William and Mary Bayesian Network Analysis Tool for Mass Spectrometry Data (R-package)

By
Karl W. Kuschner & Qian Si

1 rWMBAT Methods for Diagnostic Variable Selection from Mass Spectra……….1
1.1 Introduction…………………………………………………………………..1
1.2 Background……………………………………………………………......2
1.3 Methods………………………………………………………………………3
1.4 Example Data………………………………………………………………..6
1.5 Concerns……………………………………………………………………..9
1.6 Conclusions & Recommendations…………………………………………..10
2 Functions in the rWMBAT Library…………………………………………………..10
2.1 BuildBayesNet - Select Features and Metafeatures Based On Mutual Info.10
2.2 ChooseMetaVars - Combine Variables into Metavariables…………………11
2.3 clearirrarcs - Clear Arcs That Are Not C->V Or C->V<->V…………….…….11
2.4 clipclassconnections - Delink Variables From Class…………………………12
2.5 DoTheMath - Perform Feature Selection for the Data………………………13
2.6 FindProbTables - Estimate the Probabilities P (class=c|data=D)……………14
2.7 getarcs - Build Adjacency Matrix For A Set Of Variables……………………14
2.8 WMBAT-The William and Mary Bayesian Analysis Tool……………………..15
3 Generic Tool Functions………………………………………………………………16
3.1 Repmat - Create a Matrix Consisting Of An m-by-n Tiling Copies Of X……16
3.2 InitialProcessing – Input Data Prep after Custom Signal Processing……....17
3.3 MutualInfo - Calculate Mutual Information Of Two Variables………………..18
3.4 findmutualinfos - Find Various Mutual Info Combos among Variables……..18
3.5 CondMutualInfo - Calculate Mutual Information of Two Variables Conditioned On a Third…………………………………………………………………19
3.6 automi - Find A Threshold For Randomized MI (Var, Class)………………..19
3.7 opt2bin - Optimize Boundary for Each Variable to Maximize MI…………….20
3.8 opt3bin - Find 3 Bin Boundaries Optimizing the MI of Each Variable………21
3.9 looklr - Find Boundary…………………………………….…………………..22
3.10 TestCases - Classify Using Bayes Rule……………………………………….23
4 References………………………………………………………………………………..23
5 Acqnowledgement……………………………………………………………………….24

1. rWMBAT Methods for Diagnostic Variable Selection from Mass Spectra

1.1 Introduction

The goal of the William and Mary Bayesian Network Analysis Tool (WMBAT) is to solve two key problems in feature selection and classification of mass spectrometry (MS) data, as described by the authors of the caMassClass tool [1]: Low data reproducibility, and the possibility of false discovery.

We believe that both problems arise from a similar root cause (given the variations inherent in the instrument), that of small sample effects. Features which may appear to be important in a small sample are then found to be non-diagnostic in another group. This leads to classifiers that fail to work during blind tests and random features being pursued as biomarkers.

An additional problem which plagues some classification schemes is the existence of many correlated features in MS data. For example, if two features exist because a molecule appears at a singly and doubly charged ion state in the mass spectra, they will likely be highly correlated. Feature selection methods such as forward selection based on a naïve Bayesian classifier wrapper will select one feature, but not the other, since the second adds no additional classification power.

The primary objective was to *select stable features that would extend to new data*. The secondary objective was to describe the relationships between these features, such as which were likely to be primary, or parent, ions and which were modifications or satellites. A tertiary objective was to provide a classifier, but one limited to stable feature sets whose error rates may not be as low as other, less stable methods.

To achieve these objectives, we chose to encode a specific type of Bayesian (or belief) network. We limited the structure to have a root node that represented the disease state under study, and whose children (immediate descendants) were features whose relation to the disease were stable and independent of other features.

1.2 Background

This work is part of an ongoing project whose goal is to create tools for “computationally improved signal processing for mass spectrometry data.” One of the steps in that project, and the focus of this work, is the development of methods to exploit the improved MS data to find biologically relevant information.

A mass spectrometer is an instrument that takes some sample of material, biologic or otherwise, and measures the relative amounts of constituent materials—ordered by molecular mass —in the sample. The output is called a mass spectrum and is initially continuous in nature, with very low signals representing mass regions where nothing was found, and spike-shaped structures (called “peaks”) representing a relatively large amount of material at a particular mass.

One type of mass spectrometry instrument works by ionizing the molecules in a sample, typically by an intense laser pulse or ion collision, then accelerating the resulting ions through an electric potential of a few kV. After the molecules have been accelerated to some terminal velocity v, which depends on their mass m and electric charge z as well as the electric potential V, they float down a field-free time of flight (TOF) tube and strike a detector. The energy E gained relates the electric potential and velocity by $E = zV = \frac{1}{2}mv^2$. Low mass ions reach a higher velocity and hence strike the detector first; heavy ions are detected last. By measuring the number of detections along a time scale, then converting the time axis into mass per unit charge (m/z), a spectrum of signal intensity vs. m/z is created. While this is not the only method of mass spectrometry, it is a common one used in the field of proteomics. Its ability to survey a wide range of mass values aids the search for important proteins, as opposed to other methods, which might search for the abundance of a material at a specific m/z value.

There are several errors associated with this type of instrument. Although we would like the peaks to be infinitely narrow “spikes,” they in fact have finite width due to the method of ionization and detection. In addition, the time that a specific molecule arrives differs slightly from trial to trial, and the intensity measured can vary for reasons other than true abundance variations in the sample. Another important error arises because of the violence of the initial ionization and the several ways a single molecule can show up—with charge z >1 (called multiply-charged states), in fragments, or with small common molecules such as the chemical matrix attached (adducts) or detached (neutral loss). These processes result in peaks at different m/z values that actually represent a single underlying molecule.

1.3 Methods

WMBAT uses *mutual information* as a scoring criterium to construct a three level Bayesian network structure with the disease state as the uppermost node. For more information on these concepts, see Cover [2] and especially Jensen [3]. Figure 1 shows the final result of one analysis. In that figure, the node representing the disease state is labeled “Class,” the numbered nodes represent specific m/z values found in the data set. Features connected directly to the class (“first level features”) are those which have been determined to be stable indicators of the disease under many randomized k-fold cross validation trials. This type of cross-validation has been shown to decrease variance and increase stability [4]. Those on the second level are related to the first level features in some fashion and have been found to provide information on the disease state only through the first level features.

