MCMCBNE package v1.0 manual

Sangkyu Lee
Ph.D. candidate
Medical Physics Unit, McGill University
Montreal,Quebec, Canada

June 9, 2015

1 Package highlights

Markov Chain Monte Carlo sampling for Bayesian Network Ensemble (MCMCBNE) is a Matlab package for learning/testing an ensemble of Bayesian Network (BN) graphs on multivariate data. The main motivation was to develop a BN-based classifier for predicting radiotherapy response using longitudinal biomarker measurements and radiotherapy plan/ other patient-specific information. However, the toolkit can be used for other classification problems after customizing a data import component. It was developed as an extension to the Bayes Net Toolbox (https://github.com/bayesnet/bnt) which provides backbone functions for Bayesian Network graph and parameter training. Major additional features implemented by the MCMCBNE include:

- Number of input variables can be reduced by two types of filtering schemes: Koller-Sahami/ L1 regulated logistic regression.
- MCMC graph sampling can be guided by the two types of user-defined priors: causality (reject the edges in non-causal direction) and biological (edges reported in literatures are given higher prior probability)
- Ensemble of Bayesian networks can be formed to be used for classification.
- Classification performance can be measured in a cross-validation/0.632+ bootstrap settings.
- Codes for graph learning and performance testing are parallelized for submission to high-computing clusters.

Organization of the package is shown in figure 1.

2 Dependencies

The following packages (all Matlab based) need to be installed beforehand:

- Bayes Net Toolbox (BNT) (https://github.com/bayesnet/bnt)
- Dose Response Explorer (DREES) (https://github.com/yw2026/DREES)
- Bioinformatics toolbox from MATLAB

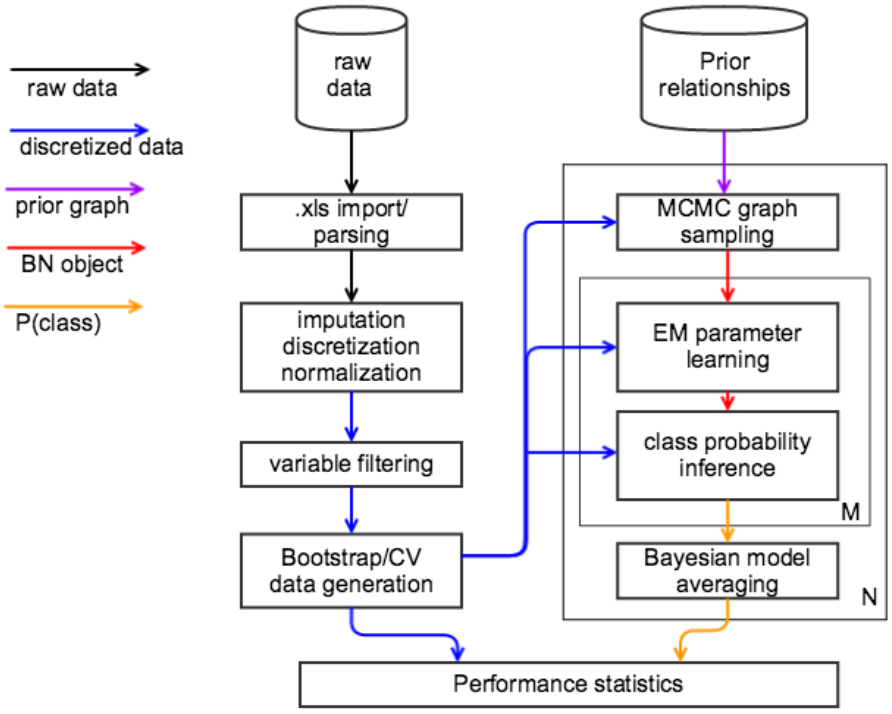

Figure 1: Schematic diagram of the modules constituting the MCMCBNE package. M: number of graphs in one ensemble, N: number of bootstrap/CV datasets

Table 1: Assignment of time point labels for reading biomarker data spreadsheet.

time label	conventional	hypo fraction
1	baseline	baseline
2	mid-treatment	end-treatment
3	end-treatment	3-month postRT
4	3-month postRT	6-month postRT
5	6-month postRT	

3 Data Import

The data importing module was written according the current convention of radiotherapy record keeping. For other uses, users are encouraged to write their own module that suits the structure of a specific dataset. In this case, jump to the section for a graph learning module.

