THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

Bayesian Network Classifiers in Weka for Version 3-5-7

Remco R. Bouckaert
remco@cs.waikato.ac.nz

February 25, 2008

Abstract

Various Bayesian network classifier learning algorithms are implemented in Weka [10]. This note provides some user documentation and implementation details.

Summary of main capabilities:

- Structure learning of Bayesian networks using various hill climbing (K2, B, etc) and general purpose (simulated annealing, tabu search) algorithms.
- Local score metrics implemented; Bayes, BDe, MDL, entropy, AIC.
- Global score metrics implemented; leave one out cv, k-fold cv and cumulative cv.
- Conditional independence based causal recovery algorithm available.
- Parameter estimation using direct estimates and Bayesian model averaging.
- GUI for easy inspection of Bayesian networks.
- Part of Weka allowing systematic experiments to compare Bayes net performance with general purpose classifiers like C4.5, nearest neighbor, support vector, etc.
- Source code available under GPL[1] allows for integration in other open-source systems and makes it easy to extend.

[1]GPL: GNU General Public License. For more information see the GNU homepage http://www.gnu.org/copyleft/gpl.html.

Contents

1 Introduction **3**

2 Local score based structure learning **6**

3 Conditional independence test based structure learning **11**

4 Global score metric based structure learning **13**

5 Fixed structure 'learning' **14**

6 Distribution learning **14**

7 Running from the command line **16**

8 Inspecting Bayesian networks **26**

9 Bayesian nets in the experimenter **29**

10 Adding your own Bayesian network learners **30**

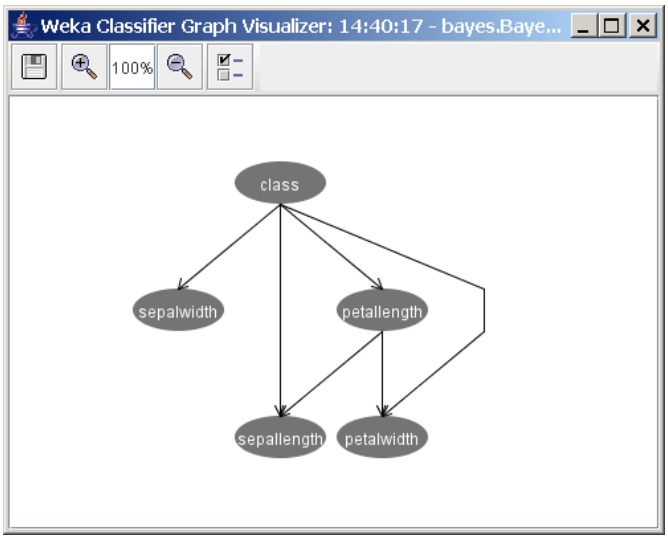
11 FAQ **31**

12 Future development **33**

1 Introduction

Let $U = \{x_1, \ldots, x_n\}$, $n \geq 1$ be a set of variables. A *Bayesian network* B over a set of variables U is a *network structure* B_S, which is a directed acyclic graph (DAG) over U and a set of probability tables $B_P = \{p(u|pa(u))|u \in U\}$ where $pa(u)$ is the set of parents of u in B_S. A Bayesian network represents a probability distributions $P(U) = \prod_{u \in U} p(u|pa(u))$.

Below, a Bayesian network is shown for the variables in the iris data set. Note that the links between the nodes class, petallength and petalwidth do not form a *directed* cycle, so the graph is a proper DAG.



This picture just shows the network structure of the Bayes net, but for each of the nodes a probability distribution for the node given its parents are specified as well. For example, in the Bayes net above there is a conditional distribution for petallength given the value of class. Since class has no parents, there is an unconditional distribution for sepalwidth.

Basic assumptions

The classification task consist of classifying a variable $y = x_0$ called the *class variable* given a set of variables $\mathbf{x} = x_1 \ldots x_n$, called *attribute variables*. A classifier $h : \mathbf{x} \rightarrow y$ is a function that maps an instance of $\mathbf{x}$ to a value of y. The classifier is learned from a dataset D consisting of samples over $(\mathbf{x}, y)$. The learning task consists of finding an appropriate Bayesian network given a data set D over U.

All Bayes network algorithms implemented in Weka assume the following for the data set:

- all variables are discrete finite variables. If you have a data set with continuous variables, you can use the following filter to discretize them: `weka.filters.unsupervised.attribute.Discretize`

- no instances have missing values. If there are missing values in the data set, values are filled in using the following filter:
  `weka.filters.unsupervised.attribute.ReplaceMissingValues`

The first step performed by `buildClassifier` is checking if the data set fulfills those assumptions. If those assumptions are not met, the data set is automatically filtered and a warning is written to STDERR.[2]

Inference algorithm

To use a Bayesian network as a classifier, one simply calculates $argmax_y P(y|\mathbf{x})$ using the distribution $P(U)$ represented by the Bayesian network. Now note that

$$\begin{aligned} P(y|\mathbf{x}) &= P(U)/P(\mathbf{x}) \\ &\propto P(U) \\ &= \prod_{u \in U} p(u|pa(u)) \end{aligned} \tag{1}$$

And since all variables in $\mathbf{x}$ are known, we do not need complicated inference algorithms, but just calculate (1) for all class values.

Learning algorithms

The dual nature of a Bayesian network makes learning a Bayesian network as a two stage process a natural division: first learn a network structure, then learn the probability tables.

There are various approaches to structure learning and in Weka, the following areas are distinguished:

- *local score metrics*: Learning a network structure B_S can be considered an optimization problem where a quality measure of a network structure given the training data $Q(B_S|D)$ needs to be maximized. The quality measure can be based on a Bayesian approach, minimum description length, information and other criteria. Those metrics have the practical property that the score of the whole network can be decomposed as the sum (or product) of the score of the individual nodes. This allows for local scoring and thus local search methods.

- *conditional independence tests*: These methods mainly stem from the goal of uncovering causal structure. The assumption is that there is a network structure that exactly represents the independencies in the distribution that generated the data. Then it follows that if a (conditional) independency can be identified in the data between two variables that there is no arrow between those two variables. Once locations of edges are identified, the direction of the edges is assigned such that conditional independencies in the data are properly represented.

[2] If there are missing values in the test data, but not in the training data, the values are filled in in the test data with a ReplaceMissingValues filter based on the training data.

- *global score metrics*: A natural way to measure how well a Bayesian network performs on a given data set is to predict its future performance by estimating expected utilities, such as classification accuracy. Cross-validation provides an out of sample evaluation method to facilitate this by repeatedly splitting the data in training and validation sets. A Bayesian network structure can be evaluated by estimating the network's parameters from the training set and the resulting Bayesian network's performance determined against the validation set. The average performance of the Bayesian network over the validation sets provides a metric for the quality of the network.

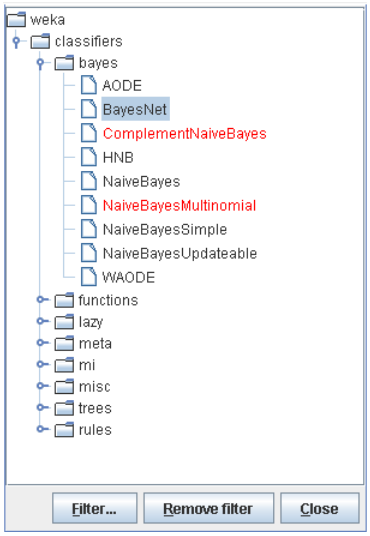
  Cross-validation differs from local scoring metrics in that the quality of a network structure often cannot be decomposed in the scores of the individual nodes. So, the whole network needs to be considered in order to determine the score.

- *fixed structure*: Finally, there are a few methods so that a structure can be fixed, for example, by reading it from an XML BIF file[3].

For each of these areas, different search algorithms are implemented in Weka, such as hill climbing, simulated annealing and tabu search.