By identifying these relationships, it may be possible for those trying to chemically identify biomarker candidates to understand the parent ion. In the data represented by Figure 1, for example, a child of feature 203 was tentatively identified as a des-argenine modification of the parent—an additional clue to the identity of the primary feature. (More in-depth discussion of the experimental data is to appear in BMC Bioinformatics, 2010.)

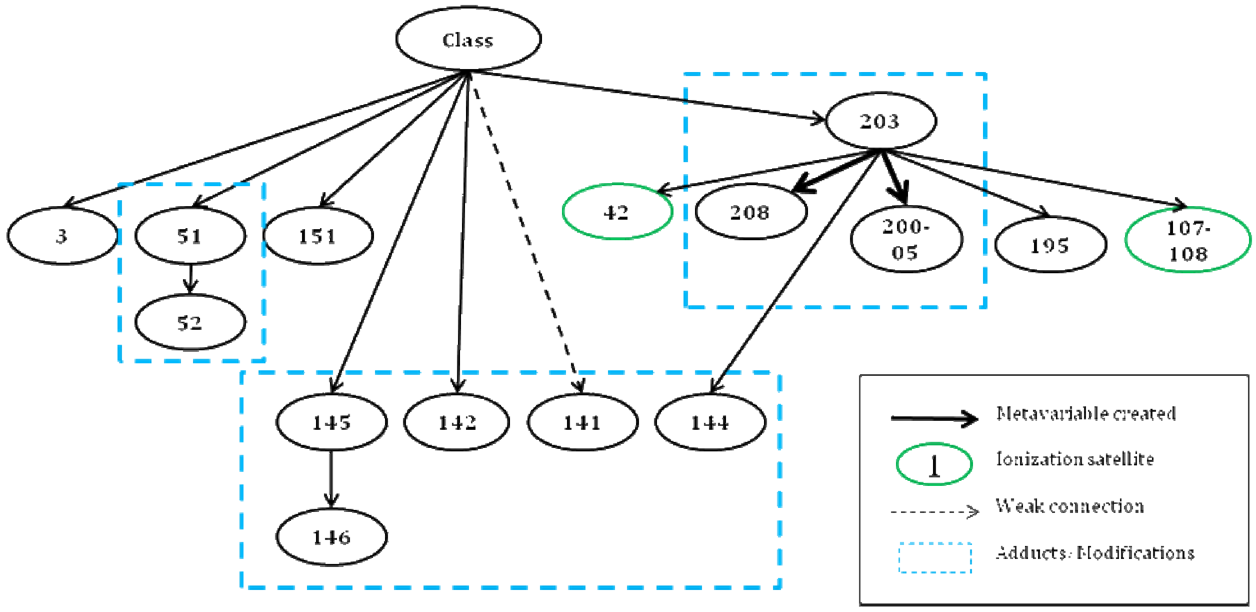

Figure 1: Final result of WMBAT analysis

The first level of features below the disease class are those features most likely to be parent ions that are diagnostic of the disease state, or class variable. The second level below the class node are those features that are found to be correlated to the parent ions. These are often ionization satellites, adducts, or modifications of a parent molecule. They are included so that the user has additional information that might assist in the identification of the parent (diagnostic) feature.

The input to the tool is a *n*-by-*v* data matrix from a set of mass spectrometry measurements, where n is the number of spectra derived from the physical specimens, and *v* is a fixed number of specific m/z points along the spectra. The entries are the measured intensity or abundance values of each species spectrum at the same (global) discrete m/z values.

The output is a set of data elements that allow the user to build a Bayesian network from the *v* features identified in the input. The frequency of connections between the disease class variable and the feature set, as well as the frequency of connections between features, is given. Feature pairs that may represent a single molecule are attempted to be combined into a meta-variable; successful combinations are noted as well. The predicted disease class for each specimen is recorded, and the overall classification error rate for each trial is reported. Other outputs return the subject ID and specimen data after any optional processing (such as normalization) is completed. While the result shown in Figure 1 is graphical, the output of the algorithm is numeric. Specifically, the first level variables are described by a vector whose i^{th} entry is the frequency with which feature *i* was found to be connected to the class.

Similarly, the second level features are described in an array whose (*i,j*) element is the frequency that a connection was found between feature *i* (a first level feature) and feature *j*. Additionally, the algorithm attempts to find feature pairs that may be arithmetically combined to create a more diagnostic feature. This might be the case when, for example, a sodium atom attaches to the parent ion during the measurement, and some of the abundance of the parent ion shows up at a different m/z position. These “metavariables” are also reported as (*i,j*) pairs, where the algorithm determines that a better feature is created when first level feature *i* is added to another feature *j*.

Since the signal processing of the MS spectra is done outside the algorithm, only a minimal amount of data pre-processing is accomplished. Options include the setting of all negative values to 0 (there are no real negative abundance), normalization, and replicate averaging. Normalization is done by multiplying the sum of all features of a given case by a factor that makes all cases have the same sum. This is an attempt to correct for unusually low (or high) signal strength for a given spectrum.

Since the data we built the tool around had replicates, or multiple measurements from single samples, we included an option to replicate average the data. If selected, features are arithmetically averaged among cases with the same patient ID.

After this preprocessing, the algorithm enters two embedded loops: the first repeats the entire process some number of times as set by the user. See Figure 2 for more detail.

