Improved Bayesian Network Structure Learning in the Model Averaging Paradigm

by

Zhenyu A. Liao

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2022

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Cory J. Butz Professor & Associate Dean (Research), Faculty of Science University of Regina
Supervisor:	Peter van Beek Professor, David R. Cheriton School of Computer Science University of Waterloo
Internal Member:	Pascal Poupart Professor, David R. Cheriton School of Computer Science University of Waterloo
Internal Member:	Robin Cohen Professor, David R. Cheriton School of Computer Science University of Waterloo
Internal-External Member:	Krzysztof Czarnecki Professor, Electrical and Computer Engineering University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This thesis is mostly based on the following co-authored published articles. We use the taxonomy developed and refined by the Consortia Advancing Standards in Research Administration (CASRAI) and the National Information Standards Organization (NISO) to specify the individual contributions (see Table 0.1).

1. Zhenyu A. Liao, Charupriya Sharma, James Cussens, and Peter van Beek. Finding All Bayesian Network Structures within a Factor of Optimal. *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, Honolulu, Hawaii, January, 2019.

Author	Roles
Zhenyu A. Liao	Writing − original draft; Conceptualization; Investigation; Software; Visualization;
Charupriya Sharma	Conceptualization; Formal analysis; Software; Visualization; Writing − review/editing;
James Cussens	Software; Writing − review/editing;
Peter van Beek	Supervision; Funding acquisition; Conceptualization; Methodology; Writing − review/editing

2. Charupriya Sharma, Zhenyu A. Liao, James Cussens, and Peter van Beek. A Score-and-Search Approach to Learning Bayesian Networks with Noisy-OR Relations. *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, Aalborg, Denmark, September, 2020.

Author	Roles
Charupriya Sharma	Writing − original draft; Conceptualization; Formal analysis; Investigation; Methodology; Software; Validation; Visualization;
Zhenyu A. Liao	Software; Writing − review/editing;
James Cussens	Software; Writing − review/editing;
Peter van Beek	Supervision; Funding acquisition; Conceptualization; Methodology; Writing − review/editing

3. Zhenyu A. Liao, Charupriya Sharma, Dongshu Luo, and Peter van Beek. An Empirical Study of Scoring Functions for Learning Bayesian Networks in Model Averaging. *Proceedings of the 35th Canadian Conference on Artificial Intelligence*, Toronto, Ontario, May, 2022.

Author	Roles
Zhenyu A. Liao	Writing − original draft; Conceptualization; Formal analysis; Investigation; Methodology; Software;
Charupriya Sharma	Conceptualization; Investigation; Software; Writing − review/editing;
Dongshu Luo	Investigation; Software;
Peter van Beek	Supervision; Funding acquisition; Validation; Visualization; Writing − review/editing

4. Zhenyu A. Liao, Junyao Duan, and Peter van Beek. On Identifying Significant Edges for Structure Learning in Bayesian Networks. *Proceedings of the 35th Canadian Conference on Artificial Intelligence*, Toronto, Ontario, May, 2022.

Author	Roles
Zhenyu A. Liao	Writing − original draft; Conceptualization; Formal analysis; Investigation; Methodology; Software; Validation;
Junyao Duan	Formal analysis; Investigation; Software
Peter van Beek	Supervision; Funding acquisition; Conceptualization; Methodology; Software; Validation; Visualization; Writing − review/editing

Table 0.1: CRediT (Contributor Roles Taxonomy), a high-level taxonomy that can be used to represent the roles typically played by contributors to research outputs. The roles describe each contributor's specific contribution to the scholarly output.

Role	Description
Conceptualization	Ideas; formulation or evolution of overarching research goals and aims.
Formal analysis	Application of statistical, mathematical, computational, or other formal techniques to analyse or synthesize study data.
Funding acquisition	Acquisition of the financial support for the project leading to this publication.
Investigation	Conducting a research and investigation process, specifically performing the experiments, or data/evidence collection.
Methodology	Development or design of methodology; creation of models.
Software	Programming, software development; designing computer programs; testing of existing code components.
Supervision	Oversight and leadership responsibility for the research activity planning and execution.
Validation	Verification, whether as a part of the activity or separate, of the overall replication/reproducibility of results/experiments and other research outputs.
Visualization	Preparation, creation and/or presentation of the published work, specifically visualization/data presentation.
Writing − original draft	Preparation, creation and/or presentation of the published work, specifically writing the initial draft.
Writing − review/edit	Preparation, creation and/or presentation of the published work, specifically critical review, commentary or revision.

Abstract

A Bayesian network (BN) is a probabilistic graphical model with applications in knowledge discovery and prediction. Its structure can be learned from data using the well-known score-and-search approach, where a scoring function is used to evaluate the fit of a proposed BN to the data in an unsupervised manner, and the space of directed acyclic graphs is searched for the best-scoring BNs. However, selecting a single model (i.e., the best-scoring BN) is often not the best choice. When one is learning a BN from limited data, selecting a single model may be misleading as there may be many other BNs that have scores that are close to optimal, and the posterior probability of even the best-scoring BN is often close to zero. A more preferred alternative to committing to a single model is to perform some form of Bayesian or frequentist model averaging. A widely used data analysis methodology is to: (i) learn a set of plausible networks that fit the data, (ii) perform model averaging to obtain confidence measure for each edge, and (iii) select a threshold and report all edges with confidence higher than the threshold. In this manner, a representative network can be constructed from the edges that are deemed significant that can then be examined for probabilistic dependencies and possible cause-effect relations.

This thesis presents several improvements to Bayesian network structure learning that benefit the data analysis methodology. We propose a novel approach to model averaging inspired by performance guarantees in approximation algorithms. Our approach has two primary advantages. First, our approach only considers *credible* models in that they are optimal or near-optimal in score. Second, our approach is more efficient and scales to significantly larger Bayesian networks than existing approaches. We empirically study a selection of widely used and also recently proposed scoring functions. We address design limitations of previous empirical studies by scaling our experiments to larger BNs, comparing on an extensive set of both ground truth BNs and real-world datasets, considering alternative performance metrics, and comparing scoring functions on two model averaging frameworks: the bootstrap and the credible set. Contrary to previous recommendations based on finding a single structure, we find that for model averaging the BDeu scoring function is the preferred choice in most scenarios for the bootstrap framework and a recent score qNML is the preferred choice for the credible set framework. We identify an important shortcoming in a widely used threshold selection method. We then propose a simple transfer learning approach for maximizing target metrics and selecting a threshold that can be generalized from proxy datasets to the target dataset and show on an extensive set of benchmarks that it can perform significantly better than previous approaches. We demonstrate via ensemble methods that combining results from multiple scores significantly improve both the bootstrap and the credible set approach on various metrics, and that combining all scores from both approaches still yields better results.

Acknowledgements

I would like to thank all the people who made this thesis possible.

Dedication

This is dedicated to the one I love.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

A Bayesian Network (BN) is a probabilistic graphical model with applications in knowledge discovery, probabilistic density estimation, and prediction [26, 51, 68]. Recently, neural networks have largely surpassed BNs as a discriminative model for prediction, and normalizing flows (see, e.g., [74]) have become a more popular choice to model complex data distributions while allowing both for sampling and exact density estimation. Nevertheless, BN as a data analysis tool in knowledge discovery is still unmatched thanks to its clear interpretability and human readability.

BN is a generative model that represents conditional independence relationships and a joint distribution that factorizes over a directed acyclic graph (DAG), a graph consisting of vertices and directed edges with no cycles. Figure 1.1 shows the famous lung cancer diagnosis BN [52], and BNs are widely used as a data analysis tool in diverse areas, including finance, medicine, and sports (Table 1.1). A BN can be learned from data using the well-known *score-and-search* approach, where a scoring function is used to evaluate the fit of a proposed BN to the data in an unsupervised manner, and the space of directed acyclic graphs is searched for the best-scoring BNs. However, selecting a single model (i.e., the best-scoring BN) is often not the best choice. When one is learning a BN from limited data, selecting a single model may be misleading as there may be many other BNs that have scores that are close to optimal, and the posterior probability of even the best-scoring BN is often close to zero. A more preferred alternative to committing to a single model is to perform some form of Bayesian or frequentist model averaging (e.g., [34, 58, 63, 95]).

Our interest here is in BNs as a knowledge discovery or data analysis tool. In the context of knowledge discovery, model averaging allows one to estimate, for example, the posterior probability or degree of confidence, denoted $\hat{P}(e \mid \mathcal{D})$, that an edge e is present

Table 1.1: Representative data and causal analyses using BNs, where n is the number of random variables, and N is the number of instances in the dataset.

Area	n	N	Description
Banking	18	1,796	Contagion interactions between credit issuers following a sovereign default [4].
Biology	10	3,500	Semantic relationships between bacterial community properties in soil datasets [12].
Biology	12	1,900	Factors that directly impact red tide species occurrences and concentrations [32].
Medicine	11	79	Biophysical interactions of pneumonitis due to radiation therapy in lung cancer [62].
Medicine	17	120	Pathological interactions between diabetes mellitus and tuberculosis [73].
Medicine	26	408	Interactions between symptoms of obsessive-compulsive disorder and depression [64].
Safety	27	3,640	Relationships between interstate motor carrier characteristics and safety performance [45].
Software	21	12,630	Interactions between code review measures and prevalence of post-release defects [54].
Sports	22	377	Relationships between psychological features and team performance in football [35].

in the true network structure describing the dependence structure in the dataset $\mathcal{D}$, rather than just knowing whether the edge is present in the best-scoring network. A widely used data analysis methodology is to: (i) learn a set of plausible networks that fit the data $\mathcal{D}$, (ii) perform model averaging to obtain $\hat{P}(e \mid \mathcal{D})$ for each edge e, and (iii) select a threshold c and report all edges e with $\hat{P}(e \mid \mathcal{D}) > c$. In this manner, a representative network can be constructed from the edges that are deemed significant that can then be examined for probabilistic dependencies and possible cause-effect relations.

Structure learning algorithms can be generally categorized into one of the following three types, (i) constraint (statistical test) based, (ii) score-and-search, and (iii) hybrid of (i) and (ii). The most representative algorithm of (i) is the PC-stable algorithm [22] that stems from the original PC algorithm [90] after its authors, Peter and Clark. The algorithm starts by initializing a complete undirected graph spanning all nodes. Then a sequential

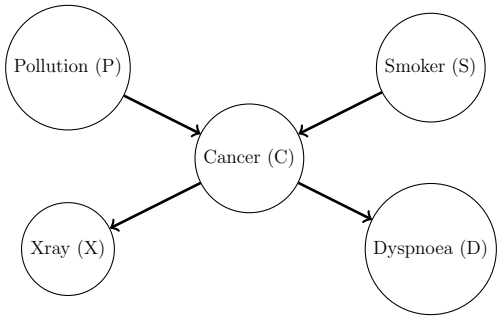


Figure 1.1: The DAG of the CANCER BN [52].

series of conditional independence tests is performed on pairs of nodes to determine if they can be separated by a subset of other nodes in the graph (usually d-separation, see Butz et al. [11] for recent developments). The next steps put a v-structure (triplets in graph where two parents point to the same child) on non-separable triplets, and the rest of the edges are oriented to avoid creating new v-structures. An overview of other heuristics that identify the separation can be found in Aliferis et al. [3]. The advantage of constraint based algorithms is that they can often achieve polynomial runtime in practice with reasonable assumptions (for example, sparsity of the graph), and thus can be applied to large number of nodes [48]. The drawback is that sequential application of statistical tests (commonly G-test or χ^2 test) has the potential for increased Type I error (false positive).

Hybrid algorithms first use similar statistical tests to identify a reduced set of candidate parents for all nodes, and then they operate score-and-search on the much reduced search space. Notable works of this type include Max-Min Hill Climbing algorithm (MMHC) from Tsamardinos et al. [96], 2-phase Restricted Maximization (RSMAX2) from Scutari et al. [83] and Hybrid HPC (H2PC) from Gasse et al. [37].

The predominant method for Bayesian network structure learning (BNSL) is the *score-and-search* method, which consists of identifying the space of BNs under consideration and measuring the goodness of fit between each structure. The method used to measure this fit is a structure scoring function. Let $\mathcal{D} = \{D_1, \ldots, D_N\}$ be a dataset where each instance D_i is an n-tuple that is a complete instantiation of the variables in $\mathcal{X}$. A *scoring function* $\sigma(\mathcal{D} \mid G)$ assigns a real value measuring the quality of the learned structure $G = (X, E)$ from the data $\mathcal{D}$. Most optimal algorithms consider the search space of all possible BNs organized by candidate parent sets for each variable. Since such a search space is $O(n^{\max_i |\Pi_i|})$, pruning techniques can be used to reduce the number of candidate parent sets that need to be considered [28]. Another approach is to limit the maximum in-degree to a small

number $d < \max_i |\Pi_i|$ so that the search space is reduced to $O(n^d)$. It has been shown that by setting $d = 2$ and using Bayesian information criterion (BIC), structure learning can stretch to thousands of variables [78].