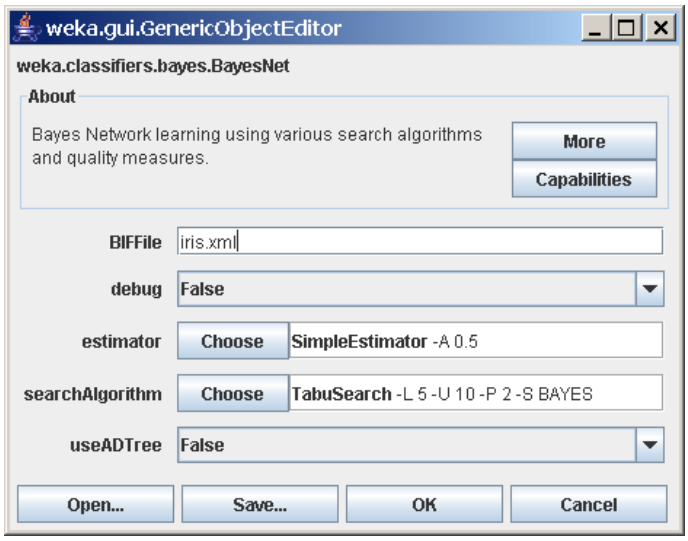
Once a good network structure is identified, the conditional probability tables for each of the variables can be estimated.

You can select a Bayes net classifier by clicking the classifier 'Choose' button in the Weka explorer, experimenter or knowledge flow and find `BayesNet` under the `weka.classifiers.bayes` package (see below).



The Bayes net classifier has the following options:

[3]See http://www-2.cs.cmu.edu/~fgcozman/Research/InterchangeFormat/ for details on XML BIF.



The `BIFFile` option can be used to specify a Bayes network stored in file in BIF format. When the `toString()` method is called after learning the Bayes network, extra statistics (like extra and missing arcs) are printed comparing the network learned with the one on file.

The `searchAlgorithm` option can be used to select a structure learning algorithm and specify its options.

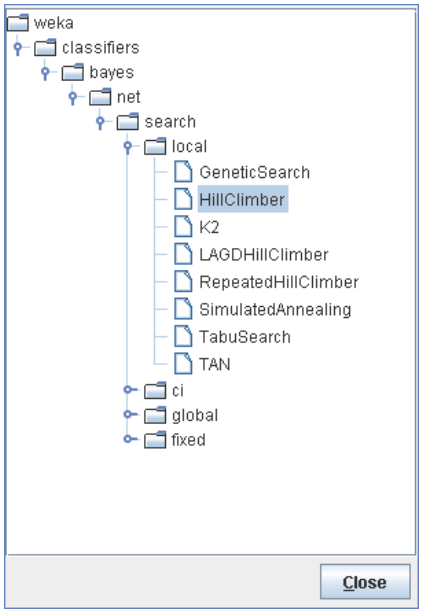
The `estimator` option can be used to select the method for estimating the conditional probability distributions (Section 6).

When setting the `useADTree` option to **`true`**, counts are calculated using the ADTree algorithm of Moore [8]. Since I have not noticed a lot of improvement for small data sets, it is set off by default. Note that this ADTree algorithm is different from the ADTree classifier algorithm from `weka.classifiers.tree.ADTree`.

The `debug` option has no effect.

2 Local score based structure learning

Distinguish score metrics (Section 2.1) and search algorithms (Section 2.2). A local score based structure learning can be selected by choosing one in the `weka.classifiers.bayes.net.search.local` package.



Local score based algorithms have the following options in common:
`initAsNaiveBayes` if set **true** (default), the initial network structure used for starting the traversal of the search space is a naive Bayes network structure. That is, a structure with arrows from the class variable to each of the attribute variables.
If set **false**, an empty network structure will be used (i.e., no arrows at all).
`markovBlanketClassifier` (**false** by default) if set **true**, at the end of the traversal of the search space, a heuristic is used to ensure each of the attributes are in the Markov blanket of the classifier node. If a node is already in the Markov blanket (i.e., is a parent, child of sibling of the classifier node) nothing happens, otherwise an arrow is added.
If set to **false** no such arrows are added.
`scoreType` determines the score metric used (see Section 2.1 for details). Currently, K2, BDe, AIC, Entropy and MDL are implemented.
`maxNrOfParents` is an upper bound on the number of parents of each of the nodes in the network structure learned.

2.1 Local score metrics

We use the following conventions to identify counts in the database D and a network structure B_S. Let r_i $(1 \leq i \leq n)$ be the cardinality of x_i. We use q_i to denote the cardinality of the parent set of x_i in B_S, that is, the number of different values to which the parents of x_i can be instantiated. So, q_i can be calculated as the product of cardinalities of nodes in $pa(x_i)$, $q_i = \prod_{x_j \in pa(x_i)} r_j$. Note $pa(x_i) = \emptyset$ implies $q_i = 1$. We use N_{ij} $(1 \leq i \leq n,\ 1 \leq j \leq q_i)$ to denote the number of records in D for which $pa(x_i)$ takes its jth value.We use N_{ijk} $(1 \leq i \leq n,\ 1 \leq j \leq q_i,\ 1 \leq k \leq r_i)$ to denote the number of records in D for which $pa(x_i)$ takes its jth value and for which x_i takes its kth value. So, $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. We use N to denote the number of records in D.

Let the *entropy metric* $H(B_S, D)$ of a network structure and database be defined as

$$H(B_S, D) = -N \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \frac{N_{ijk}}{N} \log \frac{N_{ijk}}{N_{ij}} \tag{2}$$

and the *number of parameters* K as

$$K = \sum_{i=1}^{n} (r_i - 1) \cdot q_i \tag{3}$$

AIC metric The AIC metric $Q_{AIC}(B_S, D)$ of a Bayesian network structure B_S for a database D is

$$Q_{AIC}(B_S, D) = H(B_S, D) + K \tag{4}$$

A term $P(B_S)$ can be added [1] representing prior information over network structures, but will be ignored for simplicity in the Weka implementation.

MDL metric The minimum description length metric $Q_{MDL}(B_S, D)$ of a Bayesian network structure B_S for a database D is is defined as

$$Q_{MDL}(B_S, D) = H(B_S, D) + \frac{K}{2} \log N \tag{5}$$

Bayesian metric The Bayesian metric of a Bayesian network structure B_D for a database D is

$$Q_{Bayes}(B_S, D) = P(B_S) \prod_{i=0}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk} + N_{ijk})}{\Gamma(N'_{ijk})}$$

where $P(B_S)$ is the prior on the network structure (taken to be constant hence ignored in the Weka implementation) and $\Gamma(.)$ the gamma-function. N'_{ij} and N'_{ijk} represent choices of priors on counts restricted by $N'_{ij} = \sum_{k=1}^{r_i} N'_{ijk}$. With $N'_{ijk} = 1$ (and thus $N'_{ij} = r_i$), we obtain the K2 metric [5]

$$Q_{K2}(B_S, D) = P(B_S) \prod_{i=0}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(r_i - 1 + N_{ij})!} \prod_{k=1}^{r_i} N_{ijk}!$$

With $N'_{ijk} = 1/r_i \cdot q_i$ (and thus $N'_{ij} = 1/q_i$), we obtain the **BDe metric** [7].

2.2 Search algorithms

The following search algorithms are implemented for local score metrics;

- *K2* [5]: hill climbing add arcs with a fixed ordering of variables.
  Specific option: `randomOrder` if **`true`** a random ordering of the nodes is made at the beginning of the search. If **false** (default) the ordering in the data set is used. The only exception in both cases is that in case the initial network is a naive Bayes network (`initAsNaiveBayes` set **`true`**) the class variable is made first in the ordering.

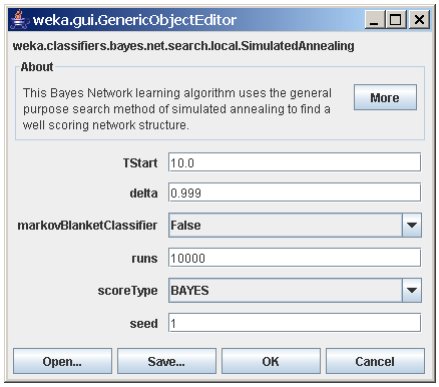
- *Hill Climbing* [2]: hill climbing adding and deleting arcs with no fixed ordering of variables.
  `useArcReversal` if **`true`**, also arc reversals are consider when determining the next step to make.

