Structure Learning on Bayesian Networks

by

Nathanael Aff

San Francisco, California

December 2015

Contents

Chapter 1

Introduction

Bayesian networks are a subset of graphical models. Whereas graphical models can be used as generative models, Bayesian networks (BNs) are most commonly used to model the conditional dependence relations that hold between some set of variables. Variables are associated with nodes in a graph and the conditional dependencies are represented as directed arcs between nodes. Bayesian networks provide a way of combining previous knowledge about causal or dependence relations between variables and dependence relations that are learned from observational data.

In this paper I survey a number of algorithms used to learn the structure of BN. One of the challenges in doing so is the diverse uses of BNs. In some applications the purpose may be to learn the structure of the graph and to make what-if inferences about how changes one or more variables would affect a dependent variable, or the goal may be to classify observations based on the learned network. For example, Bayesian networks have been used in a range of applications from decision support

systems for health services to traffic flow [17]. In other contexts, the emphasis is on the structure of the dependencies in the learned graph itself, for example, in learning gene expression pathways or protein signaling path, the emphasis is not on inference but on learning probable causal pathways [2][10].

One of the drawbacks of using Bayesian networks for modeling is the complexity of the learning the network structure based on observational data which can be exponential in the number of variables. Some score-based methods for learning the optimal graph structure are only suitable for data with less than 30 variables, but PC-based structure learning algorithms, and a recent score-based approach, have been successfully run on thousands of variables. Although there are several dominant methods currently in use, the application domain, the number of variables being modeled, and whether the emphasis is on causal pathways or on more reliable inference, will likely affect what approach is most suitable.

After introducing basic terms and concepts of BNs we give short overview of the two most common approaches: constraint-based algorithms and score-based algorithms. The constraint-based methods are based on the PC/IC algorithms and we review two early formulations of the PC family of algorithms and some recent updates to the constraint-based approach. In next section we describe the basic score-based approach. There have been many elaborations on the basic algorithm and and a few of the most recent proposed modifications are discussed. These algorithms were designed to learn a static network, but with some modification

they can be extended to Dynamic Bayesian networks(DBN) which model time-based processes, and non-stationary DBN(nsDBN) a recent modification of DBNs meant to handle processes in which the graph structure of the BN changes over time. Finally, I look at an performance comparison of various implementations of structure learning algorithms on a data set with a small number of variables.

1.1 Terminology and notation

A graph G consists of a set of vertices $V = \{v_i\}$ and edges $E = \{e_{ij}\}$ connecting nodes (v_i, v_j). A graph is then $G = (V, E)$. For graphical models, a set of random variables X_i is associated with the nodes, where a single random variable X_i corresponds to node v_i.

The conditional independence relations in a BN are represented by directed edges E_{ij} where v_i is the parent node and v_j is the child node. An assumption of the Bayesian network model is that the graph representing the conditional distribution must be acyclic. The BN, then, is represented by a directed acyclic graph (DAG) with random variables associated to each node and the directed arc e_{ij} representing a conditional dependence relationship between x_i, x_j.

The set of parents of a node v_i corresponding to the variables on which x_i is conditionally dependent is represented as $\pi(i)$. The nodes associated with the parent set $\pi(i)$ is the Markov blanket, ∂v_i of the node v_i. These are the nodes such that $(v_i, v_j) \in E$.

The Markov assumption is that for a given graphical model, nodes are conditionally independent of nodes not in its Markov blanket:

$$Pr(v_i|\partial v_i \cap v_k) = Pr(v_i|\partial v_i).$$

The conditional independence of a variables X_i, X_j given a third variable (or set of variables) X_k will be denoted:

$$X_i \perp_P X_j|X_k$$

The graph independence of two nodes, on the other hand, will be written $X_i \perp_G |X_k$ [17]

1.1.1 Factorization

Let the set of the probability distributions over each node be denoted P and the parameters of P as Θ. A Bayesian Network is the a graph and the associated distribution of $\mathcal{B} = (G, \Theta)$. The probability distribution on a BN can be represented a set of conditional probability distributions(CPD) associated with each node, which give the probability of the node given it's parent set.