Learning unrestricted BNs from data under typical scoring functions is NP-hard [20], and so local search has long been established as a practical alternative to optimal algorithms. Many local search algorithms have been proposed for different search spaces such as the space of DAGs, equivalence classes of DAGs [1], and topological orderings over the variables [44, 70], and for different search methods such as hill-climbing [36], tabu search [5], genetic algorithms [85], and simulated annealing [71]. In particular, the order-based local search (OBS) has consistently shown better performance over network structures in learning BN structures [94, 57, 79].

The majority of our focus in this thesis, however, is on optimal algorithms that can find the highest scoring structures. This family of algorithms include the dynamic programming approach [86], the A* approach [102], and the constraint programming approach (CPBayes) [98], all of which operate on the node ordering search space. Dynamic programming (DP) is perhaps the least efficient because it needs to fully evaluate the exponential search space without pruning or constraints. A* is a few times faster than DP and has been shown to extend to 30 nodes. Our work builds on the state-of-the-art optimal algorithm behind `GOBNILP` [24]. `GOBNILP` extends the SCIP Optimization Suite [38] by adding a *constraint handler* for handling the acyclicity constraint for DAGs and formulating the structure learning problem as a relaxed integer linear programming problem. `GOBNILP` has a similar effficiency as the CPBayes, and both of them have been shown to extend to networks of 70 nodes. Unlike the previous three algorithms, `GOBNILP` operates on the space of all DAGs.

As we have indicated in the beginning of this chapter, selecting a single model (i.e., the best-scoring BN) is often not the best choice. Previous work has proposed Bayesian and frequentist model averaging approaches to network structure learning that enumerate the space of all possible DAGs based on DP [50], sample from the space of all possible DAGs [40, 63], consider the space of all DAGs consistent with a given ordering of the random variables [8, 27], consider the space of tree-structured or other restricted DAGs [63, 66], and consider only the k-best scoring DAGs for some given value of k [14, 15, 16, 17, 40, 95]. Unfortunately, these existing approaches either severely restrict the structure of the BN, or have only been shown to scale to small BNs. For model averaging with structure constraints, Madigan and Raftery [63] and Meilă and Jaakkola [66] only allow tree-structured networks, and Dash and Cooper [27] only considers BNstructures consistent with a partial ordering and with bounded in-degree. For optimal model averaging, Koivisto and Sood [50] and He et al. [40] show that the DP approach and its derivatives can be

extended to 25 variables, and the k-best based approaches derived by Tian et al. [95] can be extended to 20 variables.

1.1 Contributions

The thesis primarily focuses on improvements to the exact algorithm of BNSL in the model averaging paradigm, and it also includes several improvements to the bootstrap approach, a commonly used sampling approach in BNSL.

- We propose a novel approach to model averaging inspired by performance guarantees in approximation algorithms. Our approach has two primary advantages. First, our approach only considers *credible* models in that they are optimal or near-optimal in score. Second, our approach is more efficient and scales to significantly larger BNs than existing approaches.
- We empirically study a selection of widely used and also recently proposed scoring functions. We address design limitations of previous empirical studies by scaling our experiments to larger BNs, comparing on an extensive set of both ground truth BNs and real-world datasets, considering alternative performance metrics, and comparing scoring functions on two model averaging frameworks: the bootstrap and the credible set. Contrary to previous recommendations based on finding a single structure, we find that for model averaging the likelihood-equivalence Bayesian Dirichlet score with uniform priors (BDeu) scoring function is the preferred choice in most scenarios for the bootstrap framework and a recent score called quotient normalized maximum likelihood (qNML) is the preferred choice for the credible set framework.
- We identify an important shortcoming in a widely used threshold selection method. We then propose a simple transfer learning approach for maximizing target metrics and selecting a threshold that can be generalized from proxy datasets to the target dataset and show on an extensive set of benchmarks that it can perform significantly better than previous approaches. We demonstrate via ensemble methods that combining results from multiple scores significantly improve both the bootstrap and the credible set approach on various metrics, and that combining all scores from both approaches still yields better results.

Chapter 2

Background

In this chapter, we review the necessary background in BNSL and the performance evaluation metrics used in this thesis. For more background on these topics see, for example, Darwiche [26], Koller and Friedman [51].

2.1 Probabilistic Graphical Models

BNs [68] belong to a large family of probabilistic graphical models (PGMs) that are used to compactly represent joint probability distributions and (in)dependence relations over a set of random variables. When the probabilistic influences among variables have clear directions, we can use BNs (DAGs) to formally represent such directions. On the other hand, when the directions are unclear, we can still use Markov networks (also called Markov random fields) (see, e.g., [51]) to encode conditional independence relationships.

Recently, a new type of PGM called sum-product network (SPN) [72] has been proposed to address the intractable inference issue of BNs at the cost of more complex structures. In particular, any factored probability distributions can be compiled into arithmetic circuits (equivalent to SPNs) [26] for tractable linear time inference. On the other hand, one can usually compile SPNs back to BNs for better interpretability and easier understanding of the underlying conditional independence relationships [10].

BNs have also been used as a tool for causal inference with increasing popularity in recent years [69]. Random variables in causal BNs are usually modeled by a structural equation model (SEM), and as a result the loss (scoring) function is continuous (e.g., least squares in the case of linear SEM) [104]. Such a formulation allows one to apply continuous

optimization techniques in learning the structure. However, the continuous formulation of the BNSL is not guaranteed to find the true graph, and it does not scale to large networks due to the acyclic constraint. This thesis focuses on the combinatorial formulation of the BNSL and we discuss it in Section 2.3.

The choice of PGMs and learning techniques in practice usually depends on the domain of problem, the amount of data, and perhaps most importantly, the intended usage.

2.2 Bayesian Networks

A BN is a probabilistic graphical model that consists of a labeled DAG, $G = (\mathcal{X}, E)$ in which the vertices $\mathcal{X} = \{X_1, \ldots, X_n\}$ correspond to n random variables, the edges E represent direct influence of one random variable on another, and each vertex X_i is labeled with a conditional probability distribution $P(X_i \mid \Pi_{X_i})$ that specifies the dependence of the variable X_i on its set of parents Π_{X_i} in the DAG G. A BN can alternatively be viewed as a factorized representation of the joint probability distribution over the random variables and as an encoding of the Markov condition on the nodes; i.e., given its parents, every variable is conditionally independent of its non-descendents.

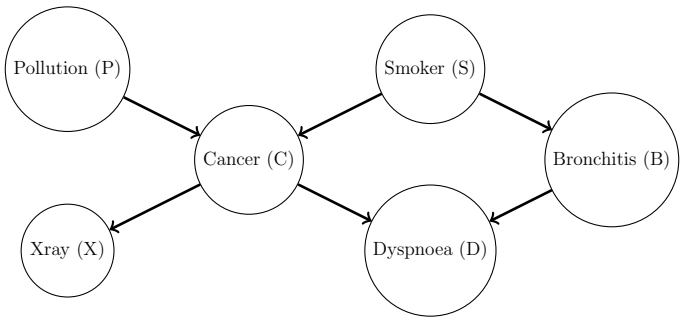


Figure 2.1: The DAG of the (augmented) CANCER BN [52] with an extra node (Bronchitis): Variables S, C, B, and D have the state space {TRUE, FALSE}. Variable P has the state space {low, high}, and variable X have state space {positive, negative}. Thus $r_P = r_S = r_C = r_B = r_X = r_D = 2$. Consider the parent set of C, $\Pi_C = \{B, C\}$ The state space of Π_C is $\Omega_{\Pi_C} =$ { {low, True}, {low, False}, {high, True}, {high, False} }, and $r_{\Pi_C} = 4$.

Figure 2.1 shows the (augmented) CANCER BN [52] with an extra node (Bronchitis). Each random variable X_i has state space $\Omega_i = \{v_{i1}, \ldots, v_{ir_i}\}$, where $r_i \geq 2$ is the car-

dinality of Ω_i and v_{ik} is the k-th value for vertex X_i. Each Π_{X_i} has state space $\Omega_{\Pi_{X_i}} = \{\pi_{i1}, \ldots, \pi_{ir_{\Pi_{X_i}}}\}$, where $r_{\Pi_{X_i}} \geq 2|\Pi_{X_i}|$ is the cardinality of $\Omega_{\Pi_{X_i}}$ and π_{ij} is the j-th vector of values for Π_{X_i} (see Figure 2.1). The set $\Theta = \{\theta_{ijk}\}$ for all $i = \{1, \ldots, n\}, j = \{1, \ldots, r_{\Pi_{X_i}}\}$ and $k = \{1, \ldots, r_i\}$ represents parameters in G where $\theta_{ijk} = P(v_{ik} \mid \pi_{ij})$. Given a uniform prior over the parameters, the expected value of $\theta_{ijk} = \frac{N_{ijk}+1}{\sum_{k=1}^{r_i} N_{ijk}+r_i}$ [23], which is generalized to the m-estimate [67] defined as $\widehat{\theta_{ijk}} = \frac{N_{ijk}+\frac{m}{r_i}}{\sum_{k=1}^{r_i} N_{ijk}+m}$.

The Markov blanket [68] of a node in a BN consists of its parents, children, and children's parents. For example in Figure 2.1, the Markov blanket for node C includes node P, S, X, D, and B. Given the Markov blanket of a node, then that node is independent of all other nodes in the BN.

2.3 Structure Learning

If the independence relations among random variables are unknown, an appropriate structure must be inferred from observable data. Such a process is referred to as BNSL [42] and is known to be NP-hard [19]. The predominant method for BNSL is the *score-and-search* method, which consists of identifying the space of BNs under consideration and measuring the goodness of fit between each structure. The method used to measure this fit is a structure scoring function. Let $\mathcal{D} = \{D_1, \ldots, D_N\}$ be a dataset where each instance D_i is an n-tuple that is a complete instantiation of the variables in $\mathcal{X}$. A *scoring function* $\sigma(\mathcal{D} \mid G)$ assigns a real value measuring the quality of the learned structure $G = (\mathcal{X}, E)$ from the data $\mathcal{D}$. Without loss of generality, we assume that a higher score represents a better quality network structure and omit $\mathcal{D}$ when the data is clear from context.

Definition 2.1 (The score-and-search objective)**.** *Suppose we have a scoring function $\sigma(\mathcal{D} \mid G)$ that assigns a real value measuring the quality of the learned structure $G = (\mathcal{X}, E)$ from the data $\mathcal{D}$, and a higher score represents a better quality network structure. The objective for the score-and-search method of BNSL is defined as,*

$$\underset{G \in \mathcal{G}}{\operatorname{argmax}}\, \sigma(\mathcal{D} \mid G), \tag{2.1}$$

where $\mathcal{G}$ is the set of all DAGs over $\mathcal{X}$.

Common scoring functions include BIC [55, 80] and BDeu [8, 42]. An important property of these (and most) scoring functions is decomposability, where the score of the entire

network $\sigma(G)$ can be aggregated as the sum of local scores associated to each vertex $\sum_{i=1}^{n} \sigma(\Pi_i)$ that only depends on X_i and its parent set Π_i in G. The full score-and-search procedure using a decomposable scoring function is summarized in Algorithm 2.1.

Optimal algorithms in BNSL consider the search space of all possible BNs defined over observed random variables that are categorized by candidate parent sets for each variable. Such algorithms will be able to identify the optimal BN given the observed random variables and a specific score. Since such a search space is $O(n^{\max_i |\Pi_i|})$, pruning techniques can be used to reduce the number of candidate parent sets that need to be considered on line 2 of Algorithm 2.1 [28]. Another approach is to limit the maximum in-degree to a small number $d < \max_i |\Pi_i|$ so that the search space is reduced to $O(n^d)$. It has been shown that by setting $d = 2$ and using BIC, structure learning can stretch to thousands of variables [78].

Algorithm 2.1: The score-and-search method for BNSL.

Input: A dataset $\mathcal{D} = \{D_1, \ldots, D_N\}$, where each instance D_i is an n-tuple that is a complete instantiation of the variables in $\mathcal{X}$.

Input: A scoring function $\sigma(\mathcal{D} \mid G)$ that assigns a higher score to a better DAG and can be decomposed to $\sigma(\mathcal{D} \mid \Pi_i)$.

Output: G^* with the highest score.