- *Repeated Hill Climber* starts with a randomly generated network and then applies hill climber to reach a local optimum. The best network found is returned.
  `useArcReversal` option as for Hill Climber.

- *LAGD Hill Climbing* does hill climbing with look ahead on a limited set of best scoring steps, implemented by Manuel Neubach. The number of look ahead steps and number of steps considered for look ahead are configurable.

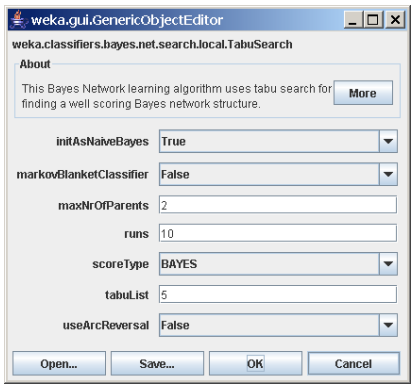
- *TAN* [3, 6]: *T*ree *A*ugmented *N*aive Bayes where the tree is formed by calculating the maximum weight spanning tree using Chow and Liu algorithm [4].
  No specific options.

- *Simulated annealing* [1]: using adding and deleting arrows.
  The algorithm randomly generates a candidate network B'_S close to the current network B_S. It accepts the network if it is better than the current, i.e., $Q(B'_S, D) > Q(B_S, D)$. Otherwise, it accepts the candidate with probability
  $$e^{t_i \cdot (Q(B'_S, D) - Q(B_S, D))}$$
  where t_i is the temperature at iteration i. The temperature starts at t_0 and is slowly decreases with each iteration.



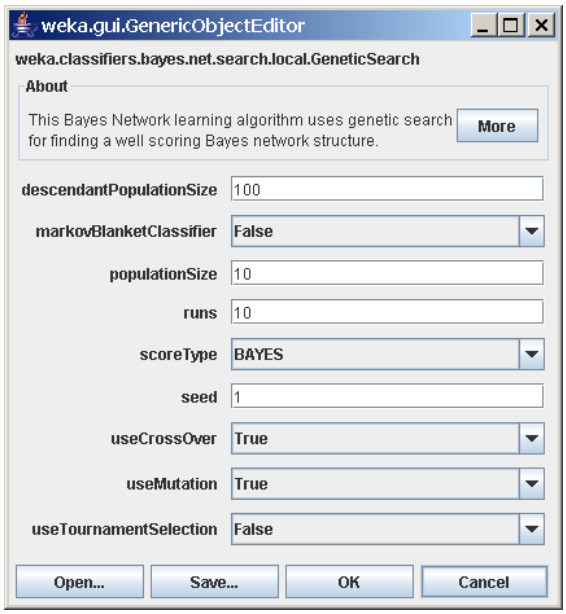
  Specific options:
  `TStart` start temperature t_0.
  `delta` is the factor δ used to update the temperature, so $t_{i+1} = t_i \cdot \delta$.
  `runs` number of iterations used to traverse the search space.
  `seed` is the initialization value for the random number generator.

- *Tabu search* [1]: using adding and deleting arrows.
  Tabu search performs hill climbing until it hits a local optimum. Then it steps to the least worse candidate in the neighborhood. However, it does not consider points in the neighborhood it just visited in the last *tl* steps. These steps are stored in a so called tabu-list.



Specific options:
`runs` is the number of iterations used to traverse the search space.
`tabuList` is the length tl of the tabu list.

- *Genetic search*: applies a simple implementation of a genetic search algorithm to network structure learning. A Bayes net structure is represented by a array of $n \cdot n$ (n = number of nodes) bits where bit $i \cdot n + j$ represents whether there is an arrow from node $j \rightarrow i$.

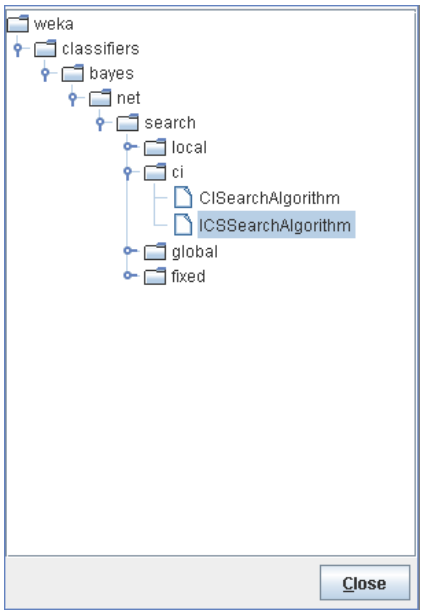


Specific options:
`populationSize` is the size of the population selected in each generation.
`descendantPopulationSize` is the number of offspring generated in each generation.
`runs` is the number of generation to generate.
`seed` is the initialization value for the random number generator.
`useMutation` flag to indicate whether mutation should be used. Mutation

is applied by randomly adding or deleting a single arc.
`useCrossOver` flag to indicate whether cross-over should be used. Cross-over is applied by randomly picking an index k in the bit representation and selecting the first k bits from one and the remainder from another network structure in the population. At least one of `useMutation` and `useCrossOver` should be set to **true**.
`useTournamentSelection` when **false**, the best performing networks are selected from the descendant population to form the population of the next generation. When **true**, tournament selection is used. Tournament selection randomly chooses two individuals from the descendant population and selects the one that performs best.

3 Conditional independence test based structure learning

Conditional independence tests in Weka are slightly different from the standard tests described in the literature. To test whether variables x and y are conditionally independent given a set of variables Z, a network structure with arrows $\forall_{z \in Z} z \rightarrow y$ is compared with one with arrows $\{x \rightarrow y\} \cup \forall_{z \in Z} z \rightarrow y$. A test is performed by using any of the score metrics described in Section 2.1.



At the moment, only the ICS [9]and CI algorithm are implemented.

The ICS algorithm makes two steps, first find a skeleton (the undirected graph with edges iff there is an arrow in network structure) and second direct all the edges in the skeleton to get a DAG.

Starting with a complete undirected graph, we try to find conditional independencies $\langle x, y | Z \rangle$ in the data. For each pair of nodes x, y, we consider sets

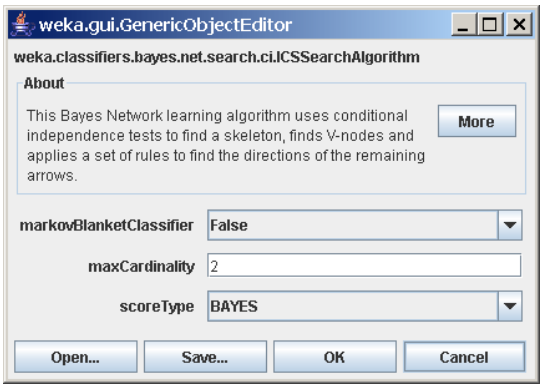
Z starting with cardinality 0, then 1 up to a user defined maximum. Furthermore, the set Z is a subset of nodes that are neighbors of both x and y. If an independency is identified, the edge between x and y is removed from the skeleton.

The first step in directing arrows is to check for every configuration $x - -z - -y$ where x and y not connected in the skeleton whether z is in the set Z of variables that justified removing the link between x and y (cached in the first step). If z is not in Z, we can assign direction $x \rightarrow z \leftarrow y$.

