One-Pass Learning Algorithm for Fast Recovery of Bayesian Network

Shunkai Fu [1], **Michel C. Desmarais** [1] **and Fan Li** [2]

Ecole Polytechnique de Montreal, Canada [1], SPSS Inc. [2]
{shukai.fu, michel.desmarais@polymtl.ca}, fli@spss.com

Abstract

An efficient framework is proposed for the fast recovery of Bayesian network classifier. A novel algorithm, called Iterative Parent-Child learning-Bayesian Network Classifier (IPC-BNC), is proposed to learn a BNC without having to learn the complete Bayesian network first. IPC-BNC was proved correct and more efficient compared with a traditional global learning algorithm, called PC, by requiring much fewer conditional independence (CI) tests. Besides, we recognize and introduce AD-tree into the implementation so that computational efficiency is further increased through collecting full statistics within a single data pass. The IPC-BNC and AD-tree combination is demonstrated very efficient in time by our empirical study, making itself an attractive solution in very large applications.

Keywords: Bayesian Network classifier, IPC-BNC, AD-tree

Introduction

Classification is a fundamental task in data mining that requires learning a classifier through the observation of data. Basically, a classifier is a function that maps instances described by a set of attributes to a class label. Naïve BAyes neworks have been widely used for the task of classification (Duda & Hart 1973, Langley 1995) (**Fig 1** upper-left). They represent a special case of the more general Bayesian networks (BN) formalism and are characterized by their strong assumption about the independence of attributes given the target node. Although they generally perform fairly well in spite of this assumption (Domingos & Pazzani, 1997), they lack the power to represent more complex dependencies among attributes and the target node that can affect performance. Tree Augmented Naïve Bayes (Friedman et al. 1997) (**Fig 1** upper-right) is an extension of Naïve Bayes that weakens its assumption, allowing additional dependence relations among attributes. It is empirically shown to yield better performance (Friedman et al. 1997).

Compared with Naïve Bayes and TAN, a BN((**Fig 1** bottom) doesn't distinguish between the target and attributes. The target either can be a parent or child of attributes, and general dependencies are found among attributes. Although such general BNs are expected to yield better performance than Naïve Bayes and TAN, the NP-completeness complexity to learn a BN inhibits its widespread application.

However, we note that not all attributes are effective in predicting the target in applying BN as a classifier. With the BN example in **Fig 1** (bottom), we have a decision rule like $P(T \mid X_1,\dots,X_7) \propto P(T, X_1,\dots,X_7) = P(T \mid X_3, X_5 \mid T)$ $P(X_2 \mid X_1)P(X_3 \mid X_1)P(X_1)P(X_4 \mid X_2,T)P(X_5 \mid X_6)P(X_6)$ $P(X_7 \mid T)$, of which some terms, namely $P(X_1)$, $P(X_2 \mid X_1)$, $P(X_3 \mid X_1)$, $P(X_5 \mid X_6)$, and $P(X_6)$, are not involve the target node, which means that their values have no influence on the classification decision of T. By removing them, we obtain a simpler decision rule with no sacrifice with regards to classification performance:

$$\underset{t}{\boldsymbol{argmax}}\ P(T = t \mid X_3, X_5)P(X_4 \mid X_2, T)P(X_7 \mid T)$$

The attributes involved in this new version of the decision rule form a set of nodes called the Markov blanket of T. This concept can be traced back to (Pearl 1988). A formal definition of a Markov blanket is given below.

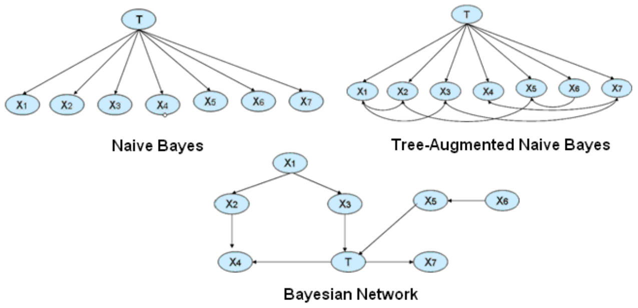


Fig 1. Examples of Bayesian classifiers, including Naïve Bayes (upper-left), Tree-Augmented Naïve Bayes (upper-right) and Bayesian Network (bottom)

Definition 1 (Markov blanket). The Markov blanket of a node X in a BN is the set of nodes composed of X's parents, children and children's parents (spouses of X, actually). It is denoted as $MB(X)$ in the remaining text.