1 **for** $i \in \{1, 2, \cdots, n\}$ **do**
2 | Identify $\pi_{ij} \in \Pi_i$, the candidate parent sets for X_i;
3 **end**
4 Find $\pi_i^* := \text{argmax}_{\pi_{ij} \in \Pi_i} \sum_{i=1}^{n} \sigma(\mathcal{D} \mid \pi_{ij})$ subject to the acyclic constraint;
5 Construct DAG G^* defined by $\Pi := \{\pi_1^*, \pi_2^*, \cdots, \pi_n^*\}$.

Most BNSL algorithms use BIC and BDeu scoring functions. Both functions optimize the log likelihood of the training data being generated by the BN defined as,

$$LL(\mathcal{D} \mid G) = \sum_{i=1}^{N} \log P_G(D_i) = \sum_{i=1}^{n} \sum_{j=1}^{r_{\Pi_i}} \sum_{k=1}^{r_i} \log \theta_{ijk}^{n_{ijk}}. \tag{2.2}$$

BIC applies the metric on training data but avoids overfitting by adding a structure regularizing term,

$$\sigma_{\text{BIC}}(G) = -\max_{\Theta} LL(\mathcal{D} \mid G) + \frac{k(G) \log N}{2N}, \tag{2.3}$$

where $k(G) = \sum_{i=1}^{n} (|X_i| - 1) \prod_{X_j \in \Pi_{X_i}} |X_j|$ represents the complexity of G as the number of free parameters in the network.

BDeu directly calculates the log likelihood by assuming a uniform structure prior over the parameters,

$$\sigma_{\text{BDeu}}(G) = -\sum_{i=1}^{n}\sum_{j=1}^{r_{\Pi_i}} \log \frac{\Gamma(\alpha)}{\Gamma(\alpha + n_{ij})} - \sum_{i=1}^{n}\sum_{j=1}^{r_{\Pi_i}}\sum_{k=1}^{r_i} \log \frac{\Gamma(\frac{\alpha}{r_i} + n_{ijk})}{\Gamma(\frac{\alpha}{r_i})}, \tag{2.4}$$

where α is the equivalent sample size and $n_{ij} = \sum_{k=1}^{r_i} n_{ijk}$.

We can use a suitable scoring function to identify candidate parent sets for each node in the DAG and then search globally for the optimal parents-child pairs. The full procedure is summarized in Algorithm 2.1, and we provide an example in Example 2.1.

Example 2.1 (The score-and-search method for BNSL)**.** *We use the (augmented) CANCER BN [52] (Figure 2.1) as an example to illustrate the process of learning the parents for the node Dyspnoea. Suppose we have gathered N empirical observations to form a dataset $\mathcal{D}$ as shown in Table 2.1. Using a scoring function (e.g., BIC, BDeu), we can generate a score for each candidate structure as shown in Figure 2.2. Finally, we need to choose a parent set (which can be empty) for each node in the network, and all such parents-child local structures form the final DAG. The score of the DAG is the summation of all local scores. The search for the desired combination of local structures is the underlying combinatorial optimization problem behind BNSL.*

Pollution (P)	Smoker (S)	Cancer (C)	Xray (X)	Dyspnoea (D)	Bronchitis (B)
low	TRUE	TRUE	positive	FALSE	FALSE
low	FALSE	FALSE	negative	FALSE	FALSE
...	...	...	...	...	...

Table 2.1: An example of a dataset gathered from empirical observations.

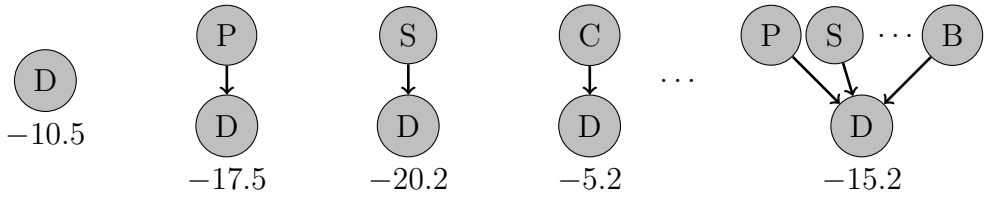


Figure 2.2: Some candidate parent sets for Dyspnoea (D) with illustrative scores.

2.4 Performance Evaluation Metrics

Scoring functions used by the score-and-search algorithms are often used as direct quality measures for BNSL with heuristic methods, i.e., one heuristic method is superior if it can find a structure with higher values for a specific score. However, if the search algorithm is exact (can identify the structures with the optimal value for every score), we cannot use them directly to assess performance. To evaluate the quality of the learned structures, we employ the following four metrics that operate on the edges, with considerations for directions and trade-offs regarding misidentified edges.

2.4.1 Structural Hamming Distance

To evaluate scoring functions on knowledge discovery, we use the structural Hamming distance (SHD). We need to define a few structural concepts to introduce the definition for SHD. The **skeleton** of a DAG G is the undirected graph with the same vertices and edges as G, and the **v-structures** are triplets in G where two parents point to the same child, e.g., $X_1 \to X_2 \leftarrow X_3$. A Markov equivalence class (MEC) is a set of DAGs that encode the same set of conditional independence relationships.

Theorem 2.1 (Verma and Pearl [99]). *Two DAGs belong to the same MEC if they have the same skeleton and v-structures.*

All DAGs in the same MEC can be uniquely represented by a completed partially directed acyclic graph (CPDAG) [65]. A directed edge $X_1 \to X_2$ in a CPDAG indicates the edge is present in every DAG in the MEC represented by this CPDAG. Such edges are also called **compelled** edges since it is irreversible in a MEC. For any undirected edge $X_3 - X_4$ in a CPDAG, the MEC contains a DAG with $X_3 \leftarrow X_4$ and a DAG with $X_3 \to X_4$. Such edges are not compelled because there exist two DAGs from the same MEC in which the edge has opposite orientation.

Example 2.2 (CPDAG and MEC). *The CPDAG of the (augmented) CANCER BN [52] (Figure 2.1) is shown in Figure 2.3.*

In particular, the v-structures $P \to C \leftarrow S$ and $C \to D \leftarrow B$ are present in the CPDAG; $C \to X$ cannot be reversed since it would otherwise form new v-structures with P and S. Note that this CPDAG represents the MEC for the two DAGs in Figure 2.4.

Given a ground truth network (predetermined target BN) and a learned network, the SHD measures the distance between the CPDAG representations of the networks, where a CPDAG captures the equivalence class to which a network belongs (see [96]).

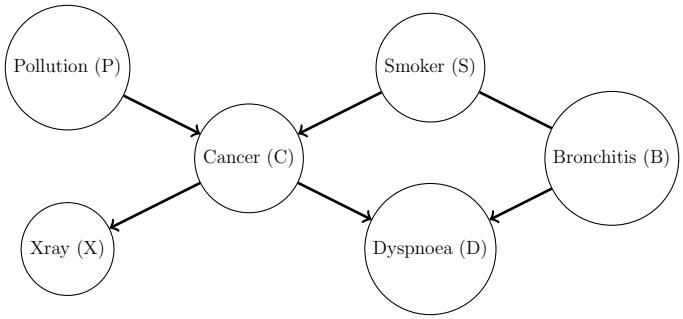


Figure 2.3: The CPDAG of the (augmented) CANCER BN [52].

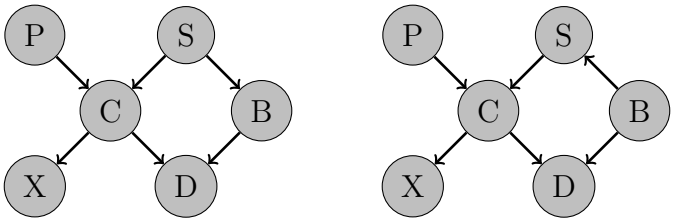


Figure 2.4: The two DAGs of the same MEC represented by the CPDAG in Figure 2.3.

Definition 2.2 (SHD [96]). *The SHD (H) between two CPDAGs, e.g., G_{CD_1} and G_{CD_2}, is the number of the following operations required for $G_{CD_1} = G_{CD_2}$,*

- *add or delete an undirected edge;*
- *add, remove, or reverse the orientation of an edge.*

Example 2.3 (SHD). *We show that the SHD between the two networks (left and right) in Figure 2.5 is 2. The two networks (top) can be represented by the two CPDAGs (bottom), and the SHD between the two CPDAGs is $H = 2$ because we need to modify the orientations of 2 edges, namely $X - C$ and $S - C$.*

Although the SHD has been widely used in evaluating structure learning, it has a number of significant limitations. First, it gives equal weight in case of a missing edge (FN), an extra edge (FP), and an edge in the wrong direction. However, in many applications of knowledge discovery, one does not wish to treat FP and FN as being of equal weight but rather wishes to specify an application-specific tradeoff. Second, adding an edge can

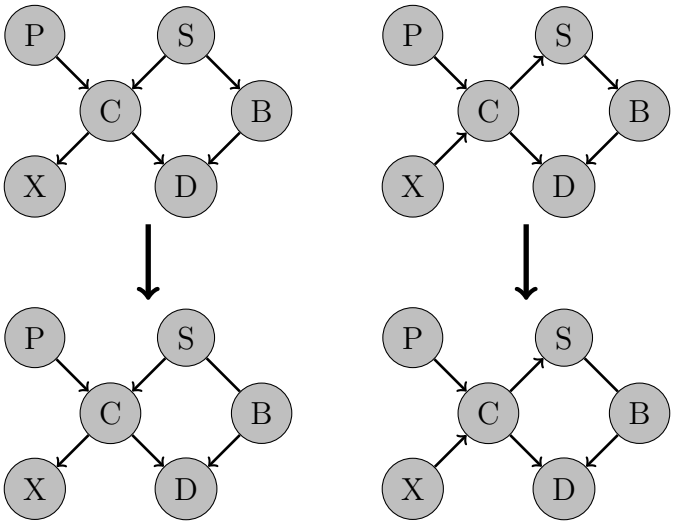


Figure 2.5: The two networks (top) can be represented by the two CPDAGs (bottom), and the SHD between the two CPDAGs is $H = 2$ because we need to modify the orientations of 2 edges, namely $X - C$ and $S - C$.

increase the SHD by more than 1 if it makes other edges not compelled anymore, and thus SHD tends to penalize FP more than FN. To address the limitations of SHD we will introduce two additional *cost sensitive* metrics: the F_β score and misclassification cost (see, e.g., [46]).

2.4.2 The F_β Score

F_β score is a generalization of the F_1 score, the harmonic mean of precision and recall. We use TP, FP, and FN to represent true positive, false positive, and false negative. Recall that precision = TP / (TP + FP) and recall = TP / (TP + FN).

Definition 2.3. *The F_β score is defined as,*

$$F_\beta = (1 + \beta^2) \times \frac{precision \times recall}{(\beta^2 \times precision) + recall}.$$

Note that the F_1 score is a special case when $\beta = 1$. When $\beta < 1$, the F_β measure gives more weight to precision and vice versa. These metrics are particularly useful for BNSL

since there is a large number of true negatives, i.e., both the ground truth BN and the learned BN are sparse with very few edges. We use two versions of the F_β score with one defined on the skeleton and the other on the CPDAG of the BN.

Binary F_β Score

Suppose we denote E as the set of true labels for the skeleton of the network, and $\hat{E}$ as the predictions. The precision and recall of the binary F_β score are defined as,

$$\text{precision} = \frac{|E \cap \hat{E}|}{|\hat{E}|}, \ \text{recall} = \frac{|E \cap \hat{E}|}{|E|}.$$

Follow Definition 2.3, we can calculate the binary F_β score that operates on the skeletons and produce a score for each network.

Multi-class F_β Score

The multi-class F_β score generalizes the binary F_β score to account for orientations in CPDAGs. Suppose that we denote E_d as the set of true labels for directed edges and E_u for undirected edges in a CPDAG and $\hat{E}_d$ and $\hat{E}_u$ for the corresponding predictions. The precision and recall of each class can be calculated in a similar way as the single class case above, and as a result we have a binary F_β score for each class, denoted F_β^d and F_β^u. The multi-class F_β score is the average of the F_β score of each class with weighting depending on the average parameter. The two most widely used averaging methods are "macro" ($\frac{F_\beta^d + F_\beta^u}{2}$) and "weighted" ($\frac{F_\beta^d \cdot |E_d| + F_\beta^u \cdot |E_u|}{|E_d| + |E_u|}$). If all of the edges in the ground truth network are undirected and the predicted CPDAG is identical then the F_β score should be 1. Since the macro approach does not consider the distribution of labels and produces undesirable results when all edges belong to one class, we decide to use the weighted variant in this thesis.