Finally, a set of graphical rules is applied [9] to direct the remaining arrows.

```
Rule 1: i->j--k & i-/-k => j->k
Rule 2: i->j->k & i--k => i->k
Rule 3  m
                /|\
               i | k  => m->j
     i->j<-k    \|/
                 j

Rule 4  m
                / \
               i---k  => i->m & k->m
       i->j     \ /
                 j
Rule 5: if no edges are directed then take a random one (first we can find)
```

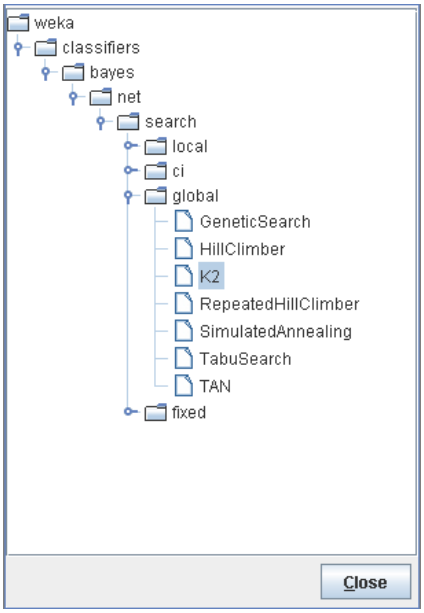
The ICS algorithm comes with the following options.



Since the ICS algorithm is focused on recovering causal structure, instead of finding the optimal classifier, the Markov blanket correction can be made afterwards.

Specific options:
The `maxCardinality` option determines the largest subset of Z to be considered in conditional independence tests $\langle x, y|Z\rangle$.
The `scoreType` option is used to select the scoring metric.

4 Global score metric based structure learning

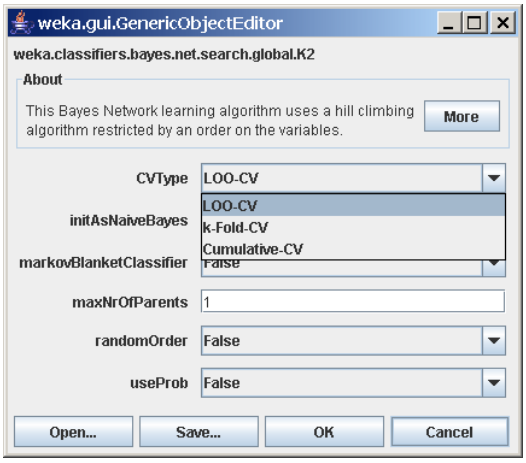


Common options for cross-validation based algorithms are:
`initAsNaiveBayes`, `markovBlanketClassifier` and `maxNrOfParents` (see Section 2 for description).

Further, for each of the cross-validation based algorithms the `CVType` can be chosen out of the following:

- *Leave one out cross-validation (loo-cv)* selects $m = N$ training sets simply by taking the data set D and removing the ith record for training set D_i^t. The validation set consist of just the ith single record. Loo-cv does not always produce accurate performance estimates.
- *K-fold cross-validation (k-fold cv)* splits the data D in m approximately equal parts $D_1, \ldots, D_m$. Training set D_i^t is obtained by removing part D_i from D. Typical values for m are 5, 10 and 20. With $m = N$, k-fold cross-validation becomes loo-cv.
- *Cumulative cross-validation (cumulative cv)* starts with an empty data set and adds instances item by item from D. After each time an item is added the next item to be added is classified using the then current state of the Bayes network.

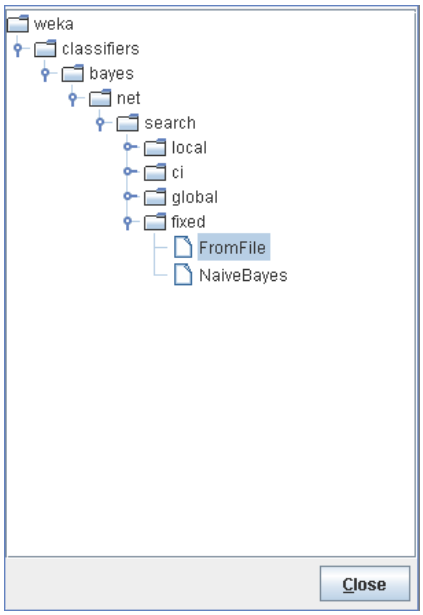
Finally, the `useProb` flag indicates whether the accuracy of the classifier should be estimated using the zero-one loss (if set to **false**) or using the estimated probability of the class.



The following search algorithms are implemented: K2, HillClimbing, RepeatedHillClimber, TAN, Tabu Search, Simulated Annealing and Genetic Search. See Section 2 for a description of the specific options for those algorithms.

5 Fixed structure 'learning'

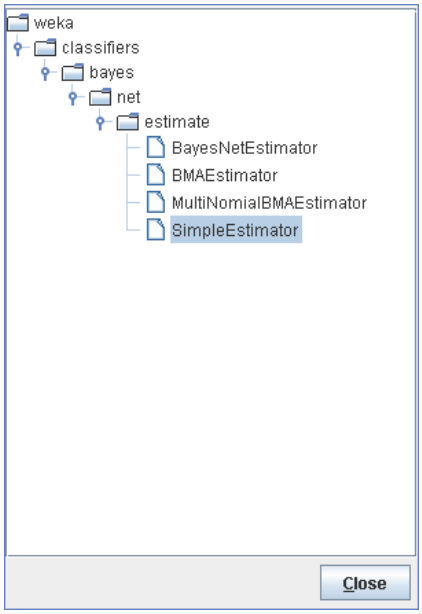
The structure learning step can be skipped by selecting a fixed network structure. There are two methods of getting a fixed structure: just make it a naive Bayes network, or reading it from a file in XML BIF format.



6 Distribution learning

Once the network structure is learned, you can choose how to learn the probability tables selecting a class in the `weka.classifiers.bayes.net.estimate`

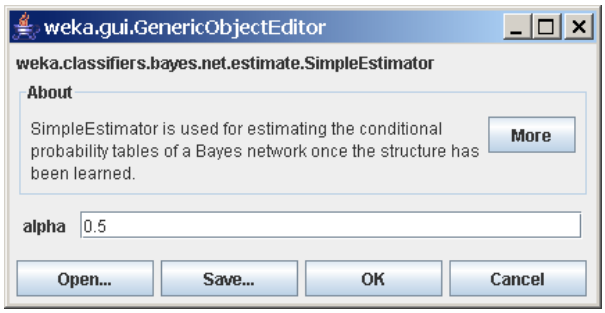
package.



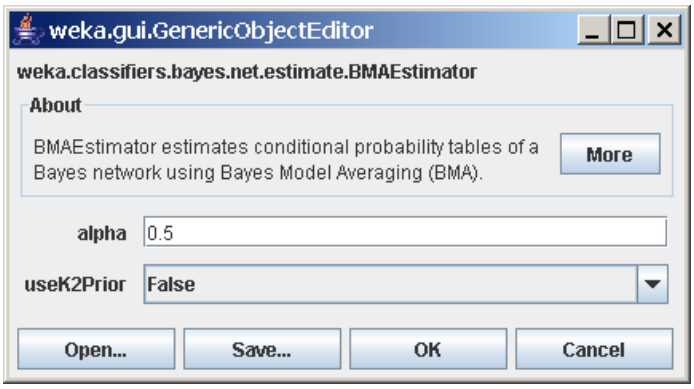
The `SimpleEstimator` class produces direct estimates of the conditional probabilities, that is,

$$P(x_i = k|pa(x_i) = j) = \frac{N_{ijk} + N'_{ijk}}{N_{ij} + N'_{ij}}$$

where N'_{ijk} is the alpha parameter that can be set and is 0.5 by default. With $alpha = 0$, we get maximum likelihood estimates.



With the `BMAEstimator`, we get estimates for the conditional probability tables based on Bayes model averaging of all network structures that are substructures of the network structure learned [1]. This is achieved by estimating the conditional probability table of a node x_i given its parents $pa(x_i)$ as a weighted average of all conditional probability tables of x_i given subsets of $pa(x_i)$. The weight of a distribution $P(x_i|S)$ with $S \subseteq pa(x_i)$ used is proportional to the contribution of network structure $\forall_{y \in S} y \rightarrow x_i$ to either the BDe metric or K2 metric depending on the setting of the `useK2Prior` option (**false** and **true** respectively).



