Evaluating the Explanatory Value of Bayesian Network Structure Learning Algorithms

Patrick Shaughnessy
University of Massachusetts, Lowell
pshaughn@cs.uml.edu

Gary Livingston
University of Massachusetts, Lowell
gary@cs.uml.edu

Abstract

This paper presents a technique for evaluating the degree of correctness of structural models produced by Bayesian network learning algorithms. In this method, (1) Bayesian networks are generated pseudo-randomly using a chosen model distribution; (2) data sets of various sizes are produced using the generated networks; (3) the data sets are passed to learning algorithms; and (4) the network structures output by the learning algorithms are compared to the original networks. In the past, similar methods have used individually hand-selected networks rather than generating large families of networks and have focused on data sets with a large number of cases.

Sample results on several search-and-score algorithms are shown, and we discuss the use of domain-specific simulators for generating data which may be used to better evaluate the causal learning.

Introduction

Machine learning algorithms can be applied for either explanatory or predictive use. A predictive use of machine learning is an attempt to fill in unknown values in data. An explanatory use is an attempt to construct a model which can give a user insight into previously unknown relationships that are manifested in the data. Machine learning algorithms are commonly evaluated on their predictive power, as prediction accuracy can be unambiguously defined and measured. (For instance, (Cheng and Greiner 1999) and (Friedman, Geiger, and Goldszmidt 1997) take this approach.) However, predictive power is not necessarily an indicator of explanatory value.

This paper presents a framework for the evaluation of explanatory value of a particular subset of machine learning algorithms: those that learn Bayesian networks (BNs). In particular, we will look at a framework for evaluating the ability of BN learning systems to learn network structure. Background information regarding Bayesian networks is presented in (Cooper and Herskovits 1992). To briefly summarize, a BN is a directed acyclic graph in which each node corresponds to a variable. The value of each variable in a BN "depends" (in a loose sense) on the values of its parents and is conditionally independent from the rest of its non-descendants given its parents.

A property of Bayesian networks which is desirable for explanatory use, but does not necessarily follow from their definition, is that the value of each attribute is a direct effect of the values of the attribute's parents and a direct cause of the values of the attribute's children. In practice, learning such causality from observational data requires many assumptions to hold (Spirtes, Glymour, and Scheines 2001, chapter 3), and in a typical interesting domain it is normal for many of these assumptions to fail. For example, while it has been shown that correct partial modeling of causality can still occur in the presence of unmeasured confounding variables (Spirtes, Glymour, and Scheines 2001, chapter 6), it must be known in advance that those confounding variables exist and the overall nature of their effects must be known.

Even though a Bayesian network may be unlikely to or even mathematically unable to model the true causality of a domain, a network which is partially incorrect may still provide valuable explanatory information. For example, even if the directionality of edges output by a learning algorithm is essentially random, it may still be the case that "nearby" nodes are more related than "distant" ones, and that an edge or short undirected path between two nodes signifies the existence, though not the true direction, of some close causal relationship.

A BN which provides a meaningful approximation of causality is a powerful and convenient view of interrelationships in the data. The degree to which the graph models the causality of the underlying process which generated the data can be considered the explanatory value of the network. This is distinct from its predictive power: it is possible that two different graphs will produce the same predictions of probability in all cases, yet one may still be a better depiction of the causal processes at work than the other.

To determine the degree to which a learned graph matches the relationships underlying the data, it is necessary that those actual relationships be known. Additionally, to thoroughly test an algorithm's ability to learn graphs, it is necessary that many different known relationships be used. The supply of real-world domains in which all relationships are known is limited and arguably nonexistent, so rather than use real data it is necessary to artificially generate relationships and to produce artificial data conforming to them. The relationships found by a learning algorithm can then be compared to those used in generating the data; this comparison evaluates the learning algorithm's ability to produce correct explanations over domains in which relationships resemble those from which the data were generated.

An approximation of the domain is thus used to measure the power of a method to find approximations, providing measures of the method's correctness which are themselves approximate. This uncertainty is unavoidable; the only true yardstick to evaluate explanatory learning is a correct explanation, and if we have a correct explanation for a domain then we have no need to learn an explanation algorithmically.