Definition 2.4. *The weighted multi-class F_β score is defined as,*

$$F_\beta^{CPDAG} = \frac{F_\beta^d \cdot |E_d| + F_\beta^u \cdot |E_u|}{|E_d| + |E_u|}.$$

Example 2.4 (F_β Score)**.** *Suppose we have the ground truth BN and a predicted BN shown in Figure 2.6.*

The binary F_β score can be determined with a precision of 1 and a recall of 1/6 since the only predicted edge is present in the ground truth and we successfully predicted 1 out of 6 edges, and it follows that $F_\beta = (1+\beta^2) \times \frac{1/6}{\beta^2+1/6}$.

For directed edges of the CPDAG, we have a precision and recall of 0 because we fail to predict any directed edges, whereas for undirected edges both precision and recall are 1 because we successfully predicted the only undrected edge $(S - B)$. *This results in* $F_\beta^d = 0$ *and* $F_\beta^u = 1$, *and it follows that* $F_\beta^{CPDAG} = \frac{1}{6}$.

If we were to adopt the "macro" version of the multi-class F_β score, we would arrive at a score of 0.5 since it does not account for class distributions. This would be too high given that 5 out of 6 edges are missing from the learned network.

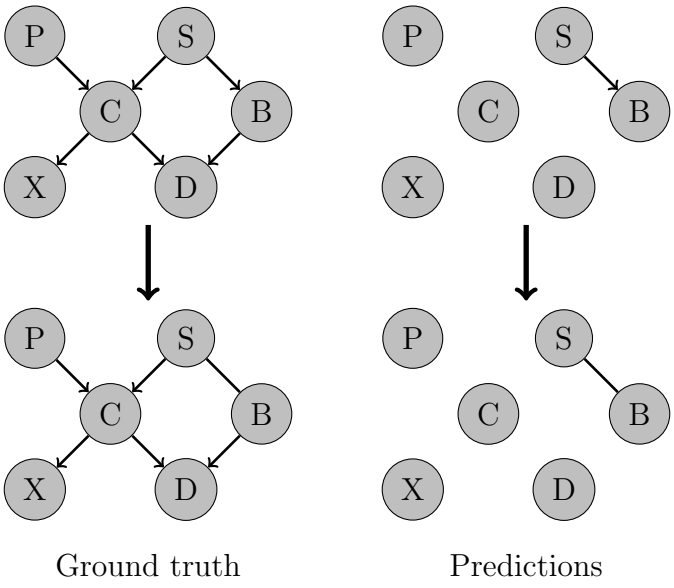


Figure 2.6: An example of ground truth network (left) and a predicted network (right). The two networks (top) can be represented by the two CPDAGs (bottom). The edge $S - B$ is undirected in both CPDAGs, and is the only correctly predicted edge. We use superscript u and d to indicate undirected and directed edges, and so $TP^u = 1, FP^u = 0, FN^u = 0, TP^d = 0, FP^d = 0, FN^d = 5$.

2.4.3 Misclassification Cost

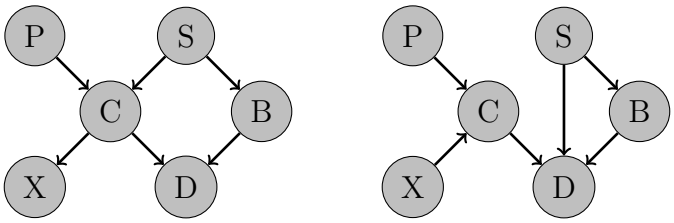
Definition 2.5. *The cost sensitive weighted misclassification cost [59] is defined as,*

$$C_\alpha = \alpha \times FN + FP,$$

where FN is the number of false negatives and FP is the number of false positives.

When $\alpha > 1$, the misclassification cost gives more weight to FN and thus penalizes missing edges over extra edges. The F_β score and the weighted misclassification cost allow us to evaluate the performance on a spectrum with different tradeoffs between precision and recall and between FP and FN. As we will show in our results, the rankings of scoring functions will fluctuate as different weights are place on FP and FN.

Example 2.5. *(Misclassification Cost) Suppose we have the ground truth BN and a predicted BN shown in Figure 2.7. We can observe that $FN = FP = 1$ since we missed the edge $S \to C$ but added the edge $S \to D$, and the misclassification cost is $C_\alpha = 1 + \alpha$.*



Ground truth Predictions

Figure 2.7: An example of ground truth network (left) and a predicted network (right). The predictions missed the edge $S \to C$ but added the edge $S \to D$, and therefore $FN = FP = 1$.

2.4.4 Borda Count

The experimental results are aggregated using the Borda count. Suppose we have k candidates to evaluate on a set of m instances using a chosen performance metric. For each instance (e.g., a fixed dataset) the candidates are ranked according to the performance metric with ties allowed, and each candidate is awarded points corresponding to the number of opponents strictly lower in the ranking. Thus, the lowest ranked candidate always

gets 0 points and the highest ranked candidate gets at most $k-1$ points (exactly $k-1$ if there are no ties for highest ranked). Points from each instance are added together in the end and the highest possible points would be $m \cdot (k-1)$. The Borda count was chosen to aggregate the results as it is known to select broadly acceptable options.

Example 2.6 (Borda Count)**.** *Suppose we want to compare three approaches A_1, A_2, and A_3 using a metric, and that we can get a metric value using each approach on three instances I_1, I_2 and I_3. In Table 2.2, we can observe that on I_1, A_3 is awarded 2 points for beating A_1 and A_2, and A_2 is awarded 1 point for beating A_1; on I_2, both A_2 and A_3 are awarded 1 point for beating A_1; on I_3 no approach gets any points. Finally, A_1 has 0 points, A_2 has 2 points, and A_3 has 3 points, and therefore A_3 is the winner.*

Instance	I_1			I_2			I_3		
Candidate	A_1	A_2	A_3	A_1	A_2	A_3	A_1	A_2	A_3
Metric value	0.0	0.5	0.8	0.2	0.4	0.4	0.5	0.5	0.5
Points	0	1	2	0	1	1	0	0	0

Table 2.2: An example of using Borda count to evaluate candidates on a set of instances using a metric.

Chapter 3

The Credible Set Approach

A Bayesian network is a widely used probabilistic graphical model with applications in knowledge discovery, density estimation, and prediction [26, 51]. Learning a BN from data can be cast as an optimization problem using the well-known score-and-search approach. However, selecting a single model (i.e., the best scoring BN) can be misleading or may not achieve the best possible accuracy. An alternative to committing to a single model is to perform some form of Bayesian or frequentist model averaging, where the space of possible BNs is sampled or enumerated in some fashion. Unfortunately, existing approaches for model averaging either severely restrict the structure of the Bayesian network or have only been shown to scale to networks with fewer than 30 random variables. In this chapter, we propose a novel approach to model averaging inspired by performance guarantees in approximation algorithms. Our approach has two primary advantages. First, our approach only considers *credible* models in that they are optimal or near-optimal in score. Second, our approach is more efficient and scales to significantly larger Bayesian networks than existing approaches. We note that boostrapping is a commonly used sampling approach in BNSL and we compare it with the credible set in Chapter 5.

3.1 Introduction

A BN can be learned from data using the well-known *score-and-search* approach, where a scoring function is used to evaluate the fit of a proposed BN to the data, and the space of DAGs is searched for the best-scoring BN. However, selecting a single model (i.e., the best-scoring BN) may not always be the best choice. When one is using BNs for knowledge discovery and explanation with limited data, selecting a single model may be misleading

as there may be many other BNs that have scores that are very close to optimal and the posterior probability of even the best-scoring BN is often close to zero. As well, when one is using BNs for prediction, selecting a single model may not achieve the best possible accuracy.

An alternative to committing to a single model is to perform some form of Bayesian or frequentist model averaging [21, 43, 51]. In the context of knowledge discovery, Bayesian model averaging allows one to estimate, for example, the posterior probability that an edge is present, rather than just knowing whether the edge is present in the best-scoring network. Previous work has proposed Bayesian and frequentist model averaging approaches to network structure learning that enumerate the space of all possible DAGs [50], sample from the space of all possible DAGs [40, 63], resample the data and learn a collection of DAGs [34, 81], consider the space of all DAGs consistent with a given ordering of the random variables [8, 27], consider the space of tree-structured or other restricted DAGs [63, 66], and consider only the k-best scoring DAGs for some given value of k [14, 15, 16, 17, 40, 95]. Unfortunately, these existing approaches either severely restrict the structure of the Bayesian network, such as only allowing tree-structured networks or only considering a single ordering, or have only been shown to scale to small Bayesian networks with fewer than 30 random variables.

In the remainder of this chapter, we propose a novel approach to model averaging for BN structure learning that is inspired by performance guarantees in approximation algorithms. Let OPT be the score of the optimal BN and assume without loss of generality that the optimization problem is to find the minimum-score BNInstead of finding the k-best networks for some fixed value of k, we propose to find all Bayesian networks $\mathcal{G}$ that are within a factor ρ of optimal; i.e.,

$$OPT \leq score(\mathcal{G}) \leq \rho \cdot OPT, \tag{3.1}$$

for some given value of $\rho \geq 1$, or equivalently,

$$OPT \leq score(\mathcal{G}) \leq OPT + \epsilon, \tag{3.2}$$

for $\epsilon = (\rho - 1) \cdot OPT$. Instead of choosing arbitrary values for ϵ, $\epsilon \geq 0$, we show that for the two scoring functions BIC/MDL and BDeu, a good choice for the value of ϵ is closely related to the Bayes factor (BF), a model selection criterion summarized by Kass and Raftery [49].

Our approach has two primary advantages. First, our approach only considers *credible* models in that they are optimal or near-optimal in score. Approaches that enumerate or sample from the space of all possible models consider DAGs with scores that can be far from

optimal; for example, for the BIC/MDL scoring function the ratio of worst-scoring to best-scoring network can be four or five orders of magnitude[1]. A similar but more restricted case can be made against the approach which finds the k-best networks since there is no *a priori* way to know how to set the parameter k such that only credible networks are considered. Second, and perhaps most importantly, our approach is significantly more efficient and scales to Bayesian networks with almost 60 random variables. Existing methods for finding the optimal Bayesian network structure, e.g., [6, 98] rely heavily for their success on a significant body of pruning rules that remove from consideration many candidate parent sets both before and during the search. We show that many of these pruning rules can be naturally generalized to preserve the Bayesian networks that are within a factor of optimal. We modify `GOBNILP` [6], a state-of-the-art method for finding an optimal Bayesian network, to implement our generalized pruning rules and to find all *near*-optimal networks. We show in an experimental evaluation that the modified `GOBNILP` scales to significantly larger networks without resorting to restricting the structure of the Bayesian networks that are learned.

3.2 Credible Bayesian Networks

The predominant method for BNSL from data is the *score-and-search* method. Let $\mathcal{D} = \{D_1, \ldots, D_N\}$ be a dataset where each instance D_i is an n-tuple that is a complete instantiation of the variables in $\mathcal{X}$. A *scoring function* $\sigma(G \mid \mathcal{D})$ assigns a real value measuring the quality of $G = (X, E)$ given the data $\mathcal{D}$. Without loss of generality, we assume that a lower score represents a better quality network structure and omit $\mathcal{D}$ when the data is clear from context.

Definition 3.1 (Credible Bayesian networks)**.** *Given a non-negative constant ϵ and a dataset $\mathcal{D} = \{D_1, \ldots, D_N\}$, a credible Bayesian network G is a network that has a score $\sigma(G)$ such that $OPT \leq \sigma(G) \leq OPT + \epsilon$, where OPT is the score of the optimal Bayesian network.*

In this chapter, we focus on finding all credible networks given ϵ, and we call it ϵ — Bayesian network structure learning (ϵBNSL). Note that the BNSL for the optimal network(s) is a special case of ϵBNSL where $\epsilon = 0$.

[1] Madigan and Raftery [63] deem such models *discredited* when they make a similar argument for not considering models whose probability is greater than a factor from the most probable.

Definition 3.2 (ϵBNSL)**.** *Given a non-negative constant ϵ, a dataset $\mathcal{D} = \{D_1, \ldots, D_N\}$ over random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ and a scoring function $\sigma()$, the ϵBNSL problem is to find all credible Bayesian networks defined in Definition 3.1.*

3.3 Scoring Functions

Scoring functions usually balance goodness of fit to the data with a penalty term for model complexity to avoid overfitting. Common scoring functions include BIC/MDL [56, 80] and BDeu [8, 42]. An important property of these (and most) scoring functions is decomposability, where the score of the entire network $\sigma(G)$ can be rewritten as the sum of local scores associated to each vertex $\sum_{i=1}^{n} \sigma(X_i, \Pi_i)$ that only depends on X_i and its parent set Π_i in G. The local score is abbreviated below as $\sigma(\Pi_i)$ when the local node X_i is clear from context. Pruning techniques can be used to reduce the number of candidate parent sets that need to be considered, but in the worst-case the number of candidate parent sets for each variable X_i is exponential in n, where n is the number of vertices in the DAG.