7 Running from the command line

These are the command line options of `BayesNet`.

```
General options:

-t <name of training file>
        Sets training file.
-T <name of test file>
        Sets test file. If missing, a cross-validation will be performed on the
        training data.
-c <class index>
        Sets index of class attribute (default: last).
-x <number of folds>
        Sets number of folds for cross-validation (default: 10).
-no-cv
        Do not perform any cross validation.
-split-percentage <percentage>
        Sets the percentage for the train/test set split, e.g., 66.
-preserve-order
        Preserves the order in the percentage split.
-s <random number seed>
        Sets random number seed for cross-validation or percentage split
        (default: 1).
-m <name of file with cost matrix>
        Sets file with cost matrix.
-l <name of input file>
        Sets model input file. In case the filename ends with '.xml',
        the options are loaded from the XML file.
-d <name of output file>
        Sets model output file. In case the filename ends with '.xml',
        only the options are saved to the XML file, not the model.
-v
        Outputs no statistics for training data.
-o
        Outputs statistics only, not the classifier.
-i
        Outputs detailed information-retrieval statistics for each class.
-k
```

```
        Outputs information-theoretic statistics.
-p <attribute range>
        Only outputs predictions for test instances (or the train
        instances if no test instances provided), along with attributes
        (0 for none).
-distribution
        Outputs the distribution instead of only the prediction
        in conjunction with the '-p' option (only nominal classes).
-r
        Only outputs cumulative margin distribution.
-g
        Only outputs the graph representation of the classifier.
-xml filename | xml-string
        Retrieves the options from the XML-data instead of the command line.

Options specific to weka.classifiers.bayes.BayesNet:

-D
        Do not use ADTree data structure

-B <BIF file>
        BIF file to compare with

-Q weka.classifiers.bayes.net.search.SearchAlgorithm
        Search algorithm

-E weka.classifiers.bayes.net.estimate.SimpleEstimator
        Estimator algorithm
```

The search algorithm option -Q and estimator option -E options are mandatory.

Note that it is important that the -E options should be used after the -Q option. Extra options can be passed to the search algorithm and the estimator after the class name specified following '--'.
For example:

```
java weka.classifiers.bayes.BayesNet -t iris.arff -D \
  -Q weka.classifiers.bayes.net.search.local.K2 -- -P 2 -S ENTROPY \
  -E weka.classifiers.bayes.net.estimate.SimpleEstimator -- -A 1.0
```

Overview of options for search algorithms

- `weka.classifiers.bayes.net.search.local.GeneticSearch`

```
  -L <integer>
          Population size
  -A <integer>
          Descendant population size
  -U <integer>
          Number of runs
  -M
          Use mutation.
```

```
        (default true)
-C
        Use cross-over.
        (default true)
-O
        Use tournament selection (true) or maximum subpopulatin (false).
        (default false)
-R <seed>
        Random number seed
-mbc
        Applies a Markov Blanket correction to the network structure,
        after a network structure is learned. This ensures that all
        nodes in the network are part of the Markov blanket of the
        classifier node.
-S [BAYES|MDL|ENTROPY|AIC|CROSS_CLASSIC|CROSS_BAYES]
        Score type (BAYES, BDeu, MDL, ENTROPY and AIC)
```

- weka.classifiers.bayes.net.search.local.HillClimber

```
-P <nr of parents>
        Maximum number of parents
-R
        Use arc reversal operation.
        (default false)
-N
        Initial structure is empty (instead of Naive Bayes)
-mbc
        Applies a Markov Blanket correction to the network structure,
        after a network structure is learned. This ensures that all
        nodes in the network are part of the Markov blanket of the
        classifier node.
-S [BAYES|MDL|ENTROPY|AIC|CROSS_CLASSIC|CROSS_BAYES]
        Score type (BAYES, BDeu, MDL, ENTROPY and AIC)
```

- weka.classifiers.bayes.net.search.local.K2

```
-N
        Initial structure is empty (instead of Naive Bayes)
-P <nr of parents>
        Maximum number of parents
-R
        Random order.
        (default false)
-mbc
        Applies a Markov Blanket correction to the network structure,
        after a network structure is learned. This ensures that all
        nodes in the network are part of the Markov blanket of the
        classifier node.
```

```
-S [BAYES|MDL|ENTROPY|AIC|CROSS_CLASSIC|CROSS_BAYES]
        Score type (BAYES, BDeu, MDL, ENTROPY and AIC)
```

- weka.classifiers.bayes.net.search.local.LAGDHillClimber

```
-L <nr of look ahead steps>
        Look Ahead Depth
-G <nr of good operations>
        Nr of Good Operations
-P <nr of parents>
        Maximum number of parents
-R
        Use arc reversal operation.
        (default false)
-N
        Initial structure is empty (instead of Naive Bayes)
-mbc
        Applies a Markov Blanket correction to the network structure,
        after a network structure is learned. This ensures that all
        nodes in the network are part of the Markov blanket of the
        classifier node.
-S [BAYES|MDL|ENTROPY|AIC|CROSS_CLASSIC|CROSS_BAYES]
        Score type (BAYES, BDeu, MDL, ENTROPY and AIC)
```

- weka.classifiers.bayes.net.search.local.RepeatedHillClimber

```
-U <integer>
        Number of runs
-A <seed>
        Random number seed
-P <nr of parents>
        Maximum number of parents
-R
        Use arc reversal operation.
        (default false)
-N
        Initial structure is empty (instead of Naive Bayes)
-mbc
        Applies a Markov Blanket correction to the network structure,
        after a network structure is learned. This ensures that all
        nodes in the network are part of the Markov blanket of the
        classifier node.
-S [BAYES|MDL|ENTROPY|AIC|CROSS_CLASSIC|CROSS_BAYES]
        Score type (BAYES, BDeu, MDL, ENTROPY and AIC)
```

- weka.classifiers.bayes.net.search.local.SimulatedAnnealing

```
-A <float>
        Start temperature
-U <integer>
        Number of runs
-D <float>
        Delta temperature
-R <seed>
        Random number seed
-mbc
        Applies a Markov Blanket correction to the network structure,
        after a network structure is learned. This ensures that all
        nodes in the network are part of the Markov blanket of the
        classifier node.
-S [BAYES|MDL|ENTROPY|AIC|CROSS_CLASSIC|CROSS_BAYES]
        Score type (BAYES, BDeu, MDL, ENTROPY and AIC)
```

- weka.classifiers.bayes.net.search.local.TabuSearch

```
-L <integer>
        Tabu list length
-U <integer>
        Number of runs
-P <nr of parents>
        Maximum number of parents
-R
        Use arc reversal operation.
        (default false)
-P <nr of parents>
        Maximum number of parents
-R
        Use arc reversal operation.
        (default false)
-N
        Initial structure is empty (instead of Naive Bayes)
-mbc
        Applies a Markov Blanket correction to the network structure,
        after a network structure is learned. This ensures that all
        nodes in the network are part of the Markov blanket of the
        classifier node.
-S [BAYES|MDL|ENTROPY|AIC|CROSS_CLASSIC|CROSS_BAYES]
        Score type (BAYES, BDeu, MDL, ENTROPY and AIC)
```

- weka.classifiers.bayes.net.search.local.TAN

```
-mbc
        Applies a Markov Blanket correction to the network structure,
        after a network structure is learned. This ensures that all
        nodes in the network are part of the Markov blanket of the
```

```
        classifier node.
-S [BAYES|MDL|ENTROPY|AIC|CROSS_CLASSIC|CROSS_BAYES]
        Score type (BAYES, BDeu, MDL, ENTROPY and AIC)
```

- weka.classifiers.bayes.net.search.ci.CISearchAlgorithm

```
-mbc
        Applies a Markov Blanket correction to the network structure,
        after a network structure is learned. This ensures that all
        nodes in the network are part of the Markov blanket of the
        classifier node.
-S [BAYES|MDL|ENTROPY|AIC|CROSS_CLASSIC|CROSS_BAYES]
        Score type (BAYES, BDeu, MDL, ENTROPY and AIC)
```