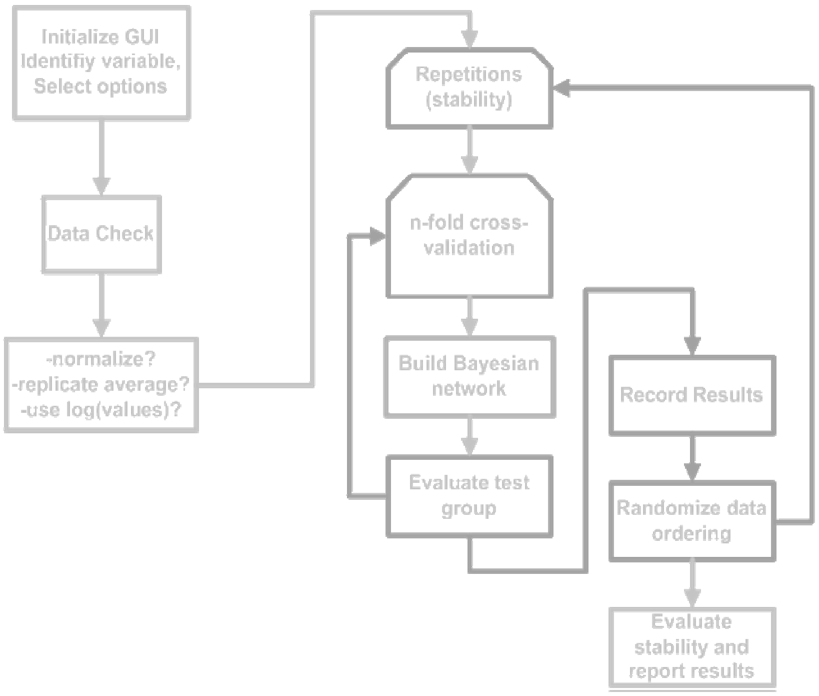

Figure 2: Algorithm flowchart

The inner loop is for implementation of the *k*-fold cross validation. Each cycle of the inner loop results in *k* networks, with *k* sets of first-level, second-level, and metavariables, but with a single error rate. This error rate is the fraction of all cases that were determined to be a different class than that recorded in the original data. In actuality, a probability of class is calculated, but the algorithm is defaulted to determine class by a simple test of probability of disease greater or less than 50%.

Once a training group and testing group have been selected inside the cross validation loop, the algorithm finds a mutual information (MI) threshold by randomizing class labels a large number of times, and determining a significance threshold by taking the 99th percentile largest randomized MI. A user derived factor can increase this significance threshold to more conservatively test for stable features.

Given this MI threshold, features are tested to see if the MI between a feature and the disease class exceeds the threshold. If so, they are tentatively placed on the first level.
All features are then tested pair-wise to determine feature to feature connections, using the same MI technique and (scaled) threshold. Those connected to a first level feature (but not a first level feature themselves) are placed on the second level.

Where two first level features are connected, the algorithm attempts to test for conditional independence between the class and each feature, given the other, in order to determine if one is perhaps a child of the other. Details on this test and other aspects of the algorithm can be found in Kuschner [5].

After all the desired repetitions of this process are complete, the output is reported as described above. The graphical description of the network, as seen in Figure 1, must be done manually by the user. Classification of blind or other external data must be done manually after the user has examined the results reported and chosen stable features based on their own needs.

1.4 Example Data

In the example data set included with this package, we reproduced those systematic and statistical properties we have found in certain real data (BMC Bioinformatics 2010), without the several artifacts that we have no specific explanation for (such as certain peaks failing to appear in some replicates).

The primary purpose of this data set is for quality control and testing of the algorithm. By mimicking known properties of the real data, then attempting to identify those properties with algorithms made for that purpose, we gain a better understanding of the reliability and stability of the protocols used.

The following steps were taken to prepare the generated data:

1. A spectrum[1] with 200 peaks is created by taking the mean and standard deviation of the non-disease members of a real data set. This provides a baseline for creating all the cases that will be used.

2. A set of spectra, with the number of cases approximating the number of unique patient identification numbers in a real data set, is generated via a draw from a Gaussian distribution for each variable independently, using those values of mean and standard deviation. At this point there should be no real distinction between any of the 200 variables.

3. One-half of the population is designated to be in the disease class. A class vector representing this choice is created and attached to the data.

4. One peak (labeled 200) is chosen as "highly diagnostic" and the mean values of the two subpopulations (normal and disease) are separated by two times the population's average standard deviation. Specifically, the disease cases are redrawn from N(μ+2σ,σ). This results in a distribution like the one shown in Figure 3.

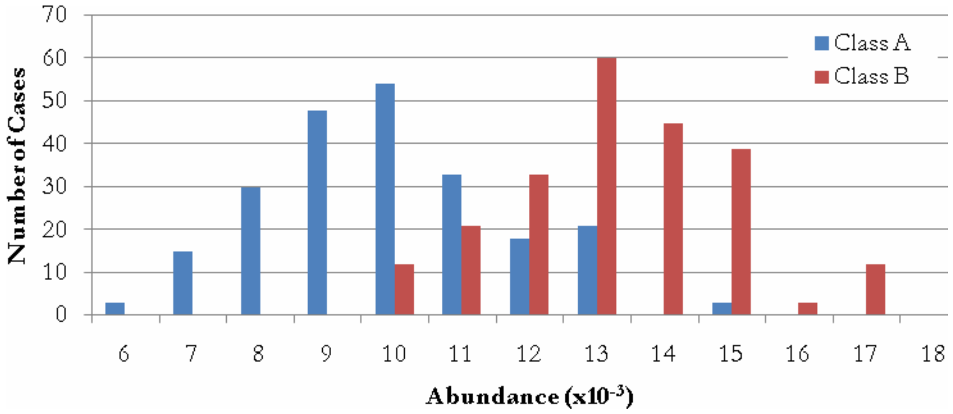

Figure 3: Generated data distribution for highly diagnostic peak

1. A random fraction (about a tenth) of the total value of this peak is placed into each of four adjacent peaks (labeled 195-199). In this manner, five diagnostic peaks are

[1] A full spectrum is not created as we do not wish to replicate the signal processing steps. Instead, the steps here are applied to the final peak list data.

created, all diagnostic of the class. This procedure mimics the measurement of adducts or modifications in the real data set, wherein slightly modified molecules show up as peaks separate from the original.

2. A small fraction of the value of the key peak (200) is moved into a peak some distance away in the list (labeled 100), representing a multiply-charged ionization satellite ($z = 2$). This is repeated to a different peak (labeled 99) for one of the adducts (199).

3. Another moderately diagnostic[2] peak is created but not added to the peak list. Instead, varying portions of the total value of that peak are placed in two non-adjacent peaks (labeled 50 and 150). This represents the breaking apart of a biomarker protein, whose mass is too great to be detected, into several fragment molecules that are in the range of measurement.