The Markov assumption permits a BN to be decomposed into sets of conditionally independent nodes. This makes for a combinatorially simpler representation of the probability distribution over the graph. That is, the conditional distribution P,

can be factored

$$Pr(x_1, ..., x_n) = \prod_i Pr(x_i|\pi(i))$$

1.1.2 Inference on Bayesian networks

Although inference on BNs is not reviewed here, we give a short introduction to motivate one of the common uses of BNs. Given a BN, $\mathcal{B}$, the network can be queried in several ways. The statistical independence of nodes can be checked either directly or conditional on another set of nodes, that is, we can check $X_i \perp_P X_j$ or $X_i \perp_P X_j|X_k$. The network can also be queried with a set of evidence $\mathcal{E}$ and a set of variables X_i of interest. The query asks for the probability of an event X_i given the evidence $\mathcal{E}$, that is: $Pr(X_i|\mathcal{E})$. This test can also be used to classify an observation with missing value by assigning the most likely value to the missing variable given the known set.

Chapter 2

Structure learning algorithms

2.1 The Inductive Causation (IC) algorithm

An early motivation for studying Bayesian networks was to describe causal relations between variables. A cause can be formalized as a directed arrow, where A causes B is written $A \rightarrow B$. It is in the context of this more general causal framework that T.S. Verma and Judea Pearl developed the Inductive Causation (IC) model described in [16].

This paper presents several key ideas used in other structure learning algorithms. First, the authors show what conditions are needed to determine that DAGs have the same causal (or independence) structures. These are the graph invariants of directed networks. To understand them we have to introduce two definitions.

Definition 2.1 (v-structure)**.** A v-structure is defined on three nodes where two

nodes have incoming directed arcs to the third. That is, for nodes v_i, v_j, v_k

$$v_i \rightarrow v_k \leftarrow v_j$$

then there is a v-structure at v_k.

Definition 2.2 (d-separation)**.** Given three disjoint subsets, A, B, C, of nodes in a DAG G, then the subset C d-separates A from B, that is $(A \perp_G B|C)$, if there exists a node v that satisfies one of the following conditions:

1. The node v is a v-structure and neither v nor any of its children that can be reached from v are in C.

2. The node v is in C and does not have any converging arcs.

[12]

Pearl and Verma show that two DAGs are equivalent if the have the same edges (irrespective of directedness) and the same v-structure. The proposed algorithm then uses d-separation to find v-structures in order to determine the structure of the DAG. The IC algorithm starts with a complete graph and successively prunes links by finding a set that d-separates each pair of nodes. In the following steps the directedness of the edges are determined based on the structure of the separating sets.

The concepts of DAG equivalence classes and the determined based on the undirected structure of the graph, or the DAG *skeleton*, and the v-structure of the DAG.

The algorithm's complexity is bound above by the (exponential) complexity of links on the the complete graph. The authors suggest reducing the complexity by first finding the Markov network (or Markov blanket), of every node. Since this contains all possible dependencies of a node the algorithm only needs to check relations between a node and members of its Markov blanket.

2.2 The PC algorithm

The PC algorithm, named after Peter Sprites and Clark Glymour, is another constraint-based algorithm that follows closely the steps of the IC algorithm Here I review the algorithm as presented in [13].

Like IC, the PC algorithm assumes that there is a DAG that faithfully represents the conditional independence structure of the data. The algorithm begins with the complete graph and sets undirected edges based on conditional dependence learned from the data. The algorithm is able to set a lower bound on complexity by iteratively checking for pairwise dependence relations of nodes v_i, v_j where the total adjacencies of v_i are less than a a set n which grows with each iteration.