In this paper, we present a process for performing this evaluation and some example results from applying the process to a small selection of fast, readily available BN learning algorithms.

Method

Evaluation Method

There are two major steps to the evaluation method. The first step is the generation of artificial data sets with known causal explanations. The second step is the comparison of these known explanations to the models produced by BN learning algorithms from the data sets. These two steps are automated and repeated many times to reach confident conclusions concerning the degree of correctness of the explanations produced by the algorithms, that is, the degree to which the learned model matches the model from which the data were generated.

The model space from which a data set is generated is the same model space which Bayesian network learning algorithms search: directed acyclic graphs (DAGs) embodying dependence relations and conditional probability tables (CPTs) parameterizing the specific relationships.

The results produced from the evaluation are dependent on the distribution from which the DAGs and CPTs used to generate the data are drawn. In order to evaluate learning algorithms for a specific real-world domain, it is logical to use a distribution which is believed similar to the distribution of causal relationships existing in actual data. This calls for using domain-specific simulators which embody domain knowledge to generate data for evaluating causal learning. Fortunately, many domains already have suitable simulators, e.g. the Gene Expression Simulation Project at the University of Pittsburgh for gene expression data (http://bioinformatics.upmc.edu/GE2/index.html) and OPNET for computer networking data (http://www.opnet.com). The flow of causality between variables in data generated by these simulators can be easily determined for an appropriate featurization, and the data can be easily passed to learning algorithms. Thus, the algorithms can be evaluated for the degree to which their output network structure matches the causality believed to be present in the domain, without having to deal with the unidentified confounders present in real experiments or the cost of performing such experiments.

With a data set thus produced, learning algorithms are invoked to infer Bayesian networks from the data set. The DAG of each network (referred to afterward as the "learned DAG") is compared with the DAG used in generating the data (referred to afterward as the "source DAG") to produce numerous measures which can be compared between learning algorithms to evaluate their explanatory power.

For each pair of learned and source DAGs, various counts are computed, such as:

- **Edge true positive count** ($\mathtt{TP}_{\mathtt{edge}}$): The number of edges which exist in both the learned DAG and source DAG.
- **Edge flip count** ($\mathtt{Flip}_{\mathtt{edge}}$): The number of edges which would be true positives if they were pointing the other way. That is, the edge exists in the undirected version of the learned and source DAGs, but the direction of the edge is not consistent between the two.
- **Edge false positive count** ($\mathtt{FP}_{\mathtt{edge}}$): The number of edges which exist in the learned DAG but are not found in the source DAG in either forward or reversed form.
- **Edge true negative count** ($\mathtt{TN}_{\mathtt{edge}}$): The number of unordered pairs of distinct nodes which have no edge between them in either direction in either the learned or source DAG. If there are N nodes and no edges in either DAG, this count has its maximum value of $(N^2 - N)/2$.
- **Edge false negative count** ($\mathtt{FN}_{\mathtt{edge}}$): The number of edges which exist in the source DAG but are not found in the learned DAG in either forward or reversed form.
- **Forward path count** ($\mathtt{F}_{\mathtt{path}}$): The number of pairs (A, B) such that a directed path exists from A to B in the source dag and a directed path from A to B exists in the learned DAG.

After summing these counts over multiple experiments (in our case 100) to reduce random variance and build statistical confidence, metrics are computed, such as the following (all of which range from 0 to 1):