4. Two more peaks (labeled 1 and 2) are selected as "mildly diagnostic" and the values chosen from two normal distributions whose means are separated by about one standard deviation of either group. Specifically, the disease cases are redrawn from $N(\mu+\sigma,\sigma)$. One of these two peaks has a portion of the other peak's value added to it to represent two peaks that are so close together that the peak value of one is "riding up" on the tail of another.

5. The cases are replicated three times (the original of each case is discarded) by multiplying each value by a de-normalization factor to replicate the signal strength and chemical preparation effects. For a single data vector **X,** a factor *f* is first selected from ~U(0.5, 2.0) to replicate the range of total ion current normalization factors found in the Leukemia data. The resulting distribution for the highly diagnostic peak is shown in Figure 4.

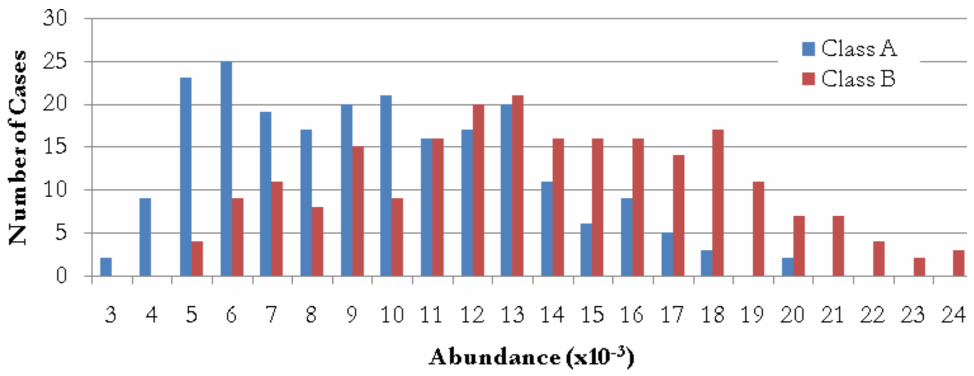

Figure 4: Distribution for highly diagnostic peak after de-normalization

A summary of the diagnostic peaks placed in the generated data is given in Table 1.

[2] Difference in means is about one and a half standard deviations of the sub populations.

Table 1: Diagnostic variables, generated data

Peak	Purpose
200	Highly diagnostic
196-199	Adducts or modifications of peak 200
99, 100	Correlated doubly charged ionization states of 199, 200
1, 2	Diagnostic with correlations due to mixing
3, 4	Mildly diagnostic
50, 150	Diagnostic—but hidden—primary peak

With this data, the algorithm should output a highly stable set of connections that result in the Bayesian network shown in Figure 5. The user may want to vary the threshold factor from 2.5 to 3.5 and note the minimum error rate around 3.2 with an average of 6 first level features. In our test, 6 features (3%) were selected nearly 100% of the time. 93% of the others are almost never selected. The first level connections occurring in more than one-half of trials in a typical run are shown in Table 2.

Table 2: First Level Variables. Table 2 shows those features found most often to be directly connected to the class node. One unintended feature was found 48% of the trials.

Feature	Selection Frequency
1	100%
2	100%
3	95.1%
4	99.7%
150	99.9%
200	99.6%

Only two first-level features had other features frequently connected at the second level. Feature 200 was found to be the parent of features 195-199 and 100, all more than 99% of the trials. Feature 150 was connected to feature 50, but in only 13% of the trials. These two features were fragments of a non-measured feature.

The only diagnostic feature not identified by the BN algorithm at either the first level or the second level of nodes more than 50% of the time was feature 99. However, this was a correct result, since feature 99 was intended to be a child of feature 199, which itself was derived from feature 200. Therefore, it should have been identified as a third level node and eliminated, which is indeed what occurred.

Error rates around 14% are achieved, higher than may be possible with other feature selection methods. However, the network chart is a near perfect representation of the planned relationships, except for the unstable connection found from the Class node to feature 99 (a third level feature) and the failure to specifically identify the hidden feature H which the algorithm is not designed to find. The connection from 1 to 2 was only found occasionally.

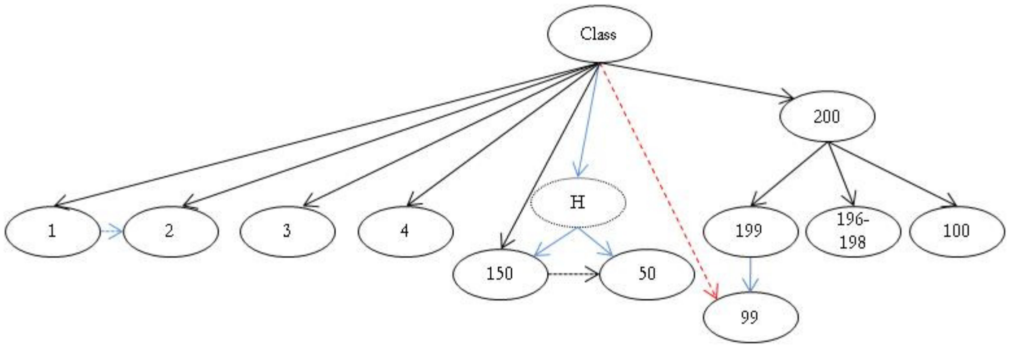

Figure 5: Planned and resulting Bayesian Network. Black represents connections planned and found. Blue represents connections planned, but not found. Red represents connections found, but not planned. Dotted lines were found infrequently.

1.5 Concerns

The thresholding procedure is somewhat subjective, although we have been able to achieve good results by starting with a threshold value of 2.0, and increasing or decreasing the value by 0.2 as is indicated by the density of connections found in the results. The mean error rate of all trials at a given threshold should be examined. The threshold which achieves near-minimum error, stable feature selection, and a reasonable number of features is the best. In the example data set in Figure 6, a reasonable choice for threshold would be 3.2, as it minimizes error rate while choosing a reasonable number (7) of variables.