This latter step allows the complexity to be bound by the greatest degree k of any node in the graph, since the algorithm runs for k iterations in the initial setting of undirected dependence relations between nodes. The bound on complexity is

roughly

$$\frac{n^2(n-1)^{k-1}}{(k-1)!}$$

, for a network with n variables and greatest node degree k. In practice, the author's say, the runtime is better than this upper complexity bound would suggests and versions of the algorithm have been used to learn BNs with thousands of nodes.

The algorithm can be modified to incorporate expert knowledge by setting known causal or relations (directed) edges or dependence (undirected) edges between variable nodes. The author also describe in more detail the implementation including the statistical test used for independence tests and possible methods of testing for estimating the goodness of fit of the learned model using simulations.

2.2.1 Causal interpretation of BN

The authors of both the previous papers speak in terms of causal discovery, claiming that the algorithm can serve as a basis for causal discovery from observational data. In particular, Pearl and Verma state that the structure discovered by the algorithm can serve as a time-independent picture of causal factors learned from data. This view of the power of a static Bayesian network overstates the case for learning causal structure from observational data. One of the fundamental assumptions of the DAG model of conditional independence is that their are no cycles. While the inter-dependence of variables may be well described by static dependence relation, this constraint would not model situations in which there are feedback loops among

the variables or time-dependent causal relations.

Another assumption of the causal interpretation of BNs is faithfulness – that any DAG representing the statistical dependency relations in the data is isomorphic to the learned DAG. Spirites *et al.* say this is sufficient for causal interpretations. This claim is not uncontroversial, and one of the primary mathematical foundations for this claim was recently challenged in [15]. The authors of the latter paper show that the probability of distributions violating versions of the faithfulness assumption is much greater than assumed by previous authors, who claimed the probability was vanishingly small.

A more cautious and common interpretation of BN is that information, including expert input, domain knowledge or data from randomized controlled experiments, are required to make inferences about causal factors [8][4]. Methods for estimating causal effects based on an unknown causal structure is also discussed in [8]. The methods, IDA and jointIDA, test effects of an intervention over a set of possible DAGs based on the the skeleton of the DAG.

2.2.2 Stability of the PC algorithm

As mentioned in the introduction, the PC and IC algorithm do not incorporate a global objective function which is optimized. Learning of the structure of the BN is done based on local structure and independence relations, rather than on overall fitness of the learned graph. Spirites *et al.* point out that since there is no back-

tracking mechanism, edges wrongly removed earlier in the algorithm may lead to edges not in the ideal graph being left in the graph at later stages. There is a similar issue when an edge removed early in the algorithm results, wrongly, in a change of directedness of an edge later in the graph. Both of these issues are related to the way in which the graph is pruned based on a local structure and without any backtracking. This can lead to situations where edges representing dependencies in the data can be removed later in the algorithm.

2.2.3 PC-stable

There have been a number of variations on the PC algorithm, some of which are reviewed in [2] and more briefly in [8]. The FCI and RFCI algorithms are modifications to the PC algorithm that allow for inclusion of hidden variables. These algorithms share the initial step of the PC algorithm – the conditional independence test which determine the skeleton of the DAG. While the PC algorithm was known to be somewhat order dependent, i.e., the skeleton of the learned DAG depended on the variable order, Maathuis *et al.* show that this dependency grows with the dimension of the data. The proposed modification to the PC algorithm, PC-stable, removes this dependency by only removing edges at each stage of the inner loop of the PC algorithm which determines the adjacency skeleton, rather than removing edges within the loop. The algorithm was tested on real and simulated high-dimensional data sets (5000+ variables) and showed decreased variance of per-

formance as measured on sets of synthetic data sets [2] The PC-stable algorithm is available in the R package `pcalg`.

2.3 Score-based methods

The other family of structure learning algorithms are are score-based algorithms. The principle of score-based algorithms is to search over the space of DAGs and use a scoring criterion to select the optimal DAG given the data. There are many variations, since the scoring criteria or the search procedure can be modified to give the algorithm different characteristics.