- **Directed edge sensitivity**: [`TP`$_{edge}$ `/ (TP`$_{edge}$ `+ Flip`$_{edge}$ `+ FN`$_{edge}$`)`]. This measures how well the algorithm can detect directed edges which should exist. The ideal value is 1.
- **Undirected edge sensitivity**: [`(TP`$_{edge}$ `+ Flip`$_{edge}$`) / (TP`$_{edge}$ `+ Flip`$_{edge}$ `+ FN`$_{edge}$`)`]. This measures how well the algorithm can detect edges, forgiving errors which are purely of direction. The ideal value is 1.
- **Undirected edge precision**: [`(TP`$_{edge}$ `+ Flip`$_{edge}$`) / (TP`$_{edge}$ `+ Flip`$_{edge}$ `+ FP`$_{edge}$`)`]. This measures how confident one can be that edges are genuine, forgiving errors of direction. The ideal value is 1.
- **Edge false positive rate**: [`FP`$_{edge}$ `/ (TN`$_{edge}$ `+ FP`$_{edge}$`)`]. This measures how likely the algorithm is to detect an edge where none exists. The ideal value is 0.
- **Edge direction ratio**: [`TP`$_{edge}$ `/ (TP`$_{edge}$ `+ Flip`$_{edge}$`)`]. This measures how well the algorithm determines the direction of an edge, given that it has correctly found the existence of one. The ideal value is 1.

Much of this method is similar to the approach used in (Spirtes, Glymour, and Scheines 2001, chapter 5) to argue the relative merits of the PC, IG and SGS algorithms; we introduce some metrics not present in that analysis. The evaluation method used in (Acid and de Campos 2003) is also similar to this method, but they generated data only from a few hand-picked models. In contrast, our method involves a larger number of models, generated to a given model distribution.

Selected Learning Algorithms

This method is applicable to any learning algorithm which produces a BN structure from a data set. Here we demonstrate it on a selection of algorithms of the *search-and-score* variety. These algorithms perform an incremental *search* over the space of possible DAGs to maximize a *scoring function* which approximates the fitness of the DAG with respect to the data. We examined three search methods and three scoring functions, yielding nine search-and-score combinations. We used the implementations of these algorithms that are found in the WEKA software package (Witten and Frank 1999).

The search methods we examined are basic hill-climbing search, repeated hill-climbing search, and the classic greedy search method named K2 (Cooper and Herskovits 1992).

Hill-climbing search begins at some start state (in our case, a "naïve Bayes" graph in which one node is arbitrarily a parent of all others). It adds and removes edges one at a time such that each action taken maintains acyclicity and increases the score of the graph as much as possible, stopping when a graph is reached for which no single add or remove operation will further increase the score.

Repeated hill-climbing search executes multiple independent hill-climbing searches, each using a different random graph as a starting point, and returns the highest-scoring graph of those found.

K2 takes as an extra input an ordering of attributes. It iterates over the attributes in that order, considering only sets of preceding attributes as possible parents and adding the edges for the best-scoring set of parents. Thus, it only considers a subset of DAGs for which the given ordering is a valid topological sort. This is a substantially smaller space than the set of all possible DAGs, giving K2 a shorter computation time to find a good local maximum within that space. In our case, the ordering passed to K2 was random and independent of the ordering used in generating the causal DAG.

The three score functions we examined are BDeu, the "Bayesian metric" found in WEKA, and MDL.

BDeu (an abbreviation for "Bayesian Direchlet equivalent with uniform priors") is the scoring function originally proposed in conjunction with K2 (Cooper and Herskovits 1992). The theoretical basis is this: starting from the assumption that each possible structure has an equal prior probability of being correct regardless of how simple or complex it is, assign a score to a structure based on its probability of being correct given the data. There are more complex structures than simple ones, so a search using BDeu as its scoring function will tend toward relative complexity, adding more edges than the same search method using one of the other fitness functions considered.

WEKA's Bayesian scoring function is similar to BDeu, but where BDeu assumes equal prior structure probabilities, WEKA BAYES also makes a substantial assumption concerning the distribution of probabilities in the CPTs. In particular, it acts as though in addition to the data presented it, there are extra cases containing equal proportions of every possible combination of values. The attributes thus appear more independent. The extra cases are given less weight than the real ones, but as our results show the difference they make in the learned graph is substantial. WEKA's Bayesian scoring function adds fewer edges than BDeu but more edges than MDL.

MDL (an abbreviation for "Minimum Description Length") is a scoring principle used in many contexts which rewards model simplicity (Rissanen 1978). The theoretical basis is minimization of the information-theoretic bits needed to encode both the model and the deviation of the data from the model. Thus, simpler models and models that better fit the data are both rewarded, with a tradeoff between the two conditions. As a search algorithm for BNs, the restrictive tendency of MDL results in graphs with relatively few edges.

