049
```

The above analyzes use the default parameter priors of diffuse Gaussian densities with mean zero and variance of 1000. These priors are fixed across all models in the search process and therefore it does not make sense for these to be given as informative during a search, but may be appropriate when fitting a single BN model e.g. using fitabn().

3. Multiple searches

To perform many searches then the following code is appropriate which repeats the above single search 1000 times (where this is implemented as a separate function at C rather than R level).

```
set.seed(10001);
myres.1<-hillsearchabn(data.df=sim10varadd,banned.m=banned,
                  hyper.params=list(
                  mean=c(0,0,0,0,0,0,0,0,0,0,0),
                  var=c(1000,1000,1000,1000,1000,
                  1000,1000,1000,1000,1000,1000)),
                  max.parents=3,init.permuts=10, num.searches=1000);
```

Multiple searches make take many hours to run and such a task is ideal for task farming over multiple cpus. In this case using R in batch mode is an obvious solution repeating the above code but using a different seed in each search. Using this approach the search results must then be recombined and any duplicates removed - since it is theoretically possibly for searches to start from the same randomly chosen network. This also applies if running all the searches within a single call to hillsearchabn() as this does not check for duplicate starting networks. Supposing *two* such separate "parallel" searches were conducted then the following code could

be used to combine the results as if they had been conducted in one call to hillsearchabn() and then also removing duplicates.

```
all.res<-list();
all.res$init.score<-c(myres.1$init.score,myres.2$init.score);
all.res$final.score<-c(myres.1$final.score,myres.2$final.score);
all.res$init.dag<-c(myres.1$init.dag,myres.2$init.dag);
all.res$final.dag<-c(myres.1$final.dag,myres.2$final.dag);

## now remove any duplicate searches
indexes<-uniquenets(all.res$init.dag);

## create a list of all results
all.res.f<-list();
all.res.f$init.score<-all.res$init.score[indexes];
all.res.f$final.score<-all.res$final.score[indexes];
all.res.f$init.dag.<-all.res$init.dag[indexes];
all.res.f$final.dag<-all.res$final.dag[indexes];
```

3.1. Summary and model visualization

After conducting many searches then the results can be summarized by either by choosing the best network found or else by computing a consensus network, where the latter is produced by building a network out of all the arcs which appear in at least a minimum proportion of locally optimal networks. To choose the best single network from the results of a call to hillsearchabn() and to write the network structure out to a file in an appropriate format for Graphviz then

```
best.scores<-which(all.res.f$final.score==max(all.res.f$final.score));
tographviz(all.res.f$final.dag[[best.scores[1]]],outfile="best1.dot");
```

To view the best single network as a matrix in the R console then simply type `all.res.f$final.dag[[best.scores[1]]]`, this matrix can then be used with `fitabn()` to fit the corresponding BN model to data.

Graphviz is probably the best known tool for plotting and visualizing networks and is available for free download (and now with an open source form of license, although not GPL) from www.graphviz.org. There is a graphviz library for R available via the bioconductor project which requires a separate graphviz installation.

To create a majority consensus network, which comprises of a network constructed of those arcs present in at least 50% of all locally optimal network, then

```
## see ?prunenets - a side effect is freq.dist of all arcs
mypruneddags<-prunenets(all.res.f$final.dag,
                                round(0.50*length(all.res.f$final.dag)));
## a convenience function for use with apply
myfunc<-function(arg1,threshold,netdata){
         if(arg1>=round(threshold*length(netdata$final.dag))){
```

```
            return(1);} else {return(0);}}

### build a consensus 50% network/matrix
con.50<-apply(mypruneddags$arcs.sum,c(1,2),
              FUN=myfunc,threshold=0.50,
              netdata=all.res.f);## make reading easier
tographviz(con.50,outfile="con50.dot");#format for reading into graphviz
```

4. Parameter estimation

We now demonstrate how to estimate the marginal parameters in an additive BN using the best network found in the single search at the start of Section 2. In this network we find that, for example, node `X7` has `X3` as a parent, `X5` has `X2` as a parent and `X6` has `X7` as a parent. Each of these corresponds to a local logistic regression with the parent being a "main effect" parameter in the classical additive glm sense. As usual we wish to estimate these parameters to see whether are statistically significantly different from zero, and also examine the magnitude of their effect on the log odds of the "response variable", e.g. `X7`, `X5` and `X6`. To estimate the marginal densities then the following is sufficient, which estimates the densities at 1000 equally spaced points between 0 and 2, which is sufficient for these parameters and were chosen after examining the densities over various ranges.