In our example (**Fig 1** bottom), $\{X_3, X_5\}$ are T's parents; $\{X_4, X_7\}$ are its children, and $\{X_2\}$ is T's spouse sharing with T a common child, X_4.

Definition 2 (Conditional independence). Variables X and Y are conditionally independent given the set of variables $\boldsymbol{Z}$, iff. $P(X \mid Y, \boldsymbol{Z}) = P(X \mid \boldsymbol{Z})$, denoted as $X \perp Y \mid \boldsymbol{Z}$.

Given a domain of $\boldsymbol{U}$, and the full knowledge of $MB(X)$, then X is independent of any Y falling outside $MB(X)$, that is $X \perp Y \mid MB(X), \forall Y \in \boldsymbol{U} \setminus MB(X) \setminus \{X\}$ This important characteristic of a Markov blanket tells us that given the whole BN over $\boldsymbol{U}$, only the sub-network over T and $MB(T)$ is effective in the prediction of T, and we refer it as a Bayesian network classifier.

Definition 3 (Bayesian Network Classifier, or BNC). In our proposal, we refer to a Bayesian network classifier as the directed acyclic graph (DAG) over T and $MB(T)$. A Bayesian network is reserved for the complete DAG over all attributes plus T. When all attributes belong to $MB(T)$, the BN actually equals the target BNC.

Given a learned BN, we can easily derive a BNC for any target node T, but until now, none of existing known BN learning algorithms claims to scale well to more than a few hundred variables. The primary contribution of this paper is to propose a learning algorithm, called Iterative Parent-Child learning of BNC(IPC-BNC), that forgo structure learning of the whole BN. We report an empirical study showing that much fewer CI tests are needed compared with the usual learning a whole BN using the PC algorithm (Spirtes et al. 2000). Besides, by introducing AD-tree (Moore & Lee 1998), an enhanced version, IPC-BNC++, is derived, where a single data pass is enough to go through the data file and collect necessary statistics. Meanwhile, this data structure provides a convenient mechanism for quick query of specific statistics. Because IPC-BNC requires intensive CI testing, the use of an AD-tree greatly reduces the computational cost. Therefore, IPC-BNC++ is expected to solve larger applications once non-tractable.

Local Structure Learning by IPC-BNC

Overview

There are two major different approaches for the structure learning of BN: scoring-based or CI-based (conditional independence based). The scoring- based learning algorithm looks for the BN that best fits the data based on some scoring function. This approach corresponds to an optimization problem. The CI-based method attempts to derive the BN structure by identifying a series of conditional independence relations among the nodes, according to the concept of d-separation (Pearl 1988). Its search procedure consists in identifying those conditional independence relations. This is also often referred to as a constraint-based algorithm (Cheng & Greiner 1999, Cheng et al 1997, Spirtes et al 2000).

We choose the second approach to design this algorithm because it enables us to determine specific connections among variables, which acts as the basis of our algorithm. We inherit two fundamental assumptions of this kind of learning algorithm: faithfulness and correct CI test.

Definition 4 (Faithfulness). A Bayesian network G and a joint distribution P are faithful to one another, iff. every conditional independence relationship encoded by the graph of G is also presented in P, i.e., $T \perp_G X \mid \boldsymbol{Z} \Leftrightarrow T \perp_P X \mid \boldsymbol{Z}$ (Spirtes et al. 2000).

In practice, we view a CI test as reliable if the number of instances in dataset is at least five times the number of degrees of freedom in the test.