- weka.classifiers.bayes.net.search.ci.ICSSearchAlgorithm

```
-cardinality <num>
        When determining whether an edge exists a search is performed
        for a set Z that separates the nodes. MaxCardinality determines
        the maximum size of the set Z. This greatly influences the
        length of the search. (default 2)
-mbc
        Applies a Markov Blanket correction to the network structure,
        after a network structure is learned. This ensures that all
        nodes in the network are part of the Markov blanket of the
        classifier node.
-S [BAYES|MDL|ENTROPY|AIC|CROSS_CLASSIC|CROSS_BAYES]
        Score type (BAYES, BDeu, MDL, ENTROPY and AIC)
```

- weka.classifiers.bayes.net.search.global.GeneticSearch

```
-L <integer>
        Population size
-A <integer>
        Descendant population size
-U <integer>
        Number of runs
-M
        Use mutation.
        (default true)
-C
        Use cross-over.
        (default true)
-O
        Use tournament selection (true) or maximum subpopulatin (false).
        (default false)
-R <seed>
```

```
        Random number seed
-mbc
        Applies a Markov Blanket correction to the network structure,
        after a network structure is learned. This ensures that all
        nodes in the network are part of the Markov blanket of the
        classifier node.
-S [LOO-CV|k-Fold-CV|Cumulative-CV]
        Score type (LOO-CV,k-Fold-CV,Cumulative-CV)
-Q
        Use probabilistic or 0/1 scoring.
        (default probabilistic scoring)
```

- weka.classifiers.bayes.net.search.global.HillClimber

```
-P <nr of parents>
        Maximum number of parents
-R
        Use arc reversal operation.
        (default false)
-N
        Initial structure is empty (instead of Naive Bayes)
-mbc
        Applies a Markov Blanket correction to the network structure,
        after a network structure is learned. This ensures that all
        nodes in the network are part of the Markov blanket of the
        classifier node.
-S [LOO-CV|k-Fold-CV|Cumulative-CV]
        Score type (LOO-CV,k-Fold-CV,Cumulative-CV)
-Q
        Use probabilistic or 0/1 scoring.
        (default probabilistic scoring)
```

- weka.classifiers.bayes.net.search.global.K2

```
-N
        Initial structure is empty (instead of Naive Bayes)
-P <nr of parents>
        Maximum number of parents
-R
        Random order.
        (default false)
-mbc
        Applies a Markov Blanket correction to the network structure,
        after a network structure is learned. This ensures that all
        nodes in the network are part of the Markov blanket of the
        classifier node.
-S [LOO-CV|k-Fold-CV|Cumulative-CV]
        Score type (LOO-CV,k-Fold-CV,Cumulative-CV)
```

```
-Q
        Use probabilistic or 0/1 scoring.
        (default probabilistic scoring)
```

- weka.classifiers.bayes.net.search.global.RepeatedHillClimber

```
-U <integer>
        Number of runs
-A <seed>
        Random number seed
-P <nr of parents>
        Maximum number of parents
-R
        Use arc reversal operation.
        (default false)
-N
        Initial structure is empty (instead of Naive Bayes)
-mbc
        Applies a Markov Blanket correction to the network structure,
        after a network structure is learned. This ensures that all
        nodes in the network are part of the Markov blanket of the
        classifier node.
-S [LOO-CV|k-Fold-CV|Cumulative-CV]
        Score type (LOO-CV,k-Fold-CV,Cumulative-CV)
-Q
        Use probabilistic or 0/1 scoring.
        (default probabilistic scoring)
```

- weka.classifiers.bayes.net.search.global.SimulatedAnnealing

```
-A <float>
        Start temperature
-U <integer>
        Number of runs
-D <float>
        Delta temperature
-R <seed>
        Random number seed
-mbc
        Applies a Markov Blanket correction to the network structure,
        after a network structure is learned. This ensures that all
        nodes in the network are part of the Markov blanket of the
        classifier node.
-S [LOO-CV|k-Fold-CV|Cumulative-CV]
        Score type (LOO-CV,k-Fold-CV,Cumulative-CV)
-Q
        Use probabilistic or 0/1 scoring.
        (default probabilistic scoring)
```

- weka.classifiers.bayes.net.search.global.TabuSearch

```
-L <integer>
        Tabu list length
-U <integer>
        Number of runs
-P <nr of parents>
        Maximum number of parents
-R
        Use arc reversal operation.
        (default false)
-P <nr of parents>
        Maximum number of parents
-R
        Use arc reversal operation.
        (default false)
-N
        Initial structure is empty (instead of Naive Bayes)
-mbc
        Applies a Markov Blanket correction to the network structure,
        after a network structure is learned. This ensures that all
        nodes in the network are part of the Markov blanket of the
        classifier node.
-S [LOO-CV|k-Fold-CV|Cumulative-CV]
        Score type (LOO-CV,k-Fold-CV,Cumulative-CV)
-Q
        Use probabilistic or 0/1 scoring.
        (default probabilistic scoring)
```

- weka.classifiers.bayes.net.search.global.TAN

```
-mbc
        Applies a Markov Blanket correction to the network structure,
        after a network structure is learned. This ensures that all
        nodes in the network are part of the Markov blanket of the
        classifier node.
-S [LOO-CV|k-Fold-CV|Cumulative-CV]
        Score type (LOO-CV,k-Fold-CV,Cumulative-CV)
-Q
        Use probabilistic or 0/1 scoring.
        (default probabilistic scoring)
```

- weka.classifiers.bayes.net.search.fixed.FromFile

```
-B <BIF File>
        Name of file containing network structure in BIF format
```

- weka.classifiers.bayes.net.search.fixed.NaiveBayes

```
No options.
```

Overview of options for estimators

- `weka.classifiers.bayes.net.estimate.BayesNetEstimator`

```
-A <alpha>
        Initial count (alpha)
```

- `weka.classifiers.bayes.net.estimate.BMAEstimator`

```
-k2
        Whether to use K2 prior.
-A <alpha>
        Initial count (alpha)
```

- `weka.classifiers.bayes.net.estimate.MultiNomialBMAEstimator`

```
-k2
        Whether to use K2 prior.
-A <alpha>
        Initial count (alpha)
```

- `weka.classifiers.bayes.net.estimate.SimpleEstimator`

```
-A <alpha>
        Initial count (alpha)
```

Generating random networks and artificial data sets

You can generate random Bayes nets and data sets using
`weka.classifiers.bayes.net.BayesNetGenerator`
The options are:

```
-B
        Generate network (instead of instances)
-N <integer>
        Nr of nodes
-A <integer>
        Nr of arcs
-M <integer>
        Nr of instances
-C <integer>
        Cardinality of the variables
-S <integer>
        Seed for random number generator
-F <file>
        The BIF file to obtain the structure from.
```

The network structure is generated by first generating a tree so that we can ensure that we have a connected graph. If any more arrows are specified they are randomly added.

8 Inspecting Bayesian networks

You can inspect some of the properties of Bayesian networks that you learned in the Explorer in text format and also in graphical format.

Bayesian networks in text

Below, you find output typical for a 10 fold cross-validation run in the Weka Explorer with comments where the output is specific for Bayesian nets.

```
=== Run information ===

Scheme:       weka.classifiers.bayes.BayesNet -D -B iris.xml -Q weka.classifiers.bayes.net
```

Options for `BayesNet` include the class names for the structure learner and for the distribution estimator.

```
Relation:     iris-weka.filters.unsupervised.attribute.Discretize-B2-M-1.0-Rfirst-last
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

Bayes Network Classifier
not using ADTree
```

Indication whether the ADTree algorithm [8] for calculating counts in the data set was used.

```
#attributes=5 #classindex=4
```

This line lists the number of attribute and the number of the class variable for which the classifier was trained.

```
Network structure (nodes followed by parents)
sepallength(2): class
sepalwidth(2): class
petallength(2): class sepallength
petalwidth(2): class petallength
class(3):
```

This list specifies the network structure. Each of the variables is followed by a list of parents, so the *petallength* variable has parents *sepallength* and *class*, while *class* has no parents. The number in braces is the cardinality of the variable. It shows that in the iris dataset there are three class variables. All other variables are made binary by running it through a discretization filter.

```
LogScore Bayes: -374.9942769685747
LogScore BDeu: -351.85811477631626
LogScore MDL: -416.86897021246466
LogScore ENTROPY: -366.76261727150217
LogScore AIC: -386.76261727150217
```

These lines list the logarithmic score of the network structure for various methods of scoring.