Results on Selected Algorithms

In order to explore and demonstrate this method, we tested each of the nine search/score combinations with 60 groups of 100 small data sets each. Each group of 100 was generated from a different combination of number-of-cases, number-of-attributes, and DAG distribution. An arbitrary distribution of models is explored here rather than a domain-specific one, as our intent here is to demonstrate the method itself and means of drawing results from it more so than any actual result.

The number of cases in each data set was varied among 5, 10, 25, 50, and 100. The number of attributes was varied among 5, 10, 25, and 50, with all attributes discrete and having three values. The distribution of BNs was varied among:

- DAGs containing no edges (and thus all attributes independent of each other). Most of the metrics are undefined or constant in this case, but false positive rate is meaningful.
- Sparse arbitrary DAGs limited to two parents per node.
- Denser arbitrary DAGs limited to three parents per node.

The same arbitrary CPT construction algorithm was used for each distribution. All data generated were passed to the learning algorithms; no variables were kept hidden for this experiment.

For each of the 60 groups, we invoked 9 search-and-score combinations (3 search methods × 3 score functions). We invoked each search method with the constraint that it assign no greater than 3 parents to any node, as the search time and memory usage are potentially exponential in the number of parents. The restrictions in the DAG distribution parameters ensured that this restriction would not place the correct model outside the constrained search space. The ordering passed to K2 was generated randomly, independently from the ordering used in the DAG generation. Repeated hill-climbing search was invoked to perform 10 iterations. The computation time for the experiment was dominated by the many runs of the learning algorithms; all other computation was negligible.

As the distribution of models was arbitrary, no significant conclusions regarding any specific domain can be drawn from this experiment. However, some statements about the relative character of the algorithms can be made, with the caveat that these observations may, in a given domain, vary.

The most essential observation to be drawn from the results is that the choice of scoring function is much more important to the results than the search method used to maximize that score. MDL achieved

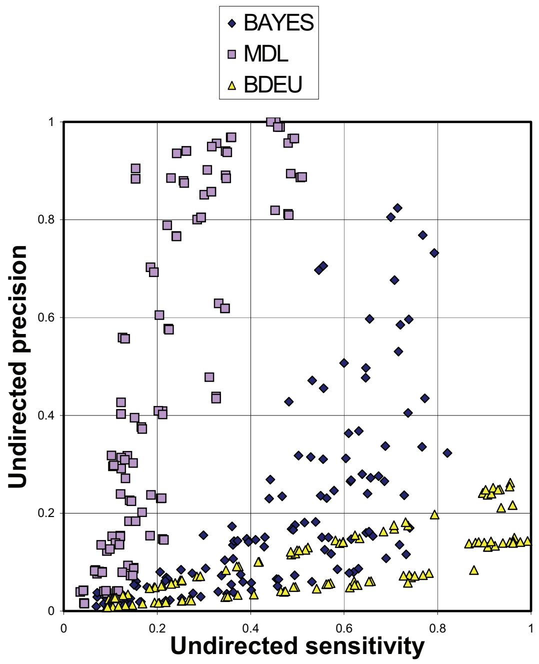

Figure 1. Undirected edge precision and sensitivity, with trials marked by scoring heuristic. Note that each scoring heuristic has a distinct sensitivity/precision relationship.

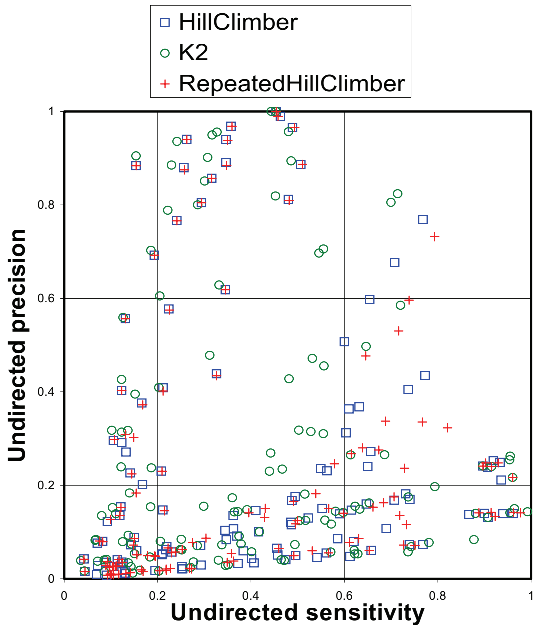

Figure 2. Undirected edge precision and sensitivity, with trials marked by search method. Note the generally similar performance of the search methods.