Heckerman *et al.* describe a score-based approach which includes many of the features still in use today [7]. The approach of this class of algorithms is generally Bayesian and users are able to place a prior on the structure of the network – or determine the directedness of certain links before hand. The search phase then looks for the network with the highest posterior probability. Unlike the previously mentioned papers, the authors clearly differentiate between a causal network and what they call a belief network. The latter only encodes conditional dependence relations between the variables, not necessarily causal relations. The authors introduce the a score-equivalent metric that applies to belief networks but not causal networks. The score-equivalent metric is simply a score that is equivalent for any Markov-equivalent(isomorphic) BN. The authors also present algorithms for exact structure learning and heuristic structure learning.

There are several assumptions that the BN must satisfy under this model, some of which can be relaxed – such as the requirement for a complete database (no missing observations). The score derived meets the requirement that it is equivalent for all isomorphic DAGs but it is applicable only to discrete data. A similar score, however, can be derived under the assumption that all variables follow a Gaussian distribution. The metric is termed the BDe metric (also called the BDeu metric) and is formed by putting a Dirichlet prior conjugate to the multinomial distribution of Θ associated with each node. Using Bayes rule, the posterior probability of the BN give the Data D is

$$P(D|G^h) = \frac{P(D|G^h)P(G^h)}{P(D)}$$

Using the Dirichlet priors the derived BDe metric is:

$$P(D|G^h) = \prod_i \prod_{\pi(i)} = \frac{\Gamma\left(\sum_i N'_{i,\pi(i)}\right)}{\Gamma\left(\sum_i N'_{i,\pi(i)} + N(\pi(i))\right)} \prod_{x_i} \frac{\Gamma\left(N'_{i,\pi(i)} + N(i,\pi(i))\right)}{\Gamma\left(N'_{i,\pi(i)}\right)}$$

Where G^h is the hypothesis G satisfies the independence relation represented in the learned graph, Γ is the gamma function and N' are the hyperparameters of the prior and $N(X)$ is the cardinality of the set X.

The authors describe two heuristic algorithms for searching the space of possible DAGs. Both use a set of possible changes ($\triangle(e)$) consisting of the addition, removal or direction reversal of edges for each pair of nodes. The two search methods are a greedy search and a simulated annealing method. The authors describe the results

of testing the algorithms where performance is measured against known or gold-standard model.

The concept of score equivalence and the scoring metric are still commonly used. The basic score-based approach is highly modular, and publicly available implementations allow users to vary the search method, the scoring method and the conditional independence tests. An overview of the variations is given in [17]. We look at a comparison of these implementations later in this survey.

2.4 Reducing the complexity of the search space

A drawback of the traditional score-based algorithms is the size of the search space. Even the non-exact or heuristic approaches to learning the max-score graph are complex in the number of variables or nodes. Two more recent variations address this by using using a search space other than the space of all DAGs. The Max-Min Hill-Climbing(MMHC) algorithm is a hybrid algorithm that uses the PC to find a skeleton of the DAG and score-based search is applied to this reduced space [8].

An alteration to the score-based method was proposed in [14], where the search space is over possible orderings of the variables – called ordering based search or OBS in [11]. Searching over the set of orderings reduces the search space to $2^{\mathcal{O}n \log n}$ compared to $2^{\Omega n^2}$ for the space of DAGS. Checks of acyclicity at each step are also avoided and the resulting modifications to the learned structure are more global, reducing the likelihood of being caught in local minima [14]. A paper based on

a search of the ordering space, termed the acyclic selection OBS or ASOBS, was recently described in [11]. The approach taken in this paper compares favorably both in speed and accuracy to other score-based approaches.

There are other structure learning approaches, some of which are gone over briefly in [17]. One of the more promising is a formulation of the structure learning problem as an integer linear program. A review of recent work in this direction can be found in [3].

2.5 Structure learning of dynamic Bayesian networks

One of the basic assumptions of Bayesian networks is that the dependence relations is acyclic and static. This implies there are no feedback relations among the data and that parameters describing dependencies is time-invariant. In many complex systems of interest these assumptions are unrealistic. Dynamic Bayesian networks (DBN) allow many of the techniques developed for BNs to be applied to systems with a time component.