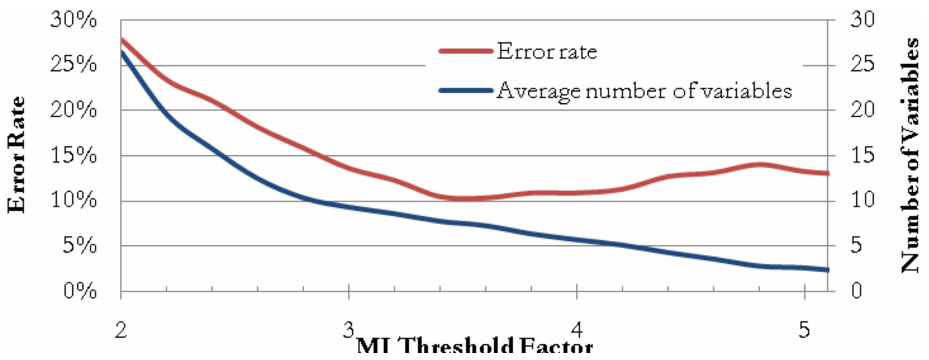

Figure 6: MI threshold effects under 10-fold cross-validation

A second area of concern is the metavariables creation. We have not yet found this to significantly decrease error rate in withheld data, and feel more work needs to be done on this part of the analysis. We recommend that users consider the combination of features carefully before using the results in more complicated analyses.

1.6 Conclusions and Recommendations

We have tested the tool on several data sets with varying results. Some artificial data (example included), derived using the parameters of real blood sera data (see coming BMC 2010 publication and [5]), recreated the intended relationships between features including all induced correlations. In real data, we have had some mixed results. A leukemia study found very stable features that classified withheld data with error rates as predicted by training data. Another preliminary data set had less stable features and is undergoing more study.

We believe that if the data produces stable features (>75% selection rate for network connections), the resulting most likely network will provide insight into the underlying feature relationships with the disease.

2. Functions in rWMBAT Library

2.1 BuildBayesNet - Select Features and Metafeatures Based On Mutual Info

Description
BuildBayesNet selects features and metafeatures based on mutual info

Usage
BuildBayesNet(data, class, ffactor, drop)

Arguments

data	integer array containing the data used to build the Bayes net, cases in rows, variables in cols,
class	double column vector, the known class variable for each case
ffactor	multiple of auto MI to use to threshold C->V connections
drop	double, MI loss percentage threshold for testing independence. Set to .75 and adjust to filter too few/too many variable-to-variable connections.

Details
This function takes a set of training data and an additional variable called "class" and tries to learn a Bayesian Network Structure by examining Mutual Information.

Value
a matrix of zeros and ones, where one in row i, column j denotes a directed link in a Bayesian network between variable i and variable j. The class variable is the last row/column.

Note
CALLED FUNCTIONS
automi: finds an MI threshold based on data
findmutualinfos: finds all values MI(VC), MI(VV) and MI(VC|V)

Author(s)
Karl Kuschner, Qian Si and William Cooke, College of William and Mary, Dept. of Physics, 2009.

References

http://kwkusc.people.wm.edu/dissertation/dissertation.htm

Examples

adjacency <- BuildBayesNet(data, class, ffactor, drop)

2.2 ChooseMetaVars - Combine Variables into Metavariables

Description

Finds the V-V pairs in the adjacency matrix, and attempts to combine them into a metavariable with higher mutual information than either variable alone. If it is possible to do this, it returns a new data matrix with the variables combined.

Usage

ChooseMetaVars(data, class, adj)

Arguments

data	double array of discrete integer (1:n) values, cases in rows and variables in columns.
class	double column vector, also 1:n. Classification of each case.
adj	Logical adjacency matrix, number of variables+1 by number of variables. Last row is class node. Logical meaning "there is an arc from i to j.

Value

metamatrix logical whose (i,j) means "variable j was combined into variable i (and erased)"

finaldata	double data matrix with the variable combined and rebinned
leftbound	vector, the new left boundary for binning
rightbound	vector, the new right boundary for binning

Note

CALLED FUNCTIONS:
opt3bin: rebins combined variables to determine highest MI

Author(s)

Karl Kuschner, Qian Si and William Cooke, College of William and Mary, Dept. of Physics, 2009

References

http://kwkusc.people.wm.edu/dissertation/dissertation.htm

Examples

result <- ChooseMetaVars (data, class, adj)

2.3 clearirrarcs - Clear Arcs That Are Not C->V Or C->V<->V

Description

Given an adjacency matrix with V<->V arcs in a square matrix and an additional row representing C->V (class to variable), this function clears out all V1->V2 arcs where V1 is not a member of the set of V's that are class-connected, i.e. have arcs in the final row.

Usage

clearirrarcs(adjin)

Arguments

adjin logical array where a true value at position (i,j) means that there is an arc in a directed acyclic graph between (variable) i and variable j.

Value

copy of adjin with unneeded arcs cleared

Author(s)

Karl Kuschner, Qian Si and William Cooke, College of William and Mary, Dept. of Physics, 2009

References

http://kwkusc.people.wm.edu/dissertation/dissertation.htm

Examples

```
adjout <- clearirrarcs( adjin )
```

2.4 clipclassconnections - Delink Variables From Class

Description

Where two variables are connected to each other and also to the class, attempt to select one as the child of the other and disconnect it from the class. Use MI(Vi;C|Vj)<<MI(Vi;C) as a test.

Usage

clipclassconnections(adj, mivc_vec, mivcv, dropthreshold)

Arguments

adj	logical matrix where "true" entries at (i,j) mean "an arc exists from the Bayesian network node Vi to Vj." The class variable C is added at row (number of V's + 1). "0" values mean no arc.
mivc_vec	double row vector containing MI(C;Vi) for each variable
mivcv	double array whose (i,j) entry is MI(Vi,C|Vj)
dropthreshold	double percentage drop from MI(Vj;C) to MI(Vj;C|Vi) before declaring that Vi is between C and Vj

Value

copy of adj with the appropriate arcs removed

Author(s)

Karl Kuschner, Qian Si and William Cooke, College of William and Mary, Dept. of Physics, 2009

References

http://kwkusc.people.wm.edu/dissertation/dissertation.htm

Examples

```
adjout <- clipclassconnections(adj, mivc_vec, mivcv, dropthreshold)
```

2.5 DoTheMath - Perform Feature Selection for the Data

Description

DoTheMath takes a data array, class vector, and other information and builds and assesses a Bayesian network after selecting features from within the data array.