The module reads biological and physical/clinical data from separate .xls files. The spreadsheet should be stored in the Excel 97 version with the first row indicating variable names. The physical/clinical data spreadsheet is organized in a rows-for-instances and columns-for-variables convention. The biological data sheet, however, has each row for the measurement from one patient at one time point. A row identifier is written in the following format: "PatientID-time point" (ex:L012-3). One-digit number from 1 to 5 is assigned to the following time point, which differs for 2 fractionation groups due to the current data collection protocol 1:

There are three pre-processing steps applied to the raw data imported from spreadsheets:

1. Imputation: missing entries are filled in, using one of the two options:
    - median: fill the missing value with the median of the existing data
    - k-nearest neighbor: Similarity between patients is evaluated using the variables with no NaN. The missing entry is filled by the value of the patient with the highest similarity.
2. Normalization
    - standard z-score: data is shifted/scaled to the zero mean and unitary standard deviation.

- min-max: data is linearly transformed so that the smallest and the largest entry hold the value of 0 and 1 respectively.
- softmax: similarly to above, data is squeezed to the [0 1] range, but a hyper tangent function is used instead of a linear one for the mapping. This reduces the effect of outliers or extreme values on the normalization result.

3. Discretization

   - maximum mutual information: bin boundary is optimized to maximize mutual information with respect to a class variable.
   - k-means: clustering-based unsupervised discretization option.

These options can be set in the file kyu_BN_readdata_combined.m.

The import module can be called in the Matlab command window as follows:

```
>> SBRTfilter = 2; % select 2 Gy per fraction pts only.
>> disc = 3; % use K-means discretization
>> [data,data_c,data_missing,data_c_missing,...
    labels,mi,studyid,FracSize,COMSI] = ...
    kyu_BN_readdata_combined(SBRTfilter,disc)
The following variables are found:
1. a2m_pre
2. a2m_intra
3. a2m_end
...
choose the variables, separated by comma: 1,4
```

Calling the function will lead to a user prompt first showing the choices for the variables detected in the raw data spreadsheet. Specify the variables that you want to include into a data matrix by a comma separated list of numbers as shown in the list.

The required input parameters are:

- SBRTfilter: determines which fractionation group to select.

  1: every patients,

2: conventional fractionation (dose per fraction $\leq$ 2 Gy)

3: hypo+SBRT (dose per fraction $\geq$ 3 Gy)

- disc: discretization option

2. maximum mutual information

3. k-means

The output of the functions includes the data matrices that are pre-processed in different ways:

- data: discretized/ imputed data
- data_c: continuous(normalized)/ imputed data
- data_missing: discretized data/ missing entries left as NaN
- data_c_missing: continuous data/ missing entries left as NaN

Other outputs that could be used by other modules are:

- labels: a cell array of variable names
- mi: mutual information of the discretized variables with respect to a class variable
- studyid: ID of the imported patients
- FracSize: fraction size of the imported patients, required for NTCP calculation
- COMSI: superior-inferior location of a PTV centroid, required for NTCP calculation

It should be noted that the output data matrices (data, data_c, data_missing, data_c_missing) store each patient in one column and each variable in one row, which is a transpose of conventional row-based indexing, in order to be used as an input to the BNT toolbox.

4 Variable Selection

Variable selection (filtering) is often a necessary step towards constructing a robust prediction model when the number of variables is large compared to the available examples. This package includes two filtering implementations: Koller-Sahami (KS) variable filtering and L1-regularized logistic regression (LASSO).