If a BIF file was specified, the following two lines will be produced (if no such file was specified, no information is printed).

```
Missing: 0 Extra: 2 Reversed: 0
Divergence: -0.0719759699700729
```

In this case the network that was learned was compared with a file `iris.xml` which contained the naive Bayes network structure. The number after "Missing" is the number of arcs that was in the network in file that is not recovered by the structure learner. Note that a reversed arc is not counted as missing. The number after "Extra" is the number of arcs in the learned network that are not in the network on file. The number of reversed arcs is listed as well.

Finally, the divergence between the network distribution on file and the one learned is reported. This number is calculated by enumerating all possible instantiations of all variables, so it may take some time to calculate the divergence for large networks.

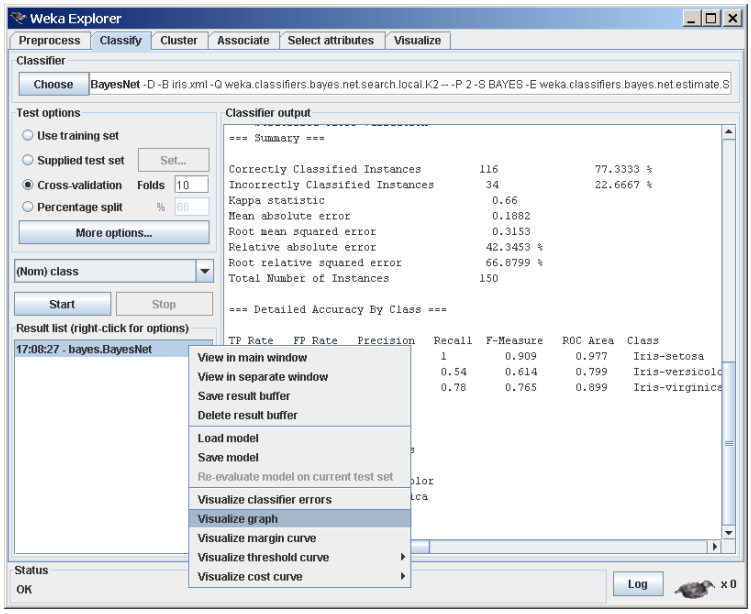
The remainder of the output is standard output for all classifiers.

```
Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         116               77.3333 %
Incorrectly Classified Instances        34               22.6667 %
```

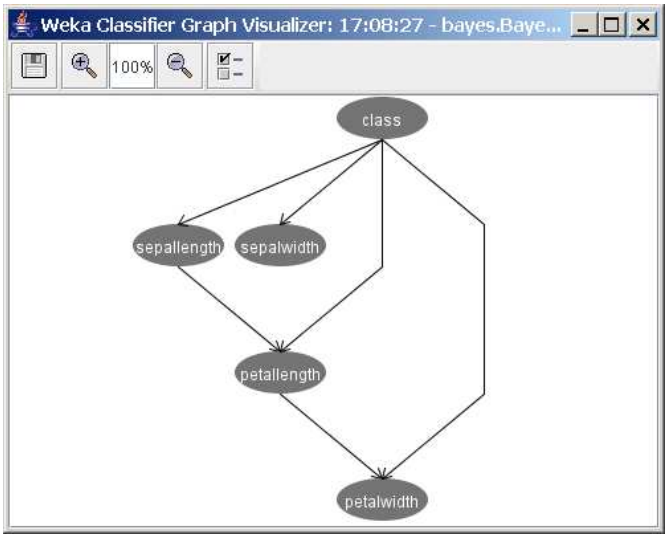
etc...

Bayesian networks in GUI

To show the graphical structure, right click the appropriate `BayesNet` in result list of the Explorer. A menu pops up, in which you select "Visualize graph".



The Bayes network is automatically layed out and drawn thanks to a graph drawing algorithm implemented by Ashraf Kibriya.

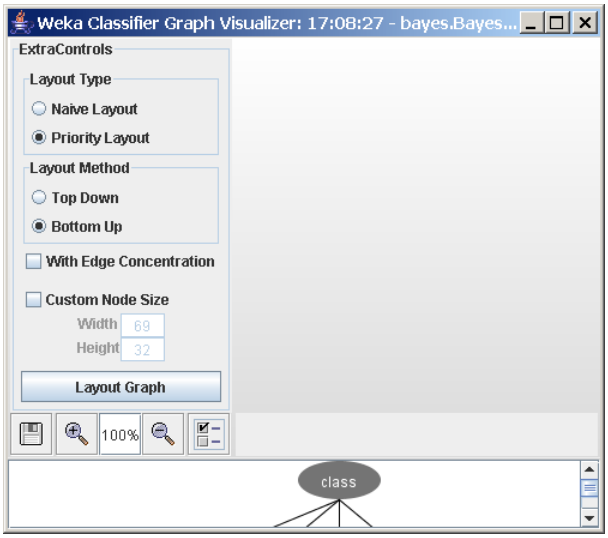


When you hover the mouse over a node, the node lights up and all its children are highlighted as well, so that it is easy to identify the relation between nodes in crowded graphs.

Saving Bayes nets You can save the Bayes network to file in the graph visualizer. You have the choice to save as XML BIF format or as dot format. Select the floppy button and a file save dialog pops up that allows you to select the file name and file format.

Zoom The graph visualizer has two buttons to zoom in and out. Also, the exact zoom desired can be entered in the zoom percentage entry. Hit enter to redraw at the desired zoom level.

Graph drawing options Hit the 'extra controls' button to show extra options that control the graph layout settings.



The `Layout Type` determines the algorithm applied to place the nodes.

The `Layout Method` determines in which direction nodes are considered.

The `Edge Concentration` toggle allows edges to be partially merged.

The `Custom Node Size` can be used to override the automatically determined node size.

When you click a node in the Bayesian net, a window with the probability table of the node clicked pops up. The left side shows the parent attributes and lists the values of the parents, the right side shows the probability of the node clicked conditioned on the values of the parents listed on the left.

Probability Distribution Table For petallength

class	sepallength	'(-inf-3.95]'	'(3.95-inf)'
Iris-setosa	'(-inf-6.1]'	0.99	0.01
Iris-setosa	'(6.1-inf)'	0.5	0.5
Iris-versicolor	'(-inf-6.1]'	0.329	0.671
Iris-versicolor	'(6.1-inf)'	0.029	0.971
Iris-virginica	'(-inf-6.1]'	0.042	0.958
Iris-virginica	'(6.1-inf)'	0.012	0.988

So, the graph visualizer allows you to inspect both network structure and probability tables.

9 Bayesian nets in the experimenter

Bayesian networks generate extra measures that can be examined in the experimenter. The experimenter can then be used to calculate mean and variance for those measures.

The following metrics are generated:

- measureExtraArcs: extra arcs compared to reference network. The network must be provided as BIFFile to the BayesNet class. If no such network is provided, this value is zero.
- measureMissingArcs: missing arcs compared to reference network or zero if not provided.
- measureReversedArcs: reversed arcs compared to reference network or zero if not provided.
- measureDivergence: divergence of network learned compared to reference network or zero if not provided.
- measureBayesScore: log of the K2 score of the network structure.
- measureBDeuScore: log of the BDeu score of the network structure.
- measureMDLScore: log of the MDL score.
- measureAICScore: log of the AIC score.
- measureEntropyScore:log of the entropy.

10 Adding your own Bayesian network learners

You can add your own structure learners and estimators.