Usage

DoTheMath(InputStructure)

Arguments

InputStructure	A list containing following inputs
Class	vector of length "cases", with discrete values identifying class of each case (may be integer)
ID	double patient ID array of length cases, with one or more cols
MZ	Vector of length "variables" holding labels for variables
Options	Logical 6x1 array. Options are: 1. Normalize on population total ion count (sum across rows) 2. Remove negative data values by setting them to zero 3. After normalizing, before binning, average cases with same ID 4. Find the MI threshold by randomization 5. Take log (data) prior to binning. Negative values set to 1. 6. Remove Low Signal cases NOT DONE: 3 Bin (2 Bin if False)
n	integer, the "n" in n-fold cross validation
repeats	Integer, times to repeat the whole process (e.g. re-crossvalidate)
threshold	double factor by which the maximum "random" MI us multiplied to find the minimum "significant" MI (double, 1.0-5.0)

Details

This is the umbrella script that loops a specified number of times (see "repeats" above), each time doing a full n-fold cross validation and recording the results. All input and output data are stored in a single data structure, described below.

Value

OutputDataStructure	all the fields of InputStructure, plus
ErrorRate	Vector containing misclassification rate for each repeat
KeyFeatures	Index to vector MZ that identifies features selected

Note

CALLED FUNCTIONS
InitialProcessing: Applies the options listed above
BuildBayesNet: Learns a Bayesian Network from the training data
ChooseMetaVars: Combines variables that may not be physically separate molecules.
TestCases: Given the BayesNet, tests the "test group" to determine the probability of being in each class.
opt3bin: Discretizes continuous data into 3 bins, optimizing MI FindProbTables: Learns the values P(C,V) for each variable cvpartition and training are MATLAB Statistics toolbox functions.

Author(s)
Karl Kuschner, Qian Si and William Cooke, College of William and Mary, Dept. of Physics, 2009

References
http://kwkusc.people.wm.edu/dissertation/dissertation.htm

Examples

```
OutputDataStructure <- DoTheMath (InputStructure)
```

2.6 FindProbTables - Estimate the Probabilities P (class=c|data=D)

Description
Input a training group of data arranged with cases in rows and variables in columns, as well as the class value c for that vector. Each case represents a data vector V. For each possible data value vi, and each variable Vi, it calculates P(C=c|Vi=vi) and stores that result in a 3-D table. The table is arranged with the dimensions (class value, data value, variable number).

Usage
FindProbTables(data, class)

Arguments

data	double array of discrete integer (1:n) values, cases in rows and variables in columns
class	double column vector, also 1:n. Classification of each case

Value
3-D array whose (c,d,v) value is P(class=c|data=p) for variable v

Author(s)
Karl Kuschner, Qian Si and William Cooke, College of William and Mary, Dept. of Physics, 2009

References
http://kwkusc.people.wm.edu/dissertation/dissertation.htm

Examples

```
probtable <- FindProbTables(data, class)
```

2.7 getarcs - Build Adjacency Matrix For A Set Of Variables

Description
By comparing mutual information between two variables to thresholds determined seperately, this function declares there to be an arc in a Bayesian network. Arcs are stored in an adjacency matrix, described below.

Usage
getarcs(mvc, vcthreshold, mvv, vvthreshold)

Arguments

mvc	double vector array with mutual information between variables and the class (variables and other variables). The (i,j) entries of mvv are MI(Vi,Vj).

vcthreshold	scalar threshold used to test for existence linkz
mvv	double vector matrix
vvthreshold	scalar threshold used to test for existence linkz

Details
The primary tests are: MI(Vi;Cj)>>vcthreshold : tests for links between Vi and the class MI(Vi;Vj)>>vvthreshold : tests the links between variables

Value
logical matrix whose entries "1" at (i,j) mean "an arc exists from the Bayesian network node Vi to Vj." The class variable C is added at row (number of V's + 1). "0" values mean no arc

Author(s)
Karl Kuschner, Qian Si and William Cooke, College of William and Mary, Dept. of Physics, 2009

References
http://kwkusc.people.wm.edu/dissertation/dissertation.htm

Examples

```
adjacency <- getarcs( mvc, vcthreshold, mvv, vvthreshold )
```

2.8 WMBAT - The William and Mary Bayesian Analysis Tool

Description
WMBAT takes an array of mass spec peak intensities, a vector describing which of two classes each sample belongs to, and other information and builds and assesses a Bayesian network after selecting features (peaks) from within the data array that are diagnostic of the class. The primary output is an adjacency matrix describing the resulting Bayesian network.

Usage
WMBAT(tofListMetaData, alignedPeakList, Options, nfold, repeats, threshold)

Arguments

alignedPeakList	List contain following information peaks: double vector holding time labels for peaks data: double vector holding peak intensity value for corresponding spectrum
tofListMetaData	a list containing information about every spectra, the information related to the spectra we needed in this package is class and ID. Class: Integer vector, values 1 or 2 identifying the class of each case, such as "disease, non-disease" ID: Double one or two column array contains the sample ID for each case. Second column is optional and would identify replicates of the same sample.
Options	Logical 6x1 array. Options are: 1. Normalize on population total ion count (sum across rows) 2. Remove negative data values by setting them to zero 3. After normalizing, before binning, average cases with same ID 4. NOT USED - SET TO FALSE 5. Take log(data) prior to binning. Negative values set to 1. 6. NOT USED - SET TO FALSE
nfold	integer, the “n” in n-fold crosses validation (integer 4-10). 10 is recommended
repeats	integer, times to repeat the whole process (e.g.re-crossvalidate). 100 is

	recommended
threshold	double, factor by which the maximum "random" MI is multiplied to find the minimum "significant" MI (double, 1.0-5.0). We recommend starting with 1 and increasing until a "reasonable" number of diagnostic peaks is reached and error rates are minimized. This setting is dependent on the data and the correlations between variables

Value

IntOut	double Intensities input array, after processing by the various options selected by the logical Options above
IDOut	double vector, the ID number of each row in the IntOut array. With no replicate averaging, each ID will be preserved (but reformatted) from the input. With replicate averaging, only the primary ID number remains.
PredClass	double matrix, the predicted class of each case, during each of the trials (from input "repeats")
Class2Vars	vector whose ith value is the fraction of times peak i (from the vector MZ) was selected as being connected to the class. The maximum times it could have been selected was nfold*repeats
Var2Vars	integer array whose (i,j) entry is the fraction of times a second level link was found from peak i to peak j, when peak i was connected to the class, as found in SumLvl1
MetaVars	integer array whose (i,j) entry is the fraction of times a metavariable was created using peak i and peak j and stored in the level
TrialErr	double vector, the error rate for each of the "repeats" possible trials. Records the percentage of cases where PredClass was not equal to the input Class.