high precision at the expense of sensitivity, while BDeu made a tradeoff in the opposite direction. WEKA's Bayesian scoring function provides both sensitivity and precision which lie between the other score algorithms. These characteristics are visible in Figure 1. As there are many reasons to create a BN, the scoring algorithm should be selected with an eye to one's motivations in doing so. Depending on the objectives of the process, one's preferences regarding the relative importance of sensitivity and precision will vary.

The choice of search method plays a smaller role. Counter to what might be expected, K2 performed relatively well despite the order of its search having been randomly constrained. We believe that this is due to the relatively thorough manner in which it examines its search space. Despite the extra work performed by WEKA's repeated hill-climbing search, it did no better than single hill-climbing and frequently did worse. We believe that this is due to the randomness of the start states, as a random graph generated with no regard for prior DAG distribution is likely to be far from the correct (fairly sparse) model and thus likely to give the search many chances to hit a poor local maximum. The magnitude of all of these differences is, however, quite small compared to the differences introduced by the choice of scoring function. Figure 2 plots sensitivity and precision by search.

As one would expect, the false positive rate and sensitivity of edge learning are related; if more edges are learned in total, more true edges will be learned and more false edges will be learned. Figure 3 shows the relationship between false positive rate and sensitivity by scoring heuristic. From this figure, it can be seen that undirected edge sensitivity [(true + flipped edges) / total edges in source DAG] is approximately bounded below by edge false positive rate [false positive edges / total non-edges in source DAG] but often exceeds the false positive rate.

Over the arbitrary distribution we chose and the search-and-score algorithms examined, it can be noted that the learning of direction appears consistently poor. Our edge direction ratio [true edges / (true + flipped edges)] never exceeded 0.77, and that was an outlier with the next-highest result being below 0.69. As the causality of our original model space was wholly arbitrary, these results are themselves arbitrary and serve solely to demonstrate the use of the metrics. If the model space were generated from a domain simulator, then this metric would indicate the degree to which edge directions output by the learning algorithm could be expected to model actual causality in that domain.

Discussion

A Bayesian network is not solely a classification tool. A good Bayesian network can elicit relationships within the data that indicate correlation or suggest causality. Measuring the value of a Bayesian network for this purpose is not an easy task, and using classification accuracy of a target as the sole metric for the quality of a learning algorithm overlooks much of the actual worth of Bayesian network methodology.

We have outlined a method for evaluating Bayesian networks for their explanatory usefulness. The utility of an explanation is expressed in terms of the degree to which the learning algorithm's output matches the causal processes underlying the data source. In order to use this method, an appropriate data source must be used; generally this data source would be derived using a simulator rather than real experiments, as only in a simulated environment can the underlying causality be fully known. Simulators which embody a great deal of domain knowledge already exist for many domains and can be exploited for evaluating learning over those domains.

Depending on the purpose of one's experimentation and the type of explanatory information one seeks, it may be more important that the network contain many edges of potential relevance or that those edges within the network are very likely to be relevant. This tradeoff between sensitivity and precision in edge