The extension of structure learning of static Bayesian networks DBNs is relatively straight forward [5]. A DBN is a Bayesian network with an added time-transition model. The transition model is assumed to be stationary and dependent only on the state at time $t-1$. Given these assumptions, the DBN can be specified in a way similar to a BN, by an initial BN $\mathcal{B}_\prime = (G|\Theta)$ and a transition network $\mathcal{B}_\rightarrow$ that

specifies the transition probabilities for the variables $X_i[t]$:

$$Pr(X_i[t+1]|X_i[t])$$

Friedman *et al.* extend the BDe and BIC score to learning DBN in [9]. The data from which $\mathcal{B}_\prime$ and $\mathcal{B}_\rightarrow$ are learned consists of a set N_{seq} of sequences of length ℓ. The initial BN is learned from the ℓ initial sets and the transition network is learned from the set of sequences. In both cases, BIC or BDe can be used as scoring criteria.

As mentioned, DBN get around the original assumption that a BN must be a DAG by allowing the transition graph to model relation to previous states of a system. For basic DBN, however, it is assumed that the structure of the DAG and transition is stationary. In many practical cases, networks have non-stationary distributions – the conditional dependencies between variables in the network may change over time. Examples of such networks are traffic networks that change over time, ecological systems over time or transcriptional regulatory networks[9].

2.5.1 Non-stationary dynamic Bayesian networks

Robinson and Hartemik have extended DBN to cases with non-stationary transitions in [9]. The author's review related techniques that model the non-stationarity of either the parameter distribution or the structural distribution of the network. The algorithm presented takes the latter approach, allowing for addition, removals or

direction reversals of the edge structure of the BN.

The method presented builds on the approach to DBN described above. The author's term their algorithm nsDBN for non-stationary dynamic Bayesian network. The general method of score-based learning is also retained, with the BDe scoring criteria extended to this new model. The non-stationarity is modeled as a piece-wise change in the structure of the network. More formally, given some multivariate time series and a number of transition times $T = \{t_1, .., , , t_m\}$ there the graph G of dependence relation is allowed to have some finite number of changes $\triangle g_i$. The authors call the time between changes in the network structure an epoch. Both the number of possible changes $|\triangle g_i|$ and the number of total epochs are assumed to be distributed geometrically, and the maximum number of both epochs can be set as a parameter of the model. An assumption of the model is that both changes in graph structure and time between transitions are smooth and not excessively large.

To extend BDe as presented in [7] the author's assume that the parameters are independent between successive graphs G_i. The complexity of the resulting algorithm for building the initial graph is exponential in the number of variables and for each following stage it is exponential in maximum number of possible changes to the graph.

The authors test their method both on synthetic data and a subset gene expression data for *Drosophila* taken over the course of its development, with 66 or observation times. For the *Drosophila* data the only reference networks were static

networks or causal relations based on experimental data. Their results largely match previously constructed networks and found new relations between other genes that suggested connections that could be verified through experiment.

The algorithm in the paper has several forms based on the information known about the number of transitions and the time between the transitions. Values can be set for both parameters if they are known before hand, or they can be learned with the algorithm. In the text case on the *Drosophila* data set the the algorithm showed good performance where both these parameters were unknown. However, there were few transitions, 3-4, which corresponded to life stages of *Drosophila*. With testing on data sets with an unknown and greater number of transitions it might be easier to gauge the sensitivity of the algorithm to more transitions. The author's do demonstrate performance of the algorithm on data sets of up to 100 variables and 250,000 samples, but it isn't clear how well the algorithm scales to a greater number of variables.

Non-stationary Bayesian networks provide an good model for many social and biological systems and this appears to be an active area of research. For example, a related approach is taken in [6] but instead of a weakly-coupled graph structure between epochs, a method for modeling weak time-dependence between graph parameters is described.