Note

CALLED FUNCTIONS:
DoTheMath: Learns a Bayesian Network from the data

Author(s)

Karl Kuschner, Qian Si and William Cooke, College of William and Mary, Dept. of Physics, 2009

References

http://kwkusc.people.wm.edu/dissertation/dissertation.htm

Examples

```
result <- WMBAT (Intensities,Class, ID, MZ, Options, nfold, repeats, threshold)
```

3. Generic Tool Functions

The functions in this section are generic tools that were written in order to support the rWMBAT library

3.1 Repmat - Create a Matrix Consisting Of An m-by-n Tiling Copies Of X

Description

creates a large matrix consisting of an m-by-n tiling of copies of X

Usage

Repmat(X, m, n)

Arguments

X	double matrix used to make copies of
m	Integer row number of the result matrix
n	Integer column number of the result matrix

Value
m-by-n double matrix

Author(s)
Qian Si, College of William and Mary, Dept. of Physics, 2009

Examples
repmat<-Repmat(X,m,n)

3.2 InitialProcessing – Input Data Prep after Custom Signal Processing

Description
Takes Peaklists That Have Been Imported into R And Prepares Them For Bayesian Analysis

Usage
InitialProcessing(StructIn)

Arguments
StructIn   list with the following double-typed arrays

Intensities: double n x m real-valued array with variables (peaks) in columns, cases (samples) in rows.

MZ: double list of the labels (m/z value) for each of the variables. Must be the same size as the number of variables in Intensities

Class: Integer vector, classification of each sample (disease state)– 1 or 2–must be the same size as the number of cases in Intensities

ID: double column array, case or patient ID number, same size as class. May have second column, so each row is [ID1 ID2 where ID2 is replicate number.

Options: logical array of processing options with elements:

1. Normalize
2. Clip Data (remove negatives)
3. Replicate Average
4. Auto threshold MI
5. Use Log of Data
6. Remove Low Signal cases

NOT DONE: 3 Bin (2 Bin if False)

Value

RawData	Intensities as input
ClipData	RawData where all values less than 1 are set to 1
NormData	ClipData normalized by total ion count, i.e. divided by the sum of all variables for each case
LogData	Natural logarithm of NormData
Class	Same as input

MZ	Same as input
ID	Single column. If replicates are not averaged, the entries are now ID1.ID2. If replicates averaged, then just ID1
DeltaMZ	difference in peak m/z values to look for adducts
RatioMZ	ratios of m/z values ot look for satellites

Author(s)
Karl Kuschner, Qian Si and William Cooke, College of William and Mary, Dept. of Physics, 2009

References
http://kwkusc.people.wm.edu/dissertation/dissertation.htm

Examples

```
StructOut <- InitialProcessing( StructIn)
```

3.3 MutualInfo - Calculate Mutual Information Of Two Variables

Description
calculating mutual information of the two variables

Usage
MutualInfo(v1, v2)

Arguments

v1	one of the two vectors of which MI is calculated
v2	one of the two vectors of which MI is calculated

Value
mutual information of the two vectors

Author(s)
Bill Cooke, College of William and Mary, Dept. of Physics, 2009

Examples
MI <- MutualInfo(v1, v2)

3.4 findmutualinfos - Find Various Mutual Info Combos among Variables

Description
Given a set of data (many cases, each with values for many variables) and
an additional value stored in the vector class, it finds MI described
below in "Value."

Usage
findmutualinfos(data, class)

Arguments

data	A number of cases (in rows), each with a measurement for a group of variables (in columns). The data should be discredited into integers 1 through k. The columns are considered variables V1, V2, ...
class	A column vector of length "cases" with integer values 1,2..., an additional measurement of class C.

Value

mi_vc	a row double vector whose ith value is MI(Vi,C).
mi_vv	double symmetric matrix with values MI(Vi,Vj).
mi_vc_v	double non-sym matrix with values MI(Vi;C|Vj).

Note
CALLED FUNCTIONS:
MutualInfo - Calculate Mutual Information Of Two Variables CondMutualInfo-Calculate Mutual Information of Two Variables Conditioned On a Third

Author(s)
Karl Kuschner, Qian Si and William Cooke, College of William and Mary, Dept. of Physics, 2009

References
http://kwkusc.people.wm.edu/dissertation/dissertation.htm

Examples

```
result <- findmutualinfos(data, class)
```

3.5 CondMutualInfo-Calculate Mutual Information of Two Variables Conditioned On a Third

Description
calculating the mutual information of two variables conditioned on a third

Usage
CondMutualInfo(V1, V2, condV)

Arguments

V1	one of the two vectors of which MI is calculated
V2	one of the two vectors of which MI is calculated
condV	given condition vector

Value
mutual information of two variables conditioned on a third

Author(s)
Qian Si and William Cooke, College of William and Mary, Dept. of Physics, 2009

Examples
MIxyz <- CondMutualInfo(V1, V2, condV)

3.6 automi - Find a Threshold for Randomized MI (V C)

Description
Finds the threshold of a data set's mutual information with a class vector, above which a variable's MI (class, variable) can be expected to be significant.

Usage
automi (data, class, repeats)

Arguments

data	double array of discrete integer (1:n) values, cases in rows and variables in columns.
class	double column vector, also 1:n. Classification of each case.
repeats	Integer, the number of times to repeat the randomization

Details
The threshold for mi (significance level) is found by taking the data set and randomizing the class vector, then calculating MI (CV) for all the variables. This is repeated a number of times. The resulting list of length (repeats *variables) is sorted, and the 99th percentile max MI is taken as the threshold.