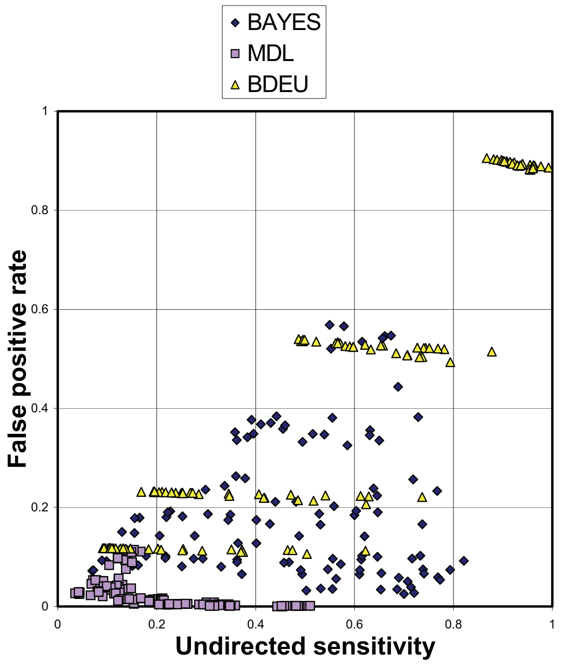

Figure 3. Undirected edge sensitivity and false positive rate, with trials marked by scoring heuristic. Each scoring heuristic has distinct behavior. The four bands of BDeu correspond to the number of attributes in the network, with the highest band corresponding to the smallest number of attributes (5).

learning is an essential characteristic to be considered in choosing algorithms, and the method here described can aid in that selection.

Another dimension of explanatory utility exists in the process used to explore the network after it is constructed. A graph which provides a sufficiently accurate model may still fail as an explanation if the tools used to visualize that graph fail to express it clearly; we hold such considerations outside the scope of this paper, except to briefly note that graphs with fewer edges are often easier to visualize well.

In the presented experiment we used data sets with a small number of cases, in contrast to much research which considers asymptotically large data [for instance, (Cheng, Bell, and Liu 1997) and most of the theorems of (Spirtes, Glymour, and Scheines 2001)]. Our method can explore the behavior of learning algorithms over a wide continuum of data set sizes. Similarly, while we revealed all variables of all data points to the learning algorithms here, variables could be just as easily hidden, allowing systematic exploration of the behavior of learning algorithms in the presence of unmeasured confounders.

This experiment only considered algorithms which use a search and score approach to output a single fully directed graph. These algorithms are easy to apply and computationally fast, but they are not a complete picture of Bayesian network structure learning. Other approaches can also be explored using the presented methodology by varying a different selection of parameters to the algorithms and slightly modifying some metrics. Such approaches include algorithms which search for families of graphs (Pearl and Verma 1991) and algorithms which search explicitly over conditional independence statements (Cheng, Bell, and Liu 1997). It may be that in some important domains search-and-score algorithms are particularly strong or weak compared to others, and it is almost certainly the case that learning algorithms incorporating domain-specific knowledge have the potential to outperform general-purpose ones within their domain. By harnessing domain-specific simulators, this too can be tested.

References

Acid, S., and L. M. de Campos. 2003. Searching for Bayesian Network Structures in the Space of Restricted Acyclic Partially Directed Graphs. *Journal of Artificial Intelligence Research* 18:445–490.

Cheng, J.; Bell, D.; and Liu, W. 1997. An Algorithm for Bayesian Belief Network Construction from Data. In Preliminary Papers of the International Workshop on Artificial Intelligence and Statistics, pp. 83–90. Fort Lauderdale, Fla.: AT&T Labs.

Cheng, J., and Greiner, R. 1999. Comparing Bayesian Network Classifiers. In Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence, pp. 101–108. Stockholm, Sweden: Morgan Kaufmann.

Cooper, G., and Herskovits, E. 1992. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning* 9:309–347.

Friedman, N.; Geiger, D.; and Goldszmidt, M. 1997. Bayesian Network Classifiers. *Machine Learning* 29:131–163.

Pearl, J., and Verma, T. 1991. A Theory of Inferred Causation. In Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning, pp. 441–452. Cambridge, MA: Morgan Kaufmann.

Rissanen, J. 1978. Modeling by Shortest Data Description, *Automatica* 14:465–471.

Spirtes, P.; Glymore, C.; and Scheines, R. 2001. *Causation, Prediction, and Search.* New York, NY: MIT Press.

Witten, I., and Frank, E. 2005. *Data Mining: Practical machine learning tools and techniques.* San Francisco, Cal.: Morgan Kaufmann.