In this chapter, we only focus on the two commonly used scoring functions in BNSL, BIC and BDeu. The BIC scoring function in this chapter is defined as,

$$\sigma_{\text{BIC}}(G) = \max_{\Theta} LL(\mathcal{D} \mid G; \Theta) - \mathbf{pen}_{\text{BIC}}(G). \tag{3.3}$$

Here, $\mathbf{pen}_{\text{BIC}}(G) = \sum_{i=1}^{n} r_{\Pi_i}(r_i - 1)\frac{\log N}{2}$ is a penalty term, and $LL(\mathcal{D} \mid G; \Theta)$ is the log likelihood, given by,

$$LL(\mathcal{D} \mid G; \Theta) = \log \prod_{i=1}^{N} P(D_i \mid G; \Theta) = \sum_{i=1}^{n} \sum_{j=1}^{r_{\Pi_i}} \sum_{k=1}^{r_i} n_{ijk} \log \theta_{ijk},$$

where n_{ijk} is the number of instances in $\mathcal{D}$ where v_{ik} and π_{ij} co-occur. As the BIC function is decomposable, we can associate a score to Π_i, a candidate parent set of X_i as follows,

$$\sigma_{\text{BIC}}(\Pi_i) = \max_{\theta_i} LL(\mathcal{D} \mid \Pi_i; \theta_i) - \mathbf{pen}_{\text{BIC}}(\Pi_i).$$

Here, $LL(\mathcal{D} \mid \Pi_i; \theta_i) = \sum_{j=1}^{r_{\Pi_i}} \sum_{k=1}^{r_i} n_{ijk} \log \theta_{ijk}$ and $\mathbf{pen}_{\text{BIC}}(\Pi_i) = r_{\Pi_i}(r_i - 1)\frac{\log N}{2}$. The BDeu scoring function in this chapter is defined as,

$$\sigma_{\text{BDeu}}(G) = \sum_{i=1}^{n} \sum_{j=1}^{r_{\Pi_i}} \frac{\sum_{k=1}^{r_i} \log \Gamma(\frac{\alpha}{r_i r_{\Pi_i}} + n_{ijk})}{\log \Gamma(\frac{\alpha}{r_{\Pi_i}} + n_{ij*})} - \frac{\sum_{k=1}^{r_i} \log \Gamma(\frac{\alpha}{r_i r_{\Pi_i}})}{\log \Gamma(\frac{\alpha}{r_{\Pi_i}})}, \tag{3.4}$$

where α is the equivalent sample size and $n_{ij*} = \sum_k n_{ijk}$. As the BDeu function is decomposable, we can associate a score to Π_i, a candidate parent set of X_i as follows,

$$\sigma_{\text{BDeu}}(\Pi_i) = \sum_{j=1}^{r_{\Pi_i}} \frac{\sum_{k=1}^{r_i} \log \Gamma(\frac{\alpha}{r_i r_{\Pi_i}} + n_{ijk})}{\log \Gamma(\frac{\alpha}{r_{\Pi_i}} + n_{ij*})} - \frac{\sum_{k=1}^{r_i} \log \Gamma(\frac{\alpha}{r_i r_{\Pi_i}})}{\log \Gamma(\frac{\alpha}{r_{\Pi_i}})}.$$

3.4 The Bayes Factor

In this section, we show that a good choice for the value of ϵ for the ϵBNSL problem is closely related to the BF, a model selection criterion summarized by Kass and Raftery [49].

The BF was proposed by Jeffreys as an alternative to significance test [47]. It was thoroughly examined as a practical model selection tool in [49].

Definition 3.3. *Let G_0 and G_1 be two DAGs in the set of all DAGs $\mathcal{G}$ defined over a set of variables X. The BF in the context of DAG is defined as,*

$$BF(G_0, G_1) = \frac{P(\mathcal{D} \mid G_0)}{P(\mathcal{D} \mid G_1)},$$

namely the odds of the probability of the data predicted by network G_0 and G_1.

The actual calculation of the BF often relies on Bayes' Theorem as follows,

$$\frac{P(G_0 \mid \mathcal{D})}{P(G_1 \mid \mathcal{D})} = \frac{P(\mathcal{D} \mid G_0)}{P(\mathcal{D} \mid G_1)} \cdot \frac{P(G_0)}{P(G_1)} = \frac{P(\mathcal{D}, G_0)}{P(\mathcal{D}, G_1)}.$$

Since it is typical to assume the prior over models is uniform in BNSL, the BF can then be obtained using either $P(G \mid \mathcal{D})$ or $P(\mathcal{D}, G) \forall G \in \mathcal{G}$. We use those two representations to show how BIC and BDeu scores relate to the BF.

Using Laplace approximation and other simplifications in [75], Ripley derived the following approximation to the logarithm of the marginal likelihood for network G (a similar derivation is given in [21]),

$$\begin{aligned}\log P(\mathcal{D} \mid G) =& LL(\mathcal{D} \mid G; \hat{\theta}) - r_{\Pi_i}(r_i - 1)\frac{\log N}{2} + r_{\Pi_i}(r_i - 1)\frac{\log 2\pi}{2} \\ & - \frac{1}{2}\log |J_{G,I}(\hat{\theta})| + \log P(\hat{\theta} \mid G),\end{aligned}$$

where $\hat{\theta}$ is the maximum likelihood estimate of model parameters and $J_{G,I}(\hat{\theta})$ is the Hessian matrix evaluated at $\hat{\theta}$. It follows that,

$$\log P(\mathcal{D} \mid G) = \text{BIC}(\mathcal{D}, G) + O(1).$$

The above equation shows that the BIC score was designed to approximate the log marginal likelihood. If we drop the lower-order term, we can then obtain the following equation,

$$\text{BIC}(\mathcal{D}, G_1) - \text{BIC}(\mathcal{D}, G_0) = \log \frac{P(\mathcal{D} \mid G_0)}{P(\mathcal{D} \mid G_1)} = \log \text{BF}(G_0, G_1).$$

It has been indicated in [49] that as $N \to \infty$, the difference of the two BIC scores, dubbed the Schwarz criterion, approaches the true value of $\log \text{BF}$ such that,

$$\frac{\text{BIC}(\mathcal{D}, G_1) - \text{BIC}(\mathcal{D}, G_0) - \log \text{BF}(G_0, G_1)}{\log \text{BF}(G_0, G_1)} \to 0.$$

Therefore, the difference of two BIC scores can be used as a rough approximation to $\log \text{BF}$. Note that some papers define BIC to be twice as large as the BIC defined in this chapter, but the above relationship still holds albeit with twice the logarithm of the BF.

Similarly, the difference of the BDeu scores can be expressed in terms of the BF. In fact, the BDeu score is the log marginal likelihood where there are Dirichlet distributions over the parameters [8, 42]; i.e.,

$$\log P(\mathcal{D}, G) = \text{BDeu}(\mathcal{D}, G),$$

and thus,

$$\text{BDeu}(\mathcal{D}, G_1) - \text{BDeu}(\mathcal{D}, G_0) = \log \frac{P(\mathcal{D}, G_0)}{P(\mathcal{D}, G_1)} = \log (G_{\mathbf{0}}, G_{\mathbf{1}}).$$

The above results are consistent with the observation by Kass and Raftery [49] that the $\log \text{BF}$ can be interpreted as a measure for the *relative success* of two models at predicting data, sometimes referred to as the "weight of evidence", without assuming either model is true. The desired value of BF, however, is often specific to a study and determined with domain knowledge, e.g., a BF of 1000 is more appropriate in forensic science. Heckerman et al. [42] proposed the following interpreting scale for the BF: a BF of 1 to 3 bears only anecdotal evidence, a BF of 3 to 20 suggests some positive evidence that G_0 is better, a BF of 20 to 150 suggests strong evidence in favor of G_0, and a BF greater than 150 indicates very strong evidence. If we deem 20 to be the desired BF in ϵBNSL, i.e., $G_0 = G^*$

and $\epsilon = \log(20)$, then any network with a score less than $\log(20)$ away from the optimal score would be *credible*, otherwise it would be *discredited*. Note that the ratio of posterior probabilities was defined as λ by Tian et al. [95], Chen and Tian [17] and was used as a metric to assess arbitrary values of k in finding the k-best networks.

Finally, the ϵBNSL problem using the BIC or BDeu scoring function given a desired BF can be written as,

$$OPT \leq score(\mathcal{G}) \leq OPT + \log \text{BF}. \tag{3.5}$$

3.5 Pruning Rules for Candidate Parent Sets

To find all near-optimal BNs given a BF, the local score $\sigma(\Pi_i)$ for each candidate parent set $\Pi_i \subseteq 2^{\mathcal{X}-\{X_i\}}$ and each random variable X_i must be computed. As this is very cost prohibitive, the search space of candidate parent sets can be pruned, provided that global optimality constraints are not violated.

A candidate parent set Π_i can be *safely pruned* given a non-negative constant $\epsilon \in \mathbb{R}^+$ if Π_i cannot be the parent set of V_i in any network in the set of credible networks. Note that for $\epsilon = 0$, the set of credible networks just contains the optimal network(s). We discuss the original rules and their generalization below and proofs for them can be found in the *extended version*.

Teyssier and Koller [94] gave a pruning rule for all decomposable scoring functions. This rule compares the score of a candidate parent set to those of its subsets. We give a relaxed version of the rule.

Lemma 3.1. *Given a vertex variable X_j, candidate parent sets Π_j and Π'_j, and some $\epsilon \in \mathbb{R}^+$, if $\Pi_j \subset \Pi'_j$ and $\sigma(\Pi_j) + \epsilon \geq \sigma(\Pi'_j)$, Π'_j can be safely pruned.*

3.5.1 Pruning with BIC/MDL Score

A pruning rule comparing the BIC score and penalty associated to a candidate parent set to those of its subsets was introduced in [28]. The following theorem gives a relaxed version of that rule.

Theorem 3.1. *Given a vertex variable X_j, candidate parent sets Π_j and Π'_j, and some $\epsilon \in \mathbb{R}^+$, if $\Pi_j \subset \Pi'_j$ and $\sigma(\Pi_j) - \boldsymbol{pen}(\Pi'_j) + \epsilon < 0$, Π'_j and all supersets of Π'_j can be safely pruned if σ is the BIC function.*

Another pruning rule for BIC appears in [28]. This provides a bound on the number of possible instantiations of subsets of a candidate parent set. The following theorem relaxes that rule.

Theorem 3.2. *Given a vertex variable V_i, and a candidate parent set Π_i such that $r_{\Pi_i} > \frac{N}{w}\frac{\log r_i}{r_i - 1} + \epsilon$ for some $\epsilon \in \mathbb{R}^+$, if $\Pi_i \subsetneq \Pi_i'$, then Π_i' can be safely pruned if σ is the BIC scoring function.*

The following corollary of Theorem 3.2 gives a useful upper bound on the size of a candidate parent set.

Corollary 3.1. *Given a vertex variable X_i and candidate parent set Π_i, if Π_i has more than $\lceil \log_2 N + \epsilon \rceil$ elements, for some $\epsilon \in \mathbb{R}^+$, Π_i can be safely pruned if σ is the BIC scoring function.*

Corollary 3.1 provides an upper-bound on the size of parent sets based solely on the sample size. The following table summarizes such an upper-bound given different amounts of data N and a BF of 20.

N	100	500	10^3	5×10^3	10^4	5×10^4	10^5
$\lvert\Pi\rvert$	10	12	13	16	17	19	20

The entropy of a candidate parent set is also a useful measure for pruning. A pruning rule, given by [29], provides an upper bound on conditional entropy of candidate parent sets and their subsets. We give a relaxed version of their rule. First, we note that entropy for a vertex variable X_i is given by,

$$H(X_i) = -\sum_{k=1}^{r_i} \frac{n_{ik}}{N} \log \frac{n_{ik}}{N},$$

where n_{ik} represents how many instances in the dataset contain v_{ik}, where v_{ik} is an element in the state space Ω_i of X_i. Similarly, entropy for a candidate parent set Π_i is given by,

$$H(\Pi_i) = -\sum_{j=1}^{r_{\Pi_i}} \frac{n_{ij}}{N} \log \frac{n_{ij}}{N}.$$

Conditional information is given by,

$$H(X \mid Y) = H(X \cup Y) - H(Y).$$

Theorem 3.3. *Given a vertex variable V_i, and candidate parent set Π_i, let $V_j \notin \Pi_i$ such that $N \cdot \min\{H(V_i \mid \Pi_i), H(V_j \mid \Pi_i)\} \geq (1 - r_j) \cdot t(\Pi_i) + \epsilon$ for some $\epsilon \in \mathbb{R}^+$. Then the candidate parent set $\Pi_i' = \Pi_i \cup \{V_j\}$ and all its supersets can be safely pruned if σ is the BIC scoring function.*

3.5.2 Pruning with BDeu Score

A pruning rule for the BDeu scoring function appears in [29] and a more general version is included in [25]. Here, we present a relaxed version of the rule in [25].