With the two assumptions and the known underlying topology knowledge of a typical BNC (see **Definition 1** and **3**), we outline here how the proposed algorithm learn the BNC. Firstly, we recognize those nodes directly connected to the target T, which are known as T's parents and children, denoted as $PC(T)$. Secondly, we scan each $X \in PC(T)$ to find $PC(X)$, of which those satisfying the condition of being T's spouse will be recognized and denoted as $SP(T)$. Those arcs discovered during this iterative procedure construct the skeleton of final BNC. Finally, a series of orientation rules can be applied to determine the direction of arcs.

Theoretical Basis

In this section, we provide a theoretical background for the correctness of our algorithm.

Theorem 1. If a Bayesian network G is faithful to a probability distribution P, then for each pair of nodes X and Y in G, X and Y are adjacent in G iff. $X \not\perp Y \mid \boldsymbol{Z}$ for all $\boldsymbol{Z}$ such that X and $Y \notin \boldsymbol{Z}$. (Spirtes et al. 2000)

Lemma 1. If a Bayesian network G is faithful to a probability distribution P, then for each pair of nodes X and Y in G, if there exists a $\boldsymbol{Z}$ such that X and $Y \notin \boldsymbol{Z}$, $X \perp Y \mid \boldsymbol{Z}$, then X and Y are NOT adjacent in G.

We get Lemma 1 from Theorem 1, and its proof is trivial. In IPC-BNC, *RecognizePC* (**Table 3**), is designed on this discovering. Actually, the classical structure learning algorithm PC (Spirtes et al. 2000) is the first one working from this basis.

Theorem 2. If a Bayesian network G is faithful to a probability distribution P, then for each triplet of nodes X, Y and V in G such that X and Y are adjacent to V, but X and Y are not adjacent, $X \rightarrow V \leftarrow Y$ is a sub-graph of G iff $X \not\perp Y \mid \boldsymbol{Z}$ for all $\boldsymbol{Z}$ such that X, Y and $V \notin \boldsymbol{Z}$. (Pearl 1988, Spirtes et al. 2000

Theorem 2 plus **Theorem 1** are necessary for IPC-BNC to discover T's spouses.

Algorithm Specification and Proof of Correctness

Overall Framework. The proposed algorithm learns the nodes and edges of the target BNC by repeating the search for parents and children of related nodes, as shown in IPC-BNC (Table 3). The whole procedure can be divided into six sequential steps as described below:

1. *IPC-BNC* begins by searching for the direct neighbors of a given target T, and these variables are known as the

parents or children of T, denoted by $PC(T)$ (line 1-5);

2. Then, false positives are removed from $PC(T)$ through a series of conditional independence tests. For $\forall X \in PC(T)$, $PC(X)$ is learned, and it is known as false positives if $T \notin PC(X)$ (line 6-9);
3. Thirdly, false spouses are filtered out, akin to step 2. For each spouse candidate $X \in SP(T)$, X is removed if $PC(X)$ doesn't contain any parent/child of T (line 10-12);
4. Next, true positives are identified among spouse candidates by identifying the underlying $v-$ structures. For any $X \in SP(T)$, if it constructs such a $v-$ structure with T and someone $Z \in PC(T)$, like $X \rightarrow Z \leftarrow T$, it is known as the true spouse of T based on **Theorem 2**(line 13-19);
5. By removing all nodes that are not connected to T or any $X \in PC(T)$, we get a network over T and $MB(T)$, possibly with some arcs oriented as we identify $v-$ structures during step 4 (line 20);
6. Finally, apply a series of orientation rules over the outcome of step 5 to determine the directions of the remaining edges and output the complete BNC.

These six steps summarize the overall design of IPC-BNC, from which one can see that we repeatedly depend on the recognition of parents and children to determine a connection between any pair of nodes (Step 1, 2 and 3). Meanwhile, we limit the search to the neighborhood of the target as much as possible, filtering out any false positives of $MB(T)$ at as early as possible to restrain the search to a local space. In the following three subsections, we will discuss IPC-BNC in more detail, including its correctness.