```
> mynetwork<-myres[[length(myres)]];#the best network is the last one
> x7.x3<-getmarginal(sim10varadd,mynetwork,whichnode="X7", whichvar="X3",
+             hyper.params=list(
+             mean=c(0,0,0,0,0,0,0,0,0,0),
+             var=c(1000,1000,1000,1000,1000,
+                   1000,1000,1000,1000,1000,1000)),
+                   post.x=seq(0,2,len=1000), verbose=FALSE);
> x5.x2<-getmarginal(sim10varadd,mynetwork,whichnode="X5", whichvar="X2",
+             hyper.params=list(
+             mean=c(0,0,0,0,0,0,0,0,0,0,0),
+             var=c(1000,1000,1000,1000,1000,
+                   1000,1000,1000,1000,1000,1000)),
+                   post.x=seq(0,2,len=1000), verbose=FALSE);
> x6.x7<-getmarginal(sim10varadd,mynetwork,whichnode="X6", whichvar="X7",
+             hyper.params=list(
+             mean=c(0,0,0,0,0,0,0,0,0,0,0),
+             var=c(1000,1000,1000,1000,1000,
+                   1000,1000,1000,1000,1000,1000)),
+                   post.x=seq(0,2,len=1000), verbose=FALSE);
>
```

Figure 1 shows the marginal posterior densities estimated using the Laplace approximation and with the same default diffuse Gaussian priors as used in the model search. The numerical values used to create each density are returned from `getmarginal()` as a two column numerical matrix with the first column the value of the variable (named `x`) and the second column the value of the density $f(x)$, (named `f`), see the `getmarginal()` help page for more details.

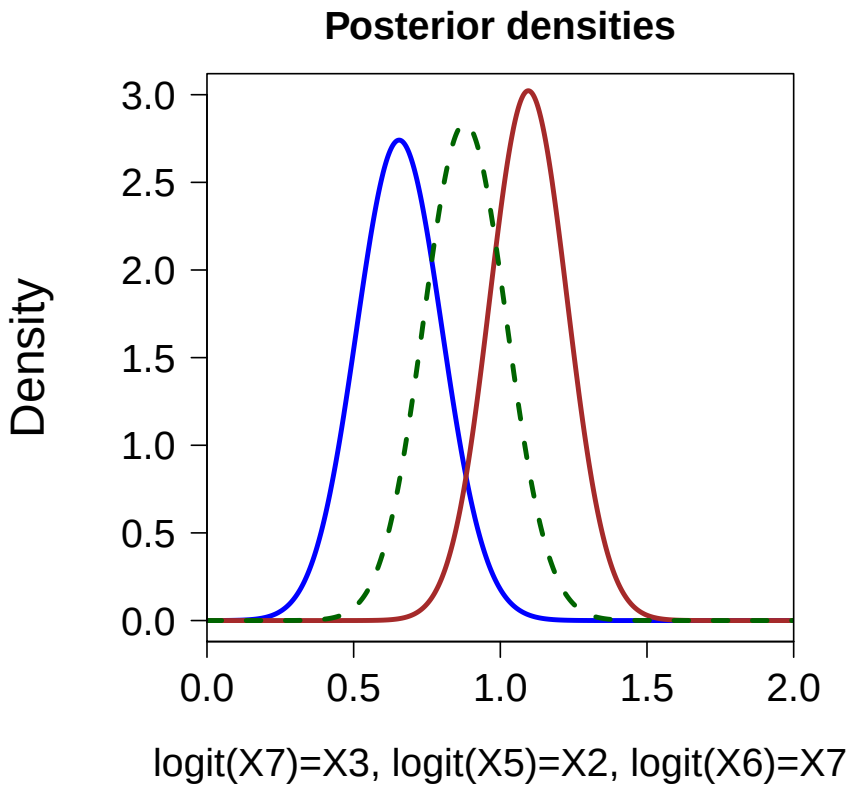

Figure 1: Marginal posterior distributions for parameters in selected nodes

5. Follow-up analyzes via MCMC

So far we have considered exploring the model space to find a good locally optimal BN. Such analyzes are ideal for initial exploration of multi-dimensional data to narrow down the possible structural features present. Estimation of the marginal posterior densities as given above is useful for checking whether the marginal likelihood has not been sufficiently parsimonious in the model selection (which can be particularly problematic with small data sets). Generally speaking, however, once a good model has been identified then more detailed modeling may be appropriate, for example by using other forms of priors on individual parameters or introducing other complexities such as random effects into the graphical model to account for grouping effects or other forms of correlation. These are not currently available within the modeling framework in **abn**, but it is generally straightforward to implement a model identified using the model search capabilities in **abn** into a graphical model within Gibbs sampler software such as `JAGS` or `WinBUGS`, which then allow almost arbitrary levels of additional complexity. In short, one approach to multivariate analyses is to use BN modeling for initial explorations through performing numerous model searches, and then once a good model has been identified then use MCMC to further refine this.

6. Other features and future directions

The package also includes model search and fitting routines for conjugate multinomial BN models, whose functions are analogous to those shown above (except without explicit marginal posterior density estimation). Similar models can be fitted using other R packages such as **deal** or **bnlearn**. In **abn** the model space for multinomial BNs can be similarly restricted as above by imposing a limit on the maximum number of parents, and also includes the K2 metric as well as the Bayesian Dirichlet metric, each of which use different parameter priors and therefore give different marginal likelihood estimates.

While the models shown above only include main effects (in the sense of an additive linear model) it is straightforward to include interaction terms by simply adding nodes to the model as additional columns in the model matrix, and as additional columns in the data.frame, so they are treated simply as additional variables.

Currently the only non-conjugate BN model implemented is the logistic BN model (note that details of a non-Bayesian equivalent are presented in **?** **?**), and only for binary data. This is arguably the simplest useful, non-conjugate additive BN model. It is also typically possible to discretize continuous or multinomial data into binary data, at least for exploratory purposes. It is hoped to extend the current functionality to include multinomial variables and then models comprising categorical and (Gaussian) continuous variables. In the longer term including more complex structures such that appropriate for ecological count data, e.g. Poisson and negative binomial distributions, and also models with random effects may be possible.

Affiliation:

F. I. Lewis
Vetsuisse Faculty
University of Zurich
Winterthurerstrasse 270
Zurich CH-8057
Switzerland
E-mail: fraseriain.lewis@uzh.ch