Theorem 3.4. *Given a vertex variable V_i and candidate parent sets Π_i and Π_i' such that $\Pi_i \subset \Pi_i'$ and $\Pi_i \neq \Pi_i'$, let $r_i^+(\Pi_i')$ be the number of positive counts in the contingency table for Π_i'. If $\sigma(\Pi_i) + \epsilon < r_i^+(\Pi_i') \log r_i$, for some $\epsilon \in \mathbb{R}^+$ then Π_i' and the supersets of Π_i' can be safely pruned if σ is the BDeu scoring function..*

Data	n	N	T_3 (s)	$\lvert\mathcal{G}_3\rvert$	$\lvert\mathcal{M}_3\rvert$	T_{20} (s)	$\lvert\mathcal{G}_{20}\rvert$	$\lvert\mathcal{M}_{20}\rvert$	T_{150} (s)	$\lvert\mathcal{G}_{150}\rvert$	$\lvert\mathcal{M}_{150}\rvert$
tic tac toe	10	958	1.9	192	64	2.0	192	64	3.3	544	160
wine	14	178	4.1	308	51	24.9	3,449	576	143.7	26,197	4,497
adult	14	32,561	17.5	324	162	45.1	1,140	570	55.7	2,281	1,137
nltcs	16	3,236	53.8	240	120	201.7	1,200	600	1,005.1	4,606	2,303
msnbc	17	58,265	3,483.0	24	24	7,146.9	960	504	8,821.4	1,938	1,026
letter	17	20,000	OT	—	—	OT	—	—	OT	—	—
voting	17	435	1.3	27	2	4.0	441	33	14.3	2,222	170
zoo	17	101	8.1	49	13	21.9	1,111	270	299.3	21,683	5,392
hepatitis	20	155	7.1	580	105	513.3	87,169	15,358	1,452.8	150,000	49,269
parkinsons	23	195	30.7	1,088	336	3,165.9	150,000	39,720	4,534.3	150,000	116,206
sensors	25	5456	OT	—	—	OT	—	—	OT	—	—
autos	26	159	95.0	560	200	2,382.8	50,374	17,790	6,666.9	150,000	54,579
insurance	27	1,000	49.8	8,226	2,062	244.9	104,870	25,580	414.5	148,925	36,072
horse	28	300	18.8	1,643	246	1,358.8	150,000	28,186	1,962.5	150,000	69,309
flag	29	194	16.1	773	169	4,051.9	150,000	39,428	5,560.9	150,000	122,185
wdbc	31	569	396.1	398	107	10,144.2	28,424	8,182	45,938.2	150,000	54,846
mildew	35	1000	1.2	1,026	2	1.2	1,026	2	2.1	2,052	4
soybean	36	266	7,729.4	150,000	150,000	16,096.8	150,000	62,704	8,893.5	150,000	118,368
alarm	37	1000	6.3	1,508	122	684.2	123,352	9,323	2,258.4	150,000	8,484
bands	39	277	100.9	7,092	810	2,032.6	150,000	44,899	16,974.8	150,000	95,774
spectf	45	267	432.4	27,770	4,510	7,425.2	150,000	51,871	19,664.8	150,000	63,965
sponge	45	76	16.8	1,102	65	1,301.0	146,097	7,905	1,254.4	150,000	90,005
barley	48	1000	0.8	182	1	0.8	364	2	1.3	1,274	5
hailfinder	56	100	171.5	150,000	20	149.4	150,000	748	214.6	150,000	294
hailfinder	56	500	286.1	150,000	30,720	314.1	150,000	18,432	217.3	150,000	24,576
lung cancer	57	32	584.3	150,000	40,621	966.6	150,000	79,680	2,739.7	150,000	48,236

Table 3.1: The search time T, the number of collected networks $\lvert\mathcal{G}\rvert$ and the number of MECs $\lvert\mathcal{M}\rvert$ in the collected networks at BF $= 3$, 20 and 150 using BIC, where n is the number of random variables in the dataset, N is the number of instances in the dataset and OT = Out of Time.

3.6 Experimental Evaluation

In this section, we evaluate the proposed BF-based method and compare its performance with published k-best solvers.

Our proposed method is more memory efficient comparing to the k-best based solvers in BDeu scoring and often collects more networks in a shorter period of time. With the pruning rules generalized above, our method can scale up to datasets with 57 variables in BIC scoring, whereas the previous best results are reported on a network of 29 variables using the k-best approach with score pruning [16].

The datasets are obtained from the UCI Machine Learning Repository [30] and the Bayesian Network Repository[2]. Some of the complete local scoring files are downloaded from the `GOBNILP` website[3] and are used for the k-best related experiments only. Since not all solvers in the k-best experiments can take in scoring files, we exclude the time to compute local scores from the comparison. Both BIC [80, 56] and BDeu [8, 42] scoring functions are used where applicable. All experiments are conducted on computers with 2.2 GHz Intel E7-4850V3 processors. Each experiment is limited to 64 GB of memory and 24 hours of CPU time.

3.6.1 The Bayes Factor Approach

We modified the development version (Version denoted 9c9f3e6) of `GOBNILP`, referred to below as `GOBNILP`_dev, to apply pruning rules presented above during scoring and supplied appropriate parameter settings for collecting near-optimal networks[4]. The code is compiled with SCIP 6.0.0 and CPLEX 12.8.0. `GOBNILP` extends the SCIP Optimization Suite [38] by adding a *constraint handler* for handling the acyclicity constraint for DAGs. If multiple BNs are required `GOBNILP`_dev just calls SCIP to ask it to collect feasible solutions. In this mode, when SCIP finds a solution, the solution is stored, a constraint is added to render that solution infeasible and the search continues. This differs from (and is much more efficient than) `GOBNILP`'s current method for finding k-best BNs where an entirely new search is started each time a new BN is found. A recent version of SCIP has a separate "reoptimization" method which might allow better k-best performance for `GOBNILP` but we do not explore that here. By default when SCIP is asked to collect solutions it turns off all cutting plane algorithms. This led to very poor `GOBNILP` performance since `GOBNILP` relies

[2] http://www.bnlearn.com/bnrepository/

[3] https://www.cs.york.ac.uk/aig/sw/gobnilp/#benchmarks

[4] The modified code is available at: https://www.cs.york.ac.uk/aig/sw/gobnilp/

on cutting plane generation. Therefore, this default setting is overridden in `GOBNILP_dev` to allow cutting planes when collecting solutions. To find only solutions with objective no worse than $(OPT + \epsilon)$, SCIP's `SCIPsetObjlimit` function is used. Note that, for efficiency reasons, this is **not** effected by adding a linear constraint.

We first use `GOBNILP_dev` to find the optimal scores since `GOBNILP_dev` takes objective limit $(OPT + \epsilon)$ for enumerating feasible networks. Then all networks falling into the limit are collected with a counting limit of 150,000. Finally the collected networks are categorized into MEC, where two networks belong to the same MEC if they have the same skeleton and v-structures [99]. The proposed approach is tested on datasets with up to 57 variables. The search time T, the number of collected networks $|\mathcal{G}|$ and the number of MECs $\mathcal{M}$ in the collected networks at BF = 3, 20 and 150 using BIC are reported in Table 3.1, where n is the number of random variables in the dataset and N is the number of instances in the dataset. The three thresholds are chosen according to the interpreting scale suggested by [42] where 3 marks the difference between anecdotal and positive evidence, 20 marks positive and strong evidence and 150 marks strong and very strong evidence. The search time mostly depends on a combined effect of the size of the network, the sample size and the number of MECs at a given BF. Some fairly large networks such as alarm, sponge and barley are solved much faster than smaller networks with a large sample size, e.g., msnbc and letter.

The results also indicate that the number of collected networks and the number of MECs at three BF levels varies substantially across different datasets. In general, datasets with smaller sample sizes tend to have more networks collected at a given BF since near-optimal networks have similar posterior probabilities to the best network. Although the desired level of BF for a study, like the p-value, is often determined with domain knowledge, the proposed approach, given sufficient samples, will produce meaningful results that can be used for further analysis.

3.6.2 Bayes Factor vs. KBest

In this section, we compare our approach with published solvers that are able to find a subset of top-scoring networks with the given parameter k. The solvers under consideration are KBest_12b[5] from [95], KBestEC[6] from [17], and `GOBNILP` 1.6.3 [6], referred to as KBest, KBestEC and `GOBNILP` below. The first two solvers are based on the dynamic programming

[5] http://web.cs.iastate.edu/~jtian/Software/UAI-10/KBest.htm
[6] http://web.cs.iastate.edu/~jtian/Software/AAAI-14-yetian/KBestEC.htm

approach introduced in [86]. Due to the lack of support for BIC in KBest and KBestEC, only BDeu with a equivalent sample size of one is used in corresponding experiments.

The most recent stable version of `GOBNILP` is 1.6.3 that works with SCIP 3.2.1. The default configuration is used and experiments are conducted for both BIC and BDeu scoring functions. However, the k-best results are omitted here due to its poor performance. Despite that `GOBNILP` can iteratively find the k-best networks in descending order by adding linear constraints, the pruning rules designed to find the best network are turned off to preserve sub-optimal networks. In fact, the memory usage often exceeded 64 GB during the initial ILP formulation, indicating that the lack of pruning rules posed serious challenge for `GOBNILP`. `GOBNILP`_dev, on the other hand, can take advantage of the pruning rules presented above in the proposed BF approach and its results compare favorably to KBest and KBestEC.

The experimental results of KBest, KBestEC and `GOBNILP`_dev are reported in Table 3.2, where n is the number of random variables in the dataset, N is the number of instances in the dataset, and k is the number of top scoring networks. The search time T is reported for KBest, KBestEC and `GOBNILP_dev` (BF = 20). The number of DAGs covered by the k MECs $|\mathcal{G}_k|$ is reported for KBestEC. In comparison, the last two columns are the number of found networks $|\mathcal{G}_{20}|$ and the number of MECs $|\mathcal{M}_{20}|$ using the BF approach with a given BF of 20 and BDeu scoring function.

As the number of requested networks k increases, the search time for both KBest and KBestEC grows exponentially. The KBest and KBestEC are designed to solve problems of size fewer than 20[7], and so they have some difficulty with larger datasets. They also fail to generate correct scoring files for msnbc. KBestEC seems to successfully expand the coverage of DAGs with some overhead for checking equivalence classes. However, KBestEC took much longer than KBest for some instances, e.g., nltcs and letter, and the number of DAGs covered by the found MECs is inconsistent for nltcs, letter and zoo. The search time for the BF approach is improved over the k-best approach except for datasets with very large sample sizes. The generalized pruning rules are very effective in reducing the search space, which then allows `GOBNILP_dev` to solve the ILP problem subsequently. Comparing to the improved results in (Chen et al. [14], 2015; 2016), our approach can scale to larger networks if the scoring file can be generated.[8]

Now we show that different datasets have distinct score patterns in the top scoring networks. The scores of the 1,000-best networks for some datasets in the KBest experiment are plotted in Figure 3.1. A specific line for a dataset indicates the deviation ϵ from the

[7] Obtained through correspondence with the author.

[8] BDeu cannot be scored for larger instances without imposing a limit on the parent set size.

optimal BDeu score by the kth-best network. For reference, the red dash lines represent different levels of BFs calculated by $\epsilon = \log \text{BF}$ (see Equation 3.5). The figure shows that it is difficult to pick a value for k *a priori* to capture the appropriate set of top scoring networks. For a few datasets such as adult and letter, it only takes fewer than 50 networks to reach a BF of 20, whereas zoo needs more than 10,000 networks. The sample size has a significant effect on the number of networks at a given BF since the lack of data leads to many BNs with similar probabilities. It would be reasonable to choose a large value for k in model averaging when data is scarce and vice versa, but only the BF approach is able to automatically find the appropriate and credible set of networks for further analysis.

3.7 Summary

Existing approaches for model averaging for BNSL either severely restrict the structure of the Bayesian network or have only been shown to scale to networks with fewer than 30 random variables. In this chapter, we proposed a novel approach to model averaging inspired by performance guarantees in approximation algorithms that considers all networks within a factor of optimal. Our approach has two primary advantages. First, our approach only considers *credible* models in that they are optimal or near-optimal in score. Second, our approach is significantly more efficient and scales to much larger Bayesian networks than existing approaches. We modified `GOBNILP` [6], a state-of-the-art method for finding an optimal Bayesian network, to implement our generalized pruning rules and to find all *near*-optimal networks. Our experimental results demonstrate that the modified `GOBNILP` scales to significantly larger networks without resorting to restricting the structure of the Bayesian networks that are learned.