Recognize Parents and Children. As the name of this algorithm indicates, the discovery of parents and children plays as a core role in constraining the search to a local space.

RecognizePC (**Table 3**) is responsible for the search of parent/child candidates of given variable. It starts by connecting the current active target T (1st input parameter) to all other nodes not visited by with *RecognizePC* before, with non-oriented edges. Then, it deletes the edge (T, X_i) if there is any subset of $\boldsymbol{ADJ}_T \setminus \{X_i\}$ conditioning on which T and X_i is independent based on the significance of a conditional independence test, I_D. The set obtained, S_{T,X_i}, is kept for later use (line 16 of *IPC-BNC*).

In *IPC-BNC* (discussed in the next section), *RecognizePC* appears at three different locations, line 3, 8 and 12. This is designed to ensure that, for each pair (X,Y), both *RecognizePC(X)* and *RecognizePC(Y)* will be called, and $X-Y$ is true only when $Y \in PC(X)$ and $X \in PC(Y)$, avoiding that any false nodes and links enter into $MB(T)$ and BNC respectively. Overall, this is similar to the conventional PC structure learning algorithm, but it limits the search to the neighbors of the target node, which is why local, instead of global, learning is required and considerable time can be saved especially in applications with a large number of variables.

The correctness of our approach to find the parents and children of a specific node T is the basis for the whole algorithm, so the following theorem is provided.

Theorem 3. Parents and children of the node T of interest can be correctly discovered given the faithfulness assumption.

*Proof. (i) A potential link between (T, X), where X is a candidate of $PC(T)$, is kept only when there is no set $\boldsymbol{S}$ such that T and $X \notin \boldsymbol{S}$, and $I_D(X,T \mid \boldsymbol{S}) \le \varepsilon$, i.e. T and X is conditional independent given $\boldsymbol{S}$. This is the direct application of **Theorem 1**, and this result guarantees that no false parent/child will be added into $PC(T)$. (ii) Our algorithm ensures this point by conditioning on all possible sets $\boldsymbol{S}$. (iii) Since we always start by connecting T with all non-scanned nodes, we won't miss any true positives that should be included. Therefore, all parents and children of T can be identified if we call RecognizePC(T) and RecognizePC(X) for each $X \in PC(T)$ as done in IPC-BNC.*

Identification of the Skeleton of BNC.

Definition 5(Skeleton). Let G be an DAG, and the undirected version of G is called the skeleton of G (Flesch & Lucas 1997, Verma & Pearl 1990).

In the main algorithm body, *IPC-BNC*, *RecognizePC* is iteratively called on demand. We describe this procedure step by step with reference to the pseudo code of *IPC-BNC* in **Table 3**:

1. Firstly (line 2-5), the target T is connected to all other nodes in G. Its parent/child candidates are recognized via calling *RecognizePC*, the result of which may contain false positives that need to be filtered out. T is connected to each $X \in PC(T)$ in G;
2. Next (line 6-9), for each $X_i \in PC(T)$, we call *RecognizePC*(X_i), and determine X_i is a false positive if $T \notin PC(X_i)$. It allows us to filter out those false parents/children of T, and they are disconnected from T in G. For any pair of X_i and X_j belonging to $PC(T)$ by the end of this step, they are connected if $X_i \in PC(X_j)$ and $X_j \in PC(X_i)$, which is guaranteed via the calls of *RecognizePC*(X_i) and *RecognizePC*(X_j). Therefore, by now, $PC(T)$ and the connections over T and $PC(T)$ are determined;
3. Thirdly (line 10-12), spouse candidates are initialized as the union of $PC(X), \forall X \in PC(T)$, except for $PC(T)$ and T itself. This set is denoted as $SP(T)$, and it is based on the known topology knowledge that the spouse of T will be connected to some $Y \in PC(T)$ in G. Given each $X \in SP(T)$, we assume it is connected to some $Y \in PC(T)$. We call *RecognizePC*(X), and it is known as false positive if $Y \notin PC(X)$ (line 12). So, by the end of this step, we get a cleaner spouse set;
4. With the outcome of the third step, we begin to recognize the true spouses in $SP(T)$ (line 13-19). For $\forall X \in SP(T)$, and assume it belongs to