Chapter 3

Performance

There is not a consensus on the optimal approach to learning Bayesian networks. The nature of the problem will likely drive the approach. The PC-stable algorithm developed in [2] has been applied to gene expression problems with thousands of variables. The PC approach avoids the complexity problems associated with searching over the space of DAGs, although the recent work in [11] means that score-based algorithms may be able to handle high-dimensional data.

For approaches with a smaller number of variables and where more precise inference is required, the score-based approaches or hybrid methods like the Max-Min Hill Climbing algorithm tend to be used.

3.1 Performance on few variables

A comparison of network learning algorithms for a small data set is carried out in [1]. This paper is part of a larger project that aims to develop a decision support system

aimed at managers of health service providers, such as hospitals. The paper is a preliminary study in which the author's evaluate available algorithms for learning Bayesian networks. The authors use data from an emergency health care service to compare the networks constructed by the algorithms and describe how the compared the performance of these respective networks.

The paper presents a practical application including preprocessing steps and a description of how the performance of different networks was rated. The authors evaluate four algorithms, two of which fit in the category of algorithms that build a network from conditional independence conditions, one score-based algorithm, and a hybrid method.

These four algorithms are 1) A version of the PC algorithm (PC).
2) The BN Power constructor (BNPC) a hybrid approach which constructs then prunes a skeleton based on conditional independence scores.
3) A score-based local search(LS) based on the algorithm described in [7] and uses the same Bayesian scoring criteria.
4) A second hybrid approach is a version of the BENEDICT algorithm (BE) and a scoring function is used to compare DAGs within the space of all equivalent DAGs.

There does not appear to be a widely used set of metrics for measuring network performance, although the (alarm) network is used in some papers to benchmark performance. In part, this is due to the wide number of domain areas and data

types and size to which Bayesian networks have been applied. Many of the papers examined demonstrated algorithms on data sets with a limited number of variables, typically less than 100. An exception was the PC-stable algorithm in [2] , which was tested on a data set of over 5,000 variables.

The algorithms were tested on a database with 11 variables with discretized data and about 33,000 instances. Another 12,000 instances were held out for testing purposes. The networks were built in under 60 seconds for each algorithm without a large discrepancy between each. The authors use two main tests to gauge the quality of the networks constructed by the four algorithms. The first set of test use Kullback-Leibler(KL) divergence to measure the discrepancy between the empirical distribution and the learned distribution of the network. Three other metrics are used which are essential Bayesian tests – BIC, BDeu and K2 – that probability of the data D given the graph G – $Pr(D|G)$.The score-based or local search(LS) method had the top score in three of four categories and was second in the fourth. The same set of scores was measured on the test set and the LS algorithm again had the top performance.

The authors next tested the performance of the networks as classifiers, by holding out individual variables and letting the networks classify the observation. In the classification task the BENEDICT (BE) and BNPC networks out performed the other two. In general, the (LS) network performed at the lower end (3rd or 4th) of most of the variable classification tasks. The LS algorithm was relearned with

different scoring metrics (BDeu, BIC, K2) but the performance of network learned under these metrics was still lower on the classification task. It was speculated that the part of the performance issue was related to the number of possible values of one of the categorical variables in the database.

The paper contributes an approach to measuring network performance based on classification accuracy of the network, and the discrepancy between the learned distribution of the network and the empirical distribution. The results show that scores for measuring the likelihood of the data given the structure do not necessarily match performance on the classification task. This result is a good warning to those working with learning algorithms: they should be tested against multiple criteria. On the other hand, the result is somewhat limited since, by design, the authors are only looking at results on a single data set. There is one suggestion relating to why performance of the score-based algorithm performed worse on a particular part of the network but it isn't clear if the result can be generalized.