Data	n	N	T_k (s)	k	T_{EC} (s)	$\lvert\mathcal{G}_k\rvert$	T_{20} (s)	$\lvert\mathcal{G}_{20}\rvert$	$\lvert\mathcal{M}_{20}\rvert$
tic tac toe	10	958	0.2	10	0.5	67	0.6	152	24
			2.8	100	6.0	673			
			70.7	1,000	78.5	7,604			
wine	14	178	3.4	10	12.0	60	35.9	8,734	6,262
			85.0	100	168.4	448			
			3,420.4	1,000	3,064.4	4,142			
adult	14	32,561	3.3	10	633.5	68	9.3	792	19
			73.6	100	63,328.9	1,340			
			2,122.8	1,000	OT	—			
nltcs	16	3,236	11.8	10	47,338.4	552	125.5	652	326
			406.6	100	OT	—			
			13,224.6	1,000	OT	—			
msnbc	17	58,265	ES	—	ES	—	4,018.9	24	24
letter	17	20,000	26.0	10	18,788.0	200	56,344.8	20	10
			909.8	100	OT	—			
			41,503.9	1,000	OT	—			
voting	17	435	34.1	10	101.9	30	6.0	621	207
			1,125.7	100	1,829.2	3,392			
			38,516.2	1,000	42,415.3	3,665			
zoo	17	101	33.5	10	99.8	52	8,418.8	29,073	6,761
			1,041.7	100	1,843.4	100			
			41,412.1	1,000	OT	—			
hepatitis	20	155	351.2	10	872.3	89	441.4	28,024	3,534
			13,560.3	100	20,244.7	842			
			OT	1,000	OT	—			
parkinsons	23	195	3,908.2	10	OT	—	1,515.9	150,000	42,448
			OT	100	OT	—			
			OT	1,000	OT	—			
autos	26	159	OM	1	OM	—	OT	—	—
insurance	27	1,000	OM	1	OM	—	8.3	1,081	133

Table 3.2: The search time T and the number of collected networks k, $\lvert\mathcal{G}_k\rvert$ and $\lvert\mathcal{G}_{20}\rvert$ for KBest, KbestEC and `GOBNILP_dev` (BF = 20) using BDeu, where n is the number of random variables in the dataset, N is the number of instances in the dataset, OM = Out of Memory, OT = Out of Time and ES = Error in Scoring. Note that $\lvert\mathcal{G}_k\rvert$ is the number of DAGs covered by the k-best MECs in KBestEC and $\lvert\mathcal{M}_{20}\rvert$ is the number of MECs in the networks collected by `GOBNILP_dev`.

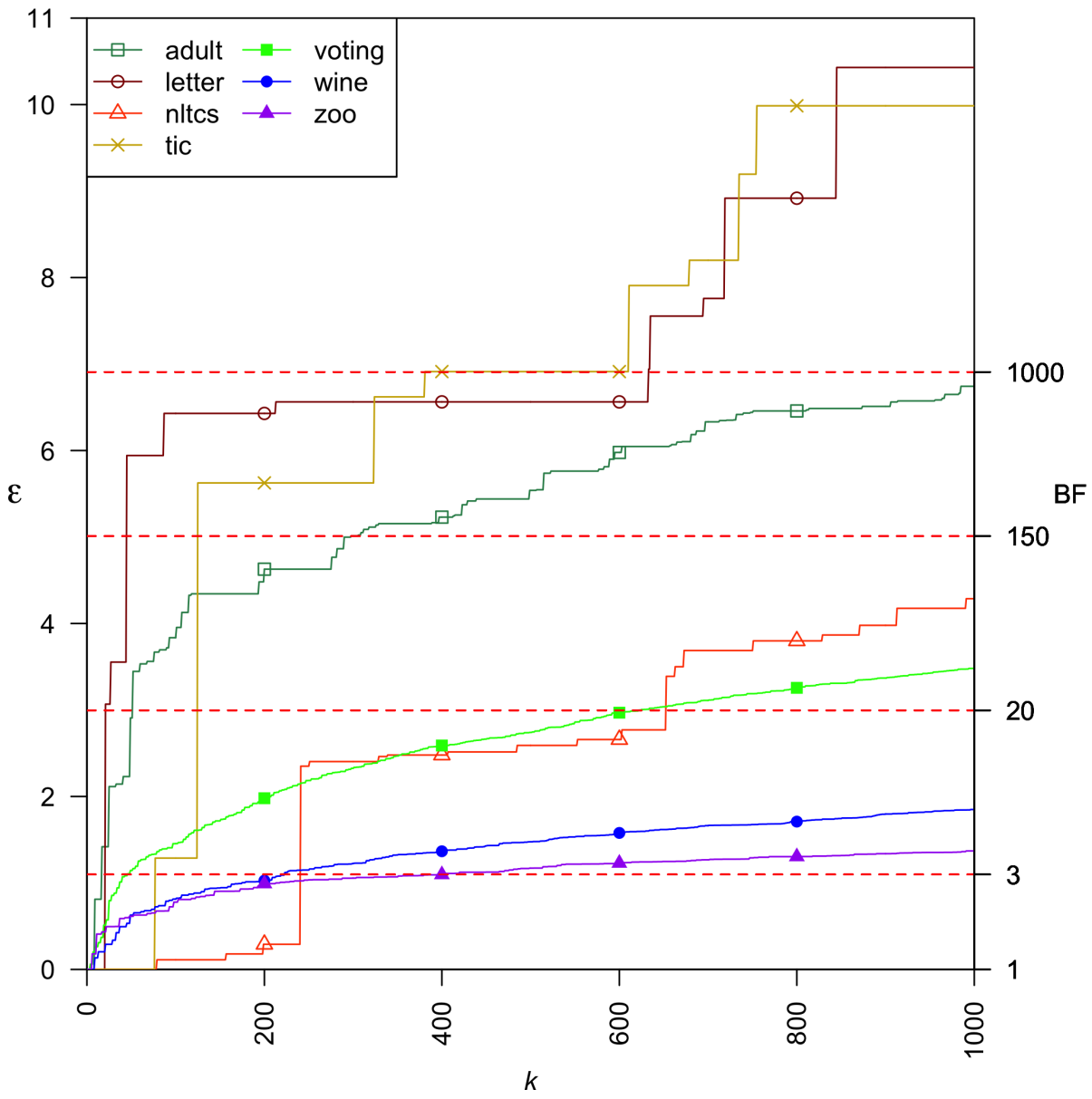


Figure 3.1: The deviation ϵ from the optimal BDeu score by k using results from KBest. The corresponding values of the BF ($\epsilon = \log(\text{BF})$, see Equation 3.5) are presented on the right. For example, if the desired BF value is 20, then all networks falling below the dash line at 20 are credible.

Chapter 4

Scoring Function Selection

Scoring functions for Bayesian network (BN) structure learning can conflict in their rankings and previous work has empirically studied their effectiveness with an aim to provide recommendations on their use. However, previous studies on scoring functions are limited by the small number and scale of the instances used in the evaluation and by a focus on learning a single network. Often, a better alternative to committing to a single network is to learn multiple networks and perform model averaging as this method provides confidence measures for knowledge discovery and improved accuracy for density estimation. In this chapter, we empirically study a selection of widely used and also recently proposed scoring functions. We address design limitations of previous empirical studies by scaling our experiments to larger BNs, comparing on an extensive set of both ground truth BNs and real-world datasets, considering alternative performance metrics, and comparing scoring functions on two model averaging frameworks: the bootstrap and the credible set. Contrary to previous recommendations based on finding a single structure, we find that for model averaging the BDeu scoring function is the preferred choice in most scenarios for the bootstrap framework and a recent score called qNML is the preferred choice for the credible set framework.

4.1 Introduction

As previously discussed, a BN can be learned from data using the well-known *score-and-search* approach, where a scoring function is used to evaluate the fit of a proposed BN to the data, and the space of directed acyclic graphs is searched for the best-scoring BN. Scoring functions commonly balance goodness of fit to the data with a penalty term for

model complexity to avoid overfitting. Common scoring functions for discrete data include Akaike information criterion (AIC) [2], BIC [80, 55, 76], and BDeu [8, 42]. More recently, the quotient Bayesian Dirichlet score based on Jeffreys' prior (qBDJ) [93] and qNML [89] scoring functions have been proposed.

There are three main aims for learning a Bayesian network [51, Ch. 16.2]: probability density estimation, classification, and knowledge discovery. BNs are still widely used for density estimation, especially in low dimension and data scarce regimes. Its knowledge discovery capability, e.g., representing causal effects and conditional independence relations, is still unmatched. Previous work has empirically studied the best scoring function to use for each of these aims. However, previous studies are limited by the small number and scale of the instances used in the evaluation, and by a focus on learning a single network as opposed to the widely used methodology of learning multiple networks and performing model averaging.

In early work, Van Allen and Greiner [97] compared AIC and BIC for density estimation. Their work learned a single network and only studied randomly generated instances up to 10 variables and two real-world networks: Alarm and Insurance. Carvalho [13] compared AIC, BDeu, and BIC for classification. However, the experimental evaluation was restricted to learning a single tree BN. Yang and Chang [100] compared BIC, BDe (a variant of BDeu), and several other scoring functions on density estimation and knowledge discovery. The evaluation focused on small instances with five or fewer variables and learned a single network.

More recently, Silander et al. [88] proposed a new scoring function called fNML and compared AIC, BDeu, BIC, and fNML on density estimation and knowledge discovery. Liu et al. [61] performed an extensive empirical comparison of AIC, BIC, BDeu, and fNML for knowledge discovery and concluded that BIC was overall the preferred choice. However, the evaluation used instances limited to at most 20 variables and did not consider model averaging.

Silander et al. [89] proposed a new scoring function called qNML and compared it against BDeu, BIC, and fNML on density estimation and knowledge discovery. The evaluation used small instances: 11 variables or fewer for evaluating knowledge discovery and 15 variables or fewer for evaluating density estimation. On the dimension of learning algorithms as opposed to scores, Scutari et al., [84] compared constraint-based, score-based, and hybrid learning algorithms. They found that the choice of statistical criteria (scores and their matching criteria) strongly affect the quality of the learned network, and that score-based algorithms and hybrid ones have similar performance with constraint-based ones slightly falling behind. They also used both BIC and BDeu as the scoring functions

and found no apparent difference. Broom et al. [7] is, to the best of our knowledge, the only empirical study of scoring functions that considers model averaging, as opposed to learning a single network. Their study uses the bootstrap framework but only performs experiments over two networks: Alarm and Insurance. By using such a limited testbed, they were not able to make any recommendations on which scoring function to prefer in general.

In this chapter, we fill the gap in previous empirical studies on scoring functions by scaling up the experiments to cover larger sized networks, by using a much more extensive testbed of instances, and by experimenting with two different model averaging frameworks: the bootstrap framework [34, 81] and the credible set framework [58]. We study five discrete scoring functions for ϵBNSL, namely AIC, BDeu, BIC, qBDJ, and qNML, and evaluate their performance on knowledge discovery and density estimation using both the ground truth BNs from bnlearn [81] and real-world datasets from the UCI repository. In addition to SHD for evaluating knowledge discovery, we also use the F_β-measure and the misclassification cost, which allows us to study tradeoffs between false positives and false negatives on discovering edges in the network. We use the negative log likelihood as an approximation to the KL divergence in density estimation. We find that the ideal score under the model averaging scheme is very different from previous recommendations resulting from learning a single structure. Based on our empirical evaluation, we conclude that BDeu is the clear preferred choice in most scenarios for the bootstrap framework. For the credible set model averaging framework, we conclude that qNML is the best choice for knowledge discovery, and that AIC is best suited for density estimation with qNML trailing slightly behind.

4.2 Scoring Functions

Scoring functions provide the model selection criteria in BNSL. Ideally a scoring function should have the following properties.

- **Consistency** [51, Def. 18.1]. The probability of the true graph $P(G^* \mid D) \to 1$ as $N \to \infty$. In terms of BNSL, the scoring function should choose the true graph G^* given a sufficiently large amount of data.
- **Decomposability** [51, Ch. 17.2.2]. The score of the entire network $\sigma(\mathcal{D} \mid G; \Theta)$ can be decomposed as the sum of local scores associated to each vertex $\sum_{i=1}^{n} \sigma(X_i \mid \Pi_i; \Theta)$.