$PC(Y), \exists Y \in PC(T)$, it is true spouse only when there exists such a $\nu -$structure: $X \to Y \leftarrow T$, given **Theorem 3.**

5. Then (line 20), among G, we remove the nodes not belonging to $\{T\} \cup PC(T) \cup SP(T)$, and those arcs connected to these removed nodes. What remains is the skeleton of BNC, with some oriented arcs (during the determination of $\nu -$ structure in step 4);

6. Finally, some orientation rules are applied to determine the remaining non-oriented links, and this will be discussed in the next section.

Theorem 4. Line 1-20 of *IPC-BNC* lets us to learn the complete skeleton of BNC, possibly with some arcs oriented.

Proof. Our step-by-step explanation of this algorithm above actually is the basis for the proof of correctness. (i) For each node pair (X , Y), their connection is determined by RecognizePC(X) and RecognizePC(Y). This is a procedure similar to what the traditional PC algorithm does, which is based on Theorem 1. (ii)Line 1-9 helps us to find $PC(T)$, and all arcs over $\{T\} \cup PC(T)$, which is trivial and can be inferred through our discussion above. (iii) True spouses of T are recognized based on Theorem 2, and this is a sound procedure. (iv)Those additional arcs among true spouses, and between spouses and parents/children of T, are added by line 11-12, and are kept in BNC since we only remove false spouses and the related arcs. Therefore, line 1-20 of IPC-BNC enables to find the complete skeleton of BNC, and some arcs' direction are determined given Theorem 2.

Orientation Rules and More. The orientation step will look for all triples $\{X, Y, Z\}$ such that edges $X - Z$ and $Y - Z$ are in the graph but not the edge $X - Y$. Then, if $Z \notin S_{XY}$, we have oriented edges as $X \to Z$ and $Y \to Z$, which creates a new $\nu -$structure: $X \to Z \leftarrow Y$. After all $\nu -$structures are recognized by repeating this rule, the rest edges are oriented following two basic principles: not to create cycles and not to create new $\nu -$ structure. In our implementation, we refer rules applied in (Kalisch & Buhlmann 2007) and (Meek 1996). However, except for the skeleton and $\nu -$ structure of BNC, those remaining arcs' orientation are trivial to find given the study by (Flesch & Lucas 2007):

Theorem 5. Two DAGs are Markov equivalent with each other if and only if they have the same skeleton and they consist of the same of $\nu -$structure (or immoralities in the original text since they are equivalent concepts).

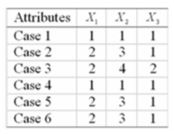
Definition 6(Markov equivalence): Two DAGs are Markov equivalent if they encode the same set of independence relations.

In the section of empirical study, we will only check the skeleton and $\nu -$ structures learned when we compare them to that of the underlying true models. This simplifies the comparison work but without sacrificing the desired effect.

AD-tree and IPC-BNC++

Being a CI test-based algorithm, IPC-BNC depends intensively on computing joint frequency distributions, e.g. $C(X_1 = x_1 \wedge X_3 = x_3)$, intensively. On an extreme, we can look through the dataset to collect a specific joint frequencies on demand, and another data pass for another a new query. This can be terribly time consuming considering thousands of CI tests are required (see our experiment example in next section), and it becomes worst when the number of attributes increases, or the dataset becomes larger. In the implementation of IPC-BNC, we try to cache as much statistics that can be expected given different conditioning set (so called cut-set) size (see **Table** 3) as possible, aiming at reducing the data passes. That approach does work but dozens of data passes still are necessary in our testing, which prohibits IPC-BNC from being an efficient candidate in large applications. An ideal solution we are looking for should be efficient not only in time, allowing all sufficient statistics to be collected in single data pass, but in memory, at least scaling much more slowly relative to complexity of problems (i.e. attribute number).