4.1 KS filter

This algorithm chooses a subset of variables in a semi-supervised fashion: every variable is measured, as its "usefulness", a class entropy with respect to a target class under the presence of other variables. The variables that already explains the target class is called the *Markov blanket*. The code implemented a backward elimination approach where it begins with a full set and repeats the rounds of eliminations where one variable with the lowest cross entropy is removed from the set. Detailed theoretical explanation can be found in the original paper by Koller and Sahami [1]

The KS filter can be called in the following way:

```
>> [selected,CEmin,CEvar,CE_rand,blanket] = ...
   KSfilter(data,labels,N,k,verbose)
```

Here are the required arguments:

- data: discretized data without NaN (output of kyu_BN_readdata_combined.m)
- labels: variable names (output of kyu_BN_readdata_combined.m)
- N: desired number of variables to be selected
- k: number of variables to include in the Markov blanket
- verbose: display messages (1) or not (0)

You can see the result of variable selection from the output argument "selected". For the information on other outputs, see a headnote in the file KSfilter.m.

The KSfilter.m requires the data dimensionality (N) and a blanket size (k) to be set by a user. For small datasets, larger k should be avoided in order to prevent over-fragmentation of data and zero counts from computation of

conditional probability. For determining N, the original paper suggests observing the cross entropy of the removed variables as a function of elimination rounds. A sudden increase in the entropy can be taken as a good indication that the optimal dimensionality has reached. However, such a "kink" does not always appear. This implementation takes the approach of measuring the cross entropy of a pseudo variable filled with random values (CE_rand) and taking that value as a cutoff for the best dimensionality. Thus, the whole process of KS filtering is two sequential runs of Stability_KS.m, once to determine the number of variables and the second time to arrive at the final choice of variables. The function repeats the KS filtering in bootstrap replicates, which gives an option for uncertainty estimate on the results.

The following example shows how to use the function Stability_KS.m:

```
>> % first round of variable selection
>> k = 1; % blanket size 1
>> N = k+1; % remove the variables to the smallest set
>> Nrand = 1000; % bootstrap the KS 1000 times
>> verbose = 0; % don't display messages
>> [selected,CElist,CEvar_avg,CErand,blanket,labels] = ...
        Stability_KS(k,N,Nrand,verbose);
The following variables are found:
1. a2m_pre
2. a2m_intra
3. a2m_end
...
choose the variables, separated by comma: 1,4,12,15
bootstrap sample #1
bootstrap sample #2
...
```

After completion of the first KS run, the results can be visualized for determining data dimensionality:

```
>> plotKSresult(CElist,CErand)
```

This will show a plot that looks like figure 2. Variables can safely be removed until its cross-entropy exceeds that of a random variable (indicating it gives no more than noise to a class distribution), which in this case leaves 6 out of 16 variables.

Then the second round of KS filtering is run:

```
>> % second round of variable selection
```