Value
a threshold for randomized MI(V C)

Author(s)
Karl Kuschner, Qian Si and William Cooke, College of William and Mary, Dept. of Physics, 2009.

References
http://kwkusc.people.wm.edu/dissertation/dissertation.htm

Examples
threshold <- automi(data, class, repeats)

3.7 opt2bin - Optimize Boundary for Each Variable to Maximize MI

Description
This function takes an array of continuous data, with cases in rows and variables in columns, along with a vector "class" which holds the known class of each of the cases, and returns an array "binneddata" that holds the 2 bin discretized data.

Usage
opt2bin(rawdata, class, steps, typesearch, minint = NA, maxint = NA)

Arguments

rawdata	double array of continuous values, cases in rows and variables in columns. Distribution is unknown
class	double column vector, values 1:c representing classification of each case

steps	Integer, number of steps to test at while finding maximum MI
typesearch	values=0: starting boundary based on data's actual max/min values =1: use the value passed in max as maximum (right) value =-1: use the value passed in min as minimum (left) value =2: used values passed via max, min
minint	vectors whose values limit the range of search for each variables boundaries
maxint	vectors whose values limit the range of search for each variables boundaries

Details
The discretization bin boundary is found by maximizing the mutual information with the class the resulting MI and boundary are also returned. The starting boundaries for the search can be given in the vectors min and max, or either one, or neither, in which case the data values determine the search boundaries.

Value

mi	double row vector holding the maximum values of MI(CVi) found
boundary	double vector, the location used to bin the data to get max MI
binneddata	double matrix, the resulting data binned into "1" (low) or "2" (hi)

Author(s)
Karl Kuschner, Qian Si and William Cooke, College of William and Mary, Dept. of Physics, 2009

References
http://kwkusc.people.wm.edu/dissertation/dissertation.htm

Examples

```
result <- maxMIbin(rawdata, class, typesearch ,min, max)
```

3.8 opt3bin - Find 3 Bin Boundaries Optimizing the MI of Each Variable

Description
This function takes an array of continuous sample data of size cases (rows) by variables (columns), along with a class vector of integers 1:c, each integer specifying the class. The class vector has the same number of cases as the data. The function outputs the position of the 2 bin boundaries (3 bins) that optimize the mutual information of each variable's data vector with the class vector.

Usage
opt3bin(data, class)

Arguments

data	double array of continuous values, cases in rows and variables in columns. Distribution is unknown
class	double column vector, values 1:c representing classification of each case

Value

l	double row vector of left boundary position for each var
r	double row vector of right boundary position for each var
binned	double data array discretized using boundaries in l and r
mi	double row vector of mutual info between each discr. variable and class

Author(s)
Karl Kuschner, Qian Si and William Cooke, College of William and Mary, Dept. of Physics, 2009

References
http://kwkusc.people.wm.edu/dissertation/dissertation.htm

Examples
result <- opt3bin(data,class)

3.9 looklr - Find Boundary

Description
given a start position, finds another boundary (to create 3 bins) that maximizes MI with the class

Usage
looklr(data, class, startbd, steps)

Arguments

data	double array, cases in rows and variables in columns
class	double column vector, values 1:c representing classification of each case
startbd	double vector, given start position for each case
steps	Integer, number of steps to test at while finding maximum MI

Value

miout	double vector, recorded highest MI value
nextboundary	double vector, boundary (to create 3 bins) that maximizes MI with the class
binned	double matrix, recorded the binned value.

Author(s)
Karl Kuschner, Qian Si and William Cooke, College of William and Mary, Dept. of Physics, 2009

References
http://kwkusc.people.wm.edu/dissertation/dissertation.htm

Examples
result<- looklr (data, class, startbd, steps)

3.10 TestCases - Classify Using Bayes Rule

Description

Tests each of a set of data vectors by looking up P(data|class) in a probability table, then finding P(case|class) by multiplying each of those values in a product. Then uses Bayes' rule to calculate P(class|data) for each possible value of class. Reports this as an array of class probabilities for each case.

Usage

TestCases(p, prior, data)

Arguments

p	3-D double array of probabilities (c,d,v). The first dimension is the class, the second is the data value, and the third is the variable number. The entry is P(var v=value d | class=value c)
prior	double columns vector, prior probabilities for each cases in data
data	double array of discrete integer (1:n) values, cases in rows and variables in columns

Value

2-D double array whose value is P (class=c|data) for each case. Cases are in rows, class in cols

Author(s)

Karl Kuschner, Qian Si and William Cooke, College of William and Mary, Dept. of Physics, 2009

References

http://kwkusc.people.wm.edu/dissertation/dissertation.htm

Examples

```
classprobs <- TestCases( p, prior, data)
```

4. References

[1] Tuszynski, Jarek. Protein Mass Spectra (SELDI) Data Processing and Classification with “caMassClass” library, accessed on Aug 21, 2009 from http://rss.acs.unt.edu/Rdoc/library/caMassClass/ doc/caMassClass-old_manual.pdf.

[2] Cover, Thomas M. and Thomas, Joy A. Elements of Information Theory. s.l. : John Wiley & Sons, Inc, 1991. 0-471-06259-6.

[3] Jensen, Finn V. and Nielson, Thomas D. Bayesian Networks and Decision Graphs. New York, NY : Springer, 2007. ISBN 0-387-68281-3.

[4] Kohavi, Ron. A Study of Cross Validation and Bootstrap for Accuracy Estimation and Model Selection. Stanford, 1995.

[5] Kuschner, Karl. A Bayesian Network Approach to Feature Selection in Mass Spectrometry Data. Accessed on Aug 21, 2009 from http://kwkusc.people.wm.edu/dissertation/dissertation.htm.

5. Acknowledgement

This research was supported by NIH computational R01 grant CA126118 from the Advanced Proteomics Platforms and Computational Sciences Program within the Clinical Proteomics Initiative of the National Cancer Institute (PI: Malyarenko). We would like to acknowledge multiple discussions with Prof. Gene Tracy, and C-code contribution and performance optimization by Prof. William Cooke.