All Dimensions tree (AD-tree), proposed by Moore and Lee (Moore & Lee 1998, Komarek & Moore 2000), is such a solution. It is introduced for representing the cached frequency statistics for a categorical data set, from which we can query any frequencies we need without having to go through the data repeatedly. **Fig 2** is an example from (Komarek & Moore 2000), where attributes X_1, X_2 and X_3 have 2, 4 and 2 categories respectively. Each rectangular node in the tree stores the value of one conjunctive counting query, and they are called AD-nodes. The children of AD-node are called Vary nodes, displayed as ovals. Each corresponds to an attribute with index greater than that of its parent node.



Attributes	X_1	X_2	X_3
Case 1	1	1	1
Case 2	2	3	1
Case 3	2	4	2
Case 4	1	1	1
Case 5	2	3	1
Case 6	2	3	1

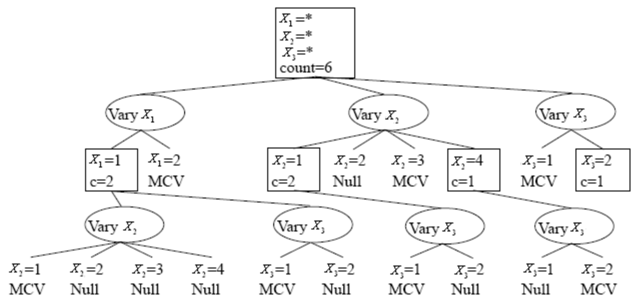


Fig 2. Sample data with tree attributes and six records (upper), and its corresponding AD-tree (bottom).

A tree built in this way would be enormous for non-trivially sized problems, and its complexity increases quickly as the number of attributes and number of categories per attribute increase. However, considering that normally only a small percent of the all possible instances happens given attributes $\{X_i\}$, the actual tree will very often be sparse, with many zero counts (Moore & Lee 1998, Komarek & Moore 2000). This finding is used by the authors to decrease the memory cost greatly, and it is implemented in ours as well. Other techniques mentioned by the authors of AD-tree to make the data structure more efficient are ignored in our current version, and interested readers can refer them in the original documents.

In this project, we refer IPC-BNC with AD-tree as IPC-BNC++, indicating that it is an enhanced version. Its algorithm specification is just same as IPC-BNC (see **Table 3**) since we hide the details of tree construction and query, allowing readers focusing on the primary architecture.

Empirical Study

Experiment Design

We illustrate our method using a well knwon BN called ALARM network (Beinlich et al 1989). ALARM network was developed for on-line monitoring of patients in intensive care units, and is widely used as a benchmark model to evaluate BN's learning algorithms. ALARM network has 37 nodes and 46 edges. We use synthetic data sampled from this network, and we run IPC-BNC with each node in the BN as the target random variable T, trying to recover the corresponding BNC.

The different sample size we have used are 5000, 10000, and 20000. With each sample size, 10 replicates are prepared for study. Due to space constraint, we are unable to present results on other common BNs.

Evaluation and Analysis

To measure time efficiency, we use the number of data passes and CI tests to represent the average cost of the algorithms.

In **Table 1**, the "# CI tests" of IPC-BNC refers to the average number of CI test we need to measure to induce the corresponding BNC given X_i ($i = 1..37$, ranging over each node in Alarm network) as the target. The amount for PC is the total number of CI tests required to learn the whole Alarm BN as by traditional approach. We see that about 70% fewer CI tests are necessary for local search, compared with the global search method. Therefore, IPC-BNC is more economical in time cost by learn only what we need, which enables it to solve larger scale of problem once non-tractable for PC.