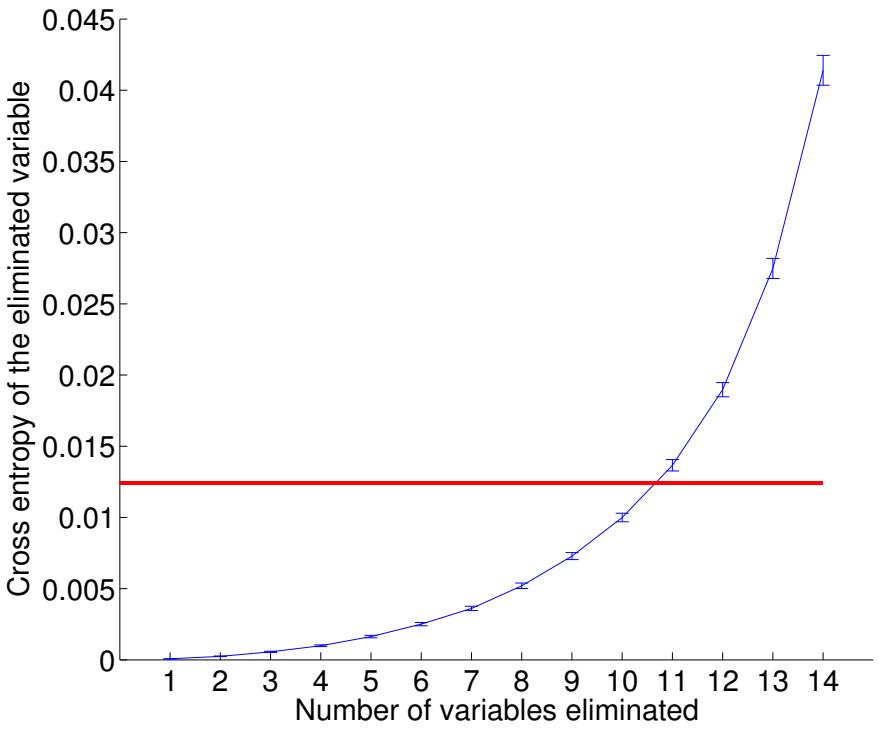

Figure 2: Cross entropy of the variables removed at each round of KS backward elimination. A red line indicates the cross entropy from a random variable.

```
>> Nopt = 6;
>> [selected, CElist, CEvar_avg, CErand, blanket, labels] = ...
        Stability_KS(k, Nopt, Nrand, 0);
```

The first output argument "selected" is a binary matrix that stores selection results for every bootstrap runs (1: selected, 0: not selected). In order to see the frequency on which each variables are selected over the bootstrap runs, simply do:

```
>> selection_frequency = mean(selected, 2);
```

4.2 LASSO

L1 regularization in logistic regression can induce sparsity in a solution by pushing some of the coefficient values towards zero. This filter fits the L1 logistic regression model to the data and selects the variables that have a non-zero coefficient. Similarly to KS, the selection is repeated in bootstrap for obtaining statistics. At each repetition, a shrinkage parameter (λ) of a L1 term is tuned to maximize the fit in a given bootstrap replicate, which is measured in two different metrics that users have to specify: mean square error (MSE) or area under the ROC curve (AUC).

```
>> Nrand = 1000; % bootstrap 1000 times
>> metric = 'AUC' % metric for tuning lambda: 'MSE' or 'AUC'
>> verbose = 0; % don't display messages
>> [selected, model_coal, lambda_hist, err, auc] = ...
    Stability_LASSO(Nrand, metric, verbose)
```

Same as the KS filter, you can look at the matrix 'selected' to see which variables are selected. 'lambda_hist' is a histogram of lambda values chosen during bootstrap repetitions. Note that the λ is chosen from a list of values that are hard-coded in Stability_LASSO.m. So users might need to do the first run, see the output 'lambda_hist' (histogram of lambda values chosen during bootstrap repetitions), and refine the search range of λ by modifying the variable 'lambda' in Stability_LASSO.m.

Inspired by a DREES functionality [2], The code also offers coalesced results which can be seen in a struct 'model_coal' in order to reduce the number of variable selection choices. Coalescence of two variable selection results occurs when the chosen variables are 'similar enough', which is measured by pairwise correlation. For example, if two selection differs by one variable, say

a variable A in one set and B in the other, the two selections are considered the same if A is highly correlated with B.

References

[1] D. Koller and M. Sahami, "Toward optimal feature selection," technical report, Stanford InfoLab, 1996.

[2] J. D. Bradley, A. Hope, I. El Naqa, A. Apte, P. E. Lindsay, W. Bosch, J. Matthews, W. Sause, M. V. Graham, and J. O. Deasy, "A nomogram to predict radiation pneumonitis, derived from a combined analysis of rtog 9311 and institutional data," *International journal of radiation oncology, biology, physics*, vol. 69, no. 4, pp. 985–992, 2007.