Adding a new structure learner

Here is the quick guide for adding a structure learner:

1. Create a class that derives from `weka.classifiers.bayes.net.search.SearchAlgorithm`. If your searcher is score based, conditional independence based or cross-validation based, you probably want to derive from `ScoreSearchAlgorithm`, `CISearchAlgorithm` or `CVSearchAlgorithm` instead of deriving from `SearchAlgorithm` directly. Let's say it is called `weka.classifiers.bayes.net.search.local.MySearcher` derived from `ScoreSearchAlgorithm`.

2. Implement the method `public void buildStructure(BayesNet bayesNet, Instances instances)`. Essentially, you are responsible for setting the parent sets in `bayesNet`. You can access the parentsets using `bayesNet.getParentSet(iAttribute)` where `iAttribute` is the number of the node/variable.

   To add a parent `iParent` to node `iAttribute`, use `bayesNet.getParentSet(iAttribute).AddParent(iParent, instances)` where `instances` need to be passed for the parent set to derive properties of the attribute.

   Alternatively, implement `public void search(BayesNet bayesNet, Instances instances)`. The implementation of `buildStructure` in the base class. This method is called by the `SearchAlgorithm` will call search after initializing parent sets and if the `initAsNaiveBase` flag is set, it will start

with a naive Bayes network structure. After calling `search` in your custom class, it will add arrows if the `markovBlanketClassifier` flag is set to ensure all attributes are in the Markov blanket of the class node.

3. If the structure learner has options that are not default options, you want to implement `public Enumeration listOptions()`, `public void setOptions(String[] options)`, `public String[] getOptions()` and the get and set methods for the properties you want to be able to set.

   NB 1. do not use the -E option since that is reserved for the `BayesNet` class to distinguish the extra options for the `SearchAlgorithm` class and the `Estimator` class. If the -E option is used, it will not be passed to your `SearchAlgorithm` (and probably causes problems in the `BayesNet` class).

   NB 2. make sure to process options of the parent class if any in the get/setOpions methods.

Adding a new estimator

This is the quick guide for adding a new estimator:

1. Create a class that derives from `weka.classifiers.bayes.net.estimate.BayesNetEstimator`. Let's say it is called `weka.classifiers.bayes.net.estimate.MyEstimator`.

2. Implement the methods
   `public void initCPTs(BayesNet bayesNet)`
   `public void estimateCPTs(BayesNet bayesNet)`
   `public void updateClassifier(BayesNet bayesNet, Instance instance)`, and
   `public double[] distributionForInstance(BayesNet bayesNet, Instance instance)`.

3. If the structure learner has options that are not default options, you want to implement `public Enumeration listOptions()`, `public void setOptions(String[] options)`, `public String[] getOptions()` and the get and set methods for the properties you want to be able to set.

   NB do not use the -E option since that is reserved for the BayesNet class to distinguish the extra options for the SearchAlgorithm class and the Estimator class. If the -E option is used and no extra arguments are passed to the SearchAlgorithm, the extra options to your Estimator will be passed to the SearchAlgorithm instead. In short, do not use the -E option.

11 FAQ

How do I use a data set with continuous variables with the BayesNet classes?

Use the class `weka.filters.unsupervised.attribute.Discretize` to discretize them. From the command line, you can use

`java weka.filters.unsupervised.attribute.Discretize -B 3 -i infile.arff -o outfile.arff`
where the -B option determines the cardinality of the discretized variables.

How do I use a data set with missing values with the BayesNet classes?

You would have to delete the entries with missing values or fill in dummy values.

How do I create a random Bayes net structure?

Running from the command line
`java weka.classifiers.bayes.net.BayesNetGenerator -B -N 10 -A 9 -C 2`
will print a Bayes net with 10 nodes, 9 arcs and binary variables in XML BIF format to standard output.

How do I create an artificial data set using a random Bayes nets?

Running
`java weka.classifiers.bayes.net.BayesNetGenerator -N 15 -A 20 -C 3 -M 300`
will generate a data set in arff format with 300 instance from a random network with 15 ternary variables and 20 arrows.

How do I create an artificial data set using a Bayes nets I have on file?

Running
`java weka.classifiers.bayes.net.BayesNetGenerator -F alarm.xml -M 1000`
will generate a data set with 1000 instances from the network stored in the file `alarm.xml`.

How do I save a Bayes net in BIF format?

- **GUI:** In the Explorer
  - learn the network structure,
  - right click the relevant run in the result list,
  - choose "Visualize graph" in the pop up menu,
  - click the floppy button in the Graph Visualizer window.
  - a file "save as" dialog pops up that allows you to select the file name to save to.
- **Java:** Create a `BayesNet` and call `BayesNet.toXMLBIF03()` which returns the Bayes network in BIF format as a String.
- **Command line:** use the `-g` option and redirect the output on stdout into a file.

How do I compare a network I learned with one in BIF format?

Specify the -B <bif-file> option to BayesNet. Calling `toString()` will produce a summary of extra, missing and reversed arrows. Also the divergence between the network learned and the one on file is reported.

How do I use the network I learned for general inference?

There is no general purpose inference in Weka, but you can export the network as XML BIF file (see above) and import it in other packages, for example JavaBayes available under GPL from http://www.cs.cmu.edu/~ javabayes.

12 Future development

If you would like to add to the current Bayes network facilities in Weka, you might consider one of the following possibilities.

- Implement more search algorithms, in particular,
  - general purpose search algorithms (such as an improved implementation of genetic search).
  - structure search based on equivalent model classes.
  - implement those algorithms both for local and global metric based search algorithms.
  - implement more conditional independence based search algorithms.
- Implement score metrics that can handle sparse instances in order to allow for processing large datasets.
- Implement traditional conditional independence tests for conditional independence based structure learning algorithms.
- Currently, all search algorithms assume that all variables are discrete. Search algorithms that can handle continuous variables would be interesting.
- A limitation of the current classes is that they assume that there are no missing values. This limitation can be undone by implementing score metrics that can handle missing values. The classes used for estimating the conditional probabilities need to be updated as well.
- Only leave-one-out, k-fold and cumulative cross-validation are implemented. These implementations can be made more efficient and other cross-validation methods can be implemented, such as Monte Carlo cross-validation and bootstrap cross validation.
- Implement methods that can handle incremental extensions of the data set for updating network structures.

And for the more ambitious people, there are the following challenges.

- A GUI for manipulating Bayesian network to allow user intervention for adding and deleting arcs and updating the probability tables.
- General purpose inference algorithms built into the GUI to allow user defined queries.
- Allow learning of other graphical models, such as chain graphs, undirected graphs and variants of causal graphs.
- Allow learning of networks with latent variables.
- Allow learning of dynamic Bayesian networks so that time series data can be handled.

References

[1] R.R. Bouckaert. Bayesian Belief Networks: from Construction to Inference. Ph.D. thesis, University of Utrecht, 1995.

[2] W.L. Buntine. A guide to the literature on learning probabilistic networks from data. IEEE Transactions on Knowledge and Data Engineering, 8:195–210, 1996.

[3] J. Cheng, R. Greiner. Comparing bayesian network classifiers. Proceedings UAI, 101–107, 1999.

[4] C.K. Chow, C.N.Liu. Approximating discrete probability distributions with dependence trees. IEEE Trans. on Info. Theory, IT-14: 426–467, 1968.

[5] G. Cooper, E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. Machine Learning, 9: 309–347, 1992.

[6] N. Friedman, D. Geiger, M. Goldszmidt. Bayesian Network Classifiers. Machine Learning, 29: 131–163, 1997.

[7] D. Heckerman, D. Geiger, D. M. Chickering. Learning Bayesian networks: the combination of knowledge and statistical data. Machine Learining, 20(3): 197–243, 1995.

[8] Moore, A. and Lee, M.S. Cached Sufficient Statistics for Efficient Machine Learning with Large Datasets, JAIR, Volume 8, pages 67-91, 1998.

[9] Verma, T. and Pearl, J.: An algorithm for deciding if a set of observed independencies has a causal explanation. Proc. of the Eighth Conference on Uncertainty in Artificial Intelligence, 323-330, 1992.

[10] I.H. Witten, E. Frank. Data Mining: Practical machine learning tools and techniques. 2nd Edition, Morgan Kaufmann, San Francisco, 2005.