The number of data passes is also listed in **Table 1** as "#Data passes". For example, given 20000 instances, IPC-BNC needs 24 data passes by average, whereas IPC-BNC++ needs only 1 pass. The time saved become more obvious when the observation is large. Therefore, IPC-BNC++ gives us an attractive solution which not only requires much fewer CI tests than traditional PC, but also needs a single data pass to learn a BNC

Table 1: Time efficiency comparison, in term of number of CI test and data passes required by PC to learn the global network and the average amount of CI tests cost by IPC-BNC and IPC-BNC++ over all the 37 nodes.

Instances	Algorithm	# CI tests	#Data passes
5k	PC	5458±35	N/A
5k	IPC-BNC	1522±12	23±0
5k	IPC-BNC++	1522±12	1±0
10k	PC	5703±24	N/A
10k	IPC-BNC	1728±14	24±0
10k	IPC-BNC++	1728±14	1±0
20k	PC	5853±32	N/A
20k	IPC-BNC	1833±24	24±0
20k	IPC-BNC++	1833±24	1±0

Table 2: Effectiveness of IPC-BNC and IPC-BNC++

Instances	Distance of Edges	Distance of v-structure
5k	0.09±.00	0.10±.00
10k	0.06±.00	0.07±.00
20k	0.06±.00	0.08±.00

Table 2 is about the effectiveness of the proposed learning algorithm. We measure the distance of edges and $v-$ structure learned fr~~om the true model. Here, the~~ distance is defined as $\sqrt{(1-\Pr ecision)^2 + (1-\operatorname{Re} call)^2}$. Precision is the number of true positives found divided by the number of edges/v-structures of the true model; recall is the value of true positives found divided by the number of edges/ $v-$ structures found.

Conclusion

In using a BN for classification, we make use of what is known as the Markov blanket of the target node, i.e. the target's parents, children and spouses, thereby ensuring that only the relevant features are selected and that all other nodes can be ignored. The DAG structure over $MB(T)$ and T is called Bayesian network classifier. An algorithm called IPC-BNC is proposed to learn BNC with a local search, iteratively looking for those parents and children of any node of interest. It is proved correct and shown to outperform other algorithms for deriving the Markov blanket in an experiment with a single data set. Although further experiments are required to validate the generality of these results, the IPC-BNC is expected to be much more time efficient than a conventional approach where a complete BN over all attributes and the target variable has to be learned first before BNC can be derived. Furthermore, by implementing AD-tree to collect all statistics in one data pass, computational efficiency is greatly improved. We claim that IPC-BNC plus AD-tree is a better approach to solve larger scale of problem, and owns more practical value in real applications.

References

Beinlich, I., Suermondt,H., Chavez,R., and Cooper,G. 1989. The Alarm Monitoring System: A Case Study with two Probabilistic Inference Techniques for Belief Networks. *In Proceedings of the Second European Conference on Artificial Intelligence in Medical Care*, pp. 247-256

Cheng,J. and Greiner,R. 1999. Comparing Bayesian network classifiers. *In Proceedings of UAI*.

Cheng,J., Bell,D. and Liu,W. 1997. Learning belief networks from data: An information theory based approach. *In Proceedings of CIMM*.

Chickering,D.M. 1996. Learning Bayesian networks is NP Complete. In Fisher,D. and Lenz,H., editors, *Learning from Data: Artificial Intelligence and Statistics V*, pp 121-130, Springer-Verlag.

Cooper, G.F. and Herskovits,E. 1992. A Bayesian method for the induction of probabilistic networks from data. *Journal of Machine Learning*, 9:309-347.

Domingos,P. and Pazzani, M.(1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning, 29, 103-130.*

Duda, R.O. and Hart, P.E. 1973. Pattern classification and scene analysis, New York: John Wiley and Sons.

Flesch, I. and Lucas, P.J.F. 2007. Markov equivalence in Bayesian networks. *Advances in Probabilistic Graphical Models*, Springer Berlin, pp 3-38.

Friedman,N., Geiger,D. and Goldszmidt,M. 1997. Bayesian network classifiers, *Journal of Machine Learning*, 29:131-163.