Chapter 4

Summary

Bayesian networks have been applied to problems in everything from biology, management and ecology to finance. The optimal structure learning problem is NP-hard and heuristic approaches have been taken that use score-based, constraint-based or hybrid approaches. The constraint-based algorithms are faster but highly unstable for large numbers of variables, a problem that has been addressed in [2] with the PC-stable algorithm. An implementation based on PC-Stable is available in the *R* package `pcalg`. For score-based algorithms there does not appear to be a dominant approach. Implementations such as that in the *R* package `bnlearn` allow for a component based approach in which scoring, methods, search strategies, and conditional dependence tests can be determined by the user[12]. There are also algorithms aimed at learning optimal BN structures, including an integer linear programming approach currently available in an open version with the GOBNILP. For the score-based methods the most recent developments have been in simplifying

the search space of the structure by searching over orderings of the variables rather than DAG structures as in [11]. There are also a number of approaches to learning non-stationary Bayesian networks [9]. Learning change-points in the variable distributions and parameters on non-stationary networks shares characteristics of learning structure on static networks. A possible area of research is the extension of developments in using order-based search spaces to dynamic or non-stationary networks.

Bibliography

[1] S Acid, L M de Campos, J M Fernández-Luna, S Rodríguez, J María Rodríguez, and J Luis Salcedo, *A comparison of learning algorithms for Bayesian networks: a case study based on data from an emergency medical service*, Artificial Intelligence in Medicine **30** (2004), no. 3, 215–232.

[2] D Colombo and MH Maathuis, *Order-independent constraint-based causal structure learning*, The Journal of Machine Learning Research **15** (2014), no. 1, 3741–3782.

[3] J Cussens and M Bartlett, *Advances in Bayesian network learning using integer programming*, arXiv preprint arXiv:1309.6825 (2013).

[4] David A. Freedman, *Statistical Models and Causal Inference*, Cambridge University Press, New York, 2010.

[5] Nir Friedman, Kevin Murphy, and Stuart Russell, *Learning the structure of dynamic probabilistic networks*, Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence (1998), 139–147.

[6] M Grzegorczyk and D Husmeier, *A non-homogeneous dynamic Bayesian network with sequentially coupled interaction parameters for applications in systems and synthetic biology*, Statistical applications in genetics and ... (2012).

[7] David M. Heckerman, David; Geiger, Dan; Chickering, *Learning Bayesian networks: The combination of knowledge and statistical data*, Workshop on Knowledge Discovery in Databases (1994), 85–96.

[8] Marloes H. Maathuis and Preetam Nandy, *A review of some recent advances in causal inference*, Preprint (2015), 23.

[9] Jw Robinson and Aj Hartemink, *Learning non-stationary dynamic Bayesian networks*, The Journal of Machine Learning ... **11** (2010), 3647–3680.

[10] K Sachs, O Perez, and D Pe'er, *Causal protein-signaling networks derived from multiparameter single-cell data*, Science (2005).

[11] Mauro Scanagatta, Cassio P de Campos, Giorgio Corani, and Marco Zaffalon, *Learning Bayesian Networks with Thousands of Variables*, Advances in Neural Information Processing Systems 28 (C Cortes, N D Lawrence, D D Lee, M Sugiyama, R Garnett, and R Garnett, eds.), Curran Associates, Inc., 2015, pp. 1855–1863.

[12] Marco Scutari, *Bayesian Networks*, first ed., Chapman and Hall, 2014.

[13] Richard Spirtes, Peter; Glymour, Clark; Scheines, *Causation, Prediction, Search*, 2nd editio ed., Bradford Books, 2001.

[14] M Teyssier and D Koller, *Ordering-based search: A simple and effective algorithm for learning Bayesian networks*, arXiv preprint arXiv:1207.1429 (2012).

[15] Caroline Uhler, *Geometry of the Faithfulness Assumption in Causal Inference*, 2013.

[16] Judea Verma, TS; Pearl, *Technical report r-150*, Uncertainty in Artificial Intelligence **6** (1991), 255–268.

[17] Yang Zhou, *Structure Learning of Probabilistic Graphical Models : A Comprehensive Survey*, arXiv: 1111.6925 (2007).