Kalisch, Z. and Bühlmann, P. 2007. Estimating high-dimensional directed acyclic graphs with the PC-Algorithm.*Journal of Machine Learning Research* 8: 613-636.

Komarek, P. and Moore, A. 2000. A dynamic adaptation of AD-Trees for efficient machine learning on large data sets, In Proceedings of ICML.

Langley,P.1995.: Order effects in incremental learning, In P.Reimann and H.Spada editors, Learning in humans and machines: Towards an Interdisciplinary Learning Science, Pergamon.

Meek,C. 1996. Causal inference and causal explanation with background knowledge. *Proc. Of 11th Uncertainty in Artificial Intelligence*, pp 403-410.

Moore,A. and Lee, M.S. 1998. Cached sufficient statistics for efficient machine learning with large datasets. Journal of Artificial Intelligence Research 8:67-91.

Pearl,J. 1988. *Probabilistic reasoning in intelligent systems*, Morgan Kaufmann.

Spirtes,P., Glymour,G. and Scheines,R. 2000. *Causation, Prediction, and Search*. The MIT Press, 2nd edition.

Verma,T. and Pearl, J. 2000. Equivalence and synthesis of causal models. *Proc. Of 6th Uncertainty in Artificial Intelligence*, pp 220-227.

Table 3. Algorithm specification of IPC-BNC and IPC-BNC++

```
RecognizePC ($T$ : target to scan, $D$ : Dataset, $\varepsilon$ :threshold )
{
01 cutSetSize = 0;
02 DO
03     Retrieve $ADJ_T$ from G ;     //Adjacent nodes of T in G
04     FOR (each $X_i \in ADJ_T$) DO
05         FOR (each $S \subseteq ADJ_T \setminus \{X_i\}$ with $|S|$ =cutSetSize) DO
06             IF( $I_D(X_i, T \mid S) \le \varepsilon$ )THEN
07                 Remove $T - X_i$ from G ;
08                 $S_{T,X_i} = S$ ;
09                 BREAK;
10     cutSetSize++;
11 WHILE($|ADJ_T| >$ cutSetSize)
}

IPC-BNC( $D$ : Dataset, $\varepsilon$ :threshold)
{ // Step1: Recognize T's parents/children
1  Scanned = $\phi$ ;
2  Init G by connecting    and all $X \in U \setminus \{T\}$;
3  RecognizePC( $T$ , $D$ , $\varepsilon$ );
4  PC = Parents/Children of T in G ;
5  Scanned = Scanned $\cup \{T\}$ ;
// Step 2: Remove false parents/children, and add spouse candidates.
6  FOR(each $X_i \in PC(T)$)DO
7      Connect $X_i$ with all $X_j \in X \setminus$ Scanned ;
8      RecognizePC( $X_i$ , $D$ , $\varepsilon$ );
9      Scanned = Scanned $\cup \{X_i\}$;
// Step 3: Remove false spouses by connection
10 $SP(T) = (\bigcup_{X_i \in PC} PC(X_i)) \setminus PC \setminus \{T\}$;
11 FOR(each $X_i \in SP(T)$) DO
12     RecognizePC( $X_i$ , $D$ , $\varepsilon$ );
// Step 4: Recognize true spouses by CI test
13 $SP(T)$ =Parents/Children of each $X_i \in PC(T)$ in G ;
14 FOR (each $X_i \in PC(T)$) DO
15     FOR (each $X_j \in PC(X_i)$) DO
16         IF( $I_D(T, X_j \mid S_{T,X_j} \cup \{X_i\}) > \varepsilon$ )THEN
17             $T \rightarrow X_i \leftarrow X_j$ in G ;
18         ELSE
19             $V_G = V_G \setminus X_j$ ;
//step 5: Abstract BNC from G
20 Remove all $X$ in G such that $X \notin PC(T)$ and $X \notin PC(PC(T))$,
   which gives us a BNC with skeleton and some oriented arcs.
// Step 6: Orientation (please refer to section Orientation Rules and More)
} return G ;
```