- **Normality** [93] (score equivalence [18]). BNs of the same equivalence class have identical scores. Two BNs are equivalent if they impose identical conditional independence relations and can be structurally identified using the skeleton and v-structure [99].
- **Regularity** [92]. For two candidate parent sets $\Pi_{ij} \subset \Pi_{ik}$ of the child X_i, if both of them have identical empirical conditional entropy $H(X_i \mid \Pi_{i*})$, the smaller parent set Π_{ij} should have a better score.

Most scoring functions for BNSL are based on either log likelihood or Bayesian Dirichlet marginal likelihood. The log likelihood is the log probability of data $\mathcal{D}$ given a structure G and is often rearranged by vertices $\mathcal{X}$ with their parent sets Π,

$$LL(\mathcal{D} \mid G; \Theta) = \log \prod_{i=1}^{N} P(D_i \mid G; \Theta) = \sum_{i=1}^{n} \sum_{j=1}^{r_{\Pi_i}} \sum_{k=1}^{r_i} n_{ijk} \log \theta_{ijk},$$

where n_{ijk} is the count for $X_i = x_{ik}$ and $\Pi_i = \pi_{ij}$ in $\mathcal{D}$. It is well-known that using the log likelihood alone in BNSL yields the complete network since the likelihood never decreases when an edge is added. Various forms of penalties have been proposed to address the problem and several scoring functions have been derived that hold the desired properties above, including AIC [2], BIC [80], and qNML [89]. The scores can be defined generally as $\sigma(*) = LL(\mathcal{D} \mid G; \Theta) - \mathbf{pen}(*)$ for some penalty $\mathbf{pen}(*)$, and we present different penalties below. The penalty for the AIC scoring function in this work is defined as,

$$\mathbf{pen}(\text{AIC}) = \sum_{i=1}^{n} r_{\Pi_i}(r_i - 1).$$

AIC is traditionally used for supervised tasks as it minimizes mean squared error of predictions [9] and is asymptotically equivalent to leave-one-out cross validation [91]. A lower AIC score means a model is considered to be closer to the truth. The penalty for the BIC scoring function in this work is defined as,

$$\mathbf{pen}(\text{BIC}) = \sum_{i=1}^{n} r_{\Pi_i}(r_i - 1)\frac{\log N}{2}.$$

BIC estimates the posterior probability of a model being true. It penalizes models more heavily than AIC and requires a sample size much larger than the number of parameters in the model [61]. This definition of BIC is also equivalent to minimum descriptive length

(MDL) [76] scoring function under the assumption that $N \to \infty$ and $D_1, \ldots, D_N$ are i.i.d. The qNML score is derived from factorized normalized maximum likelihood (fNML) [88] that uses the vertex partitions in the normalizing factor. fNML is another log likelihood based score with the penalty defined as the regret, where a possible approximation is $\mathbf{reg}(N, r) \approx N\left(\log(\beta) + (\beta+2)\log(C_\beta) - \frac{1}{C_\beta}\right) - \frac{1}{2}\log\left(C_\beta + \frac{2}{\beta}\right)$, $\beta = \frac{r}{N}$, and $C_\beta = \frac{1}{2} + \frac{1}{2}\sqrt{1+\frac{4}{\beta}}$. However, fNML is not score equivalent in order to maintain decomposability. This drawback is recently addressed by the quotient version dubbed qNML. The penalty for the qNML scoring function in this work is defined as,

$$\mathbf{pen}(\text{qNML}) = \sum_{i=1}^{n} \mathbf{reg}(N, r_{\Pi_i} r_i) - \mathbf{reg}(N, r_{\Pi_i}).$$

Analytically qNML is similar to both AIC and BIC since they are all maximum likelihood based scores, though it has a more forgiving penalty. We show how the penalty differs across the log likelihood based scores in Table 4.1 and note that qNML has a more forgiving penalty than AIC and BIC.

Table 4.1: Penalties calculated from various values of sample size (N) and the number of possible instantiations for parent sets ($r_{\Pi i}$) when the child has 2 categories.

N	$r_{\Pi i}$	**pen**(AIC)	**pen**(BIC)	**pen**(qNML)	**pen**(qBDJ)
50	10	10	19.6	9.1	10.6
	100	100	195.6	24.1	26.3
	1,000	1,000	1,956.0	33.0	33.5
	10,000	10,000	19,560.1	34.5	34.5
500	10	10	31.1	18.8	21.5
	100	100	310.7	88.4	103.1
	1,000	1,000	3,107.3	239.4	261.8
	10,000	10,000	31,073.0	329.3	334.7
5,000	10	10	42.6	29.8	33.0
	100	100	425.9	185.2	212.0
	1,000	1,000	4,258.6	881.1	1028.2
	10,000	10,000	42,586.0	2,392.7	2616.4

From the Bayesian perspective, we can assume that the model parameters θ_{ijk} are independent Dirichlet variables with the priors $\mathrm{A} = \{\alpha_{ijk}\}$. Because the Dirichlet distribution is the conjugate prior distribution of the multinomial distribution, the posteriors

$\theta_{ijk} \mid D_{ijk}; \alpha_{ijk} \sim \mathrm{Dir}(\alpha_{ijk} + n_{ijk})$. It follows that,

$$\begin{aligned} LL(\mathcal{D} \mid G, \Theta; \mathrm{A}) &= \log P(\mathcal{D} \mid G, \Theta) P(\Theta; \mathrm{A}) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{r_{\Pi_i}} \frac{\sum_{k=1}^{r_i} \log \Gamma(\alpha_{ijk} + n_{ijk})}{\log \Gamma(\alpha_{ij*} + n_{ij*})} - \frac{\sum_{k=1}^{r_i} \log \Gamma(\alpha_{ijk})}{\log \Gamma(\alpha_{ij*})}, \end{aligned}$$

where $\alpha_{ij*} = \sum_{k=1}^{r_i} \alpha_{ijk}$ and $n_{ij*} = \sum_{k=1}^{r_i} n_{ijk}$. We consider two scores from the Bayesian Dirichlet (BD) family that have different priors. BDeu [8, 41] assigns $\alpha_{ijk} = \frac{\alpha}{r_i r_{\Pi_i}}$ for some equivalent sample size α, whereas Bayesian Dirichlet score based on Jeffreys' prior (BDJ) [92] assigns $\alpha_{ijk} = 0.5$. The BDeu scoring function has an associated hyperparameter α that must be properly set prior to scoring. Previous work has shown empirically (e.g., [61, 87]) the importance of choosing a suitable value for α. Unfortunately, there is little guidance available for setting α. Recently, Suzuki [92] proves that BDeu is not regular, often yielding unnecessarily complex structures. On the other hand, the BDJ scoring function is regular yet not normal [92]. Therefore, it is not desirable to use BDJ directly in BNSL. Switching the conditional scores $\sigma_{\mathrm{BDJ}}(X_i \mid \Pi_i, \Theta; \mathrm{A})$ in BDJ to the quotient version $\frac{\sigma_{\mathrm{BDJ}}(X_i, \Pi_i \mid \Theta; \mathrm{A})}{\sigma_{\mathrm{BDJ}}(\Pi_i \mid \Theta; \mathrm{A})}$ yields qBDJ [93] that is both regular and normal. The qBDJ scoring function in this work is defined as,

$$\sigma(\mathrm{qBDJ}) = \sum_{i=1}^{n} \sum_{j=1}^{r_{\Pi_i}} \log \frac{\sum_{k=1}^{r_i} \Gamma(n_{ijk} + 0.5)}{\Gamma(n_{ij*} + 0.5)} - \sum_{i=1}^{n} \log \frac{\Gamma(0.5 r_i r_{\Pi_i} + N) \Gamma(0.5 r_{\Pi_i})}{\Gamma(0.5 r_{\Pi_i} + N) \Gamma(0.5 r_i r_{\Pi_i})}.$$

By Stirling's approximation, $\sigma(\mathrm{qBDJ}) = LL(\mathcal{D} \mid G; \Theta) + O(1) - \mathbf{pen}(\mathrm{qBDJ})$, where $\mathbf{pen}(\mathrm{qBDJ})$ is exactly the second term in the definition.

Notably BDeu is the only irregular score in our study due to its broad application in BNSL. Other scores in the following experiments hold all four desirable properties.

4.3 Parameters

The parameters in log likelihood based scores are derived from maximum likelihood estimates, i.e., $\widehat{\theta_{ijk}} = \frac{n_{ijk}}{n_{ij*}}$. Although they are the closed form solutions to $\max_\Theta LL(\mathcal{D} \mid G; \Theta)$, it is often desirable to apply smoothing to model parameters, especially when some $n_{ijk} = 0$. In this work we use the m-estimate [67] defined as,

$$\widehat{\theta^m_{ijk}} = \frac{n_{ijk} + \frac{m}{r_i r_{\Pi_i}}}{n_{ij*} + \frac{m}{r_{\Pi_i}}}.$$

Recall that for the BD family, the posteriors $\theta_{ijk} \mid D_{ijk}; \alpha_{ijk} \sim \text{Dir}(\alpha_{ijk} + n_{ijk})$. Then the expected value of the posterior $\widehat{\theta^{\text{BD}}_{ijk}}$ is,

$$\widehat{\theta^{\text{BD}}_{ijk}} = \frac{n_{ijk} + \alpha_{ijk}}{n_{ij*} + \alpha_{ij*}}.$$

Coincidentally the m-estimate is the same as the expected value of the posterior parameters for BDeu when $m = \alpha$, where m is also called the equivalent sample size but stems from the idea of additive smoothing.

From the NML principle, we have yet another estimation called conditional NML predictive probability [77] (sequential NML [89]),

$$\widehat{\theta^{\text{sNML}}_{ijk}} = \frac{(n_{ijk} + 1)e(n_{ijk})}{\sum_{k=1}^{r_i}(n_{ijk} + 1)e(n_{ijk})},$$

where $e(n) = (1 + 1/n)^n$ and $e(0) = 1$. It has been shown [77] that sNML parameter converges to Krichevsky-Trofimov predictive probability, a special cases of the m-estimate when $m = r_{\Pi_i}$. Nevertheless, sNML provides an optimality guarantee in terms of regret [88], whereas the m-estimate has no known optimality property.

4.4 Model Averaging

We consider two model averaging frameworks for BNSL — the bootstrap and the credible set frameworks — as these two methods have been shown to scale the best among all available model averaging methods.

Bootstrapping is regarded as a general, flexible tool to provide confidence measures to statistics estimates. In the context of structure learning in Bayesian networks, Friedman et al. [34] proposed bootstrapping with thresholds to determine the existence of edges and other features. In particular, the non-parametric approach samples the original dataset with replacement and then heuristically learns a structure using the re-sampled data. After repeating such procedure many times, we can get the empirical probabilities of all edges by averaging on the learned structures. A threshold is finally applied to get the averaged structure.

In the credible set approach, all networks that are optimal or near-optimal in score are learned [58]. Note that the optimization problem defined by a scoring function and a

dataset is to find the maximum-score BN. Let OPT be the score of the optimal BN. The set of networks learned from a dataset, denoted the *credible set*, is given by,

$$\{G \mid score(G) \geq OPT - \log \mathrm{BF}\},$$

where the difference between the optimal score and the score of a network under consideration is proportional to the logarithm of the BF, a well-known criteria for selecting between two models. Each network in the credible set can then be aggregated to form a combined structure weighted by their score, where the scores of the networks in the credible set are normalized to sum to 1 and the best model has the highest weight. Alternatively, the networks can be equally weighted when averaged.

4.5 Pruning

Applying a scoring function to a dataset is a computationally intensive task, as many candidate parent sets need to be considered and scored. Fortunately, effective pruning rules have been developed for some scoring functions that preserve optimality but significantly reduce the candidate parents sets that need to be considered.

One of the most effective pruning rules for AIC and BIC is an upper-bound $\lceil \log_2(N) \rceil$ on the size of parent sets based on the sample size N. The rule is originally proposed in [28] for the optimal BNSL problem and generalized in [58] for the credible set approach. This rule enables AIC and BIC to scale much better than other scores under consideration. As we will show in our experiments, in scores other than AIC and BIC we often need to manually restrict the allowable maximum number of parents in order to score larger datasets within reasonable resource limits. Another effective family of pruning rules can eliminate certain parent sets and their supersets. Such rules for AIC, BIC and BDeu are originally proposed in [28] for the optimal BNSL problem and generalized to credible sets in [58].

4.6 Experimental Methodology

In this section, we describe the methodology we followed to experimentally study and compare scoring functions for BNSL in model averaging. We explain construction of the datasets (Section 4.6.1), scoring the datasets and learning the Bayesian network structures

(Section 4.6.2), and the performance evaluation metrics (Section 4.6.3). The scoring computations were conducted on SHARCNET[1] and the structure learning experiments were conducted on a shared server with 346 GB RAM and Intel Xeon Gold 6148 at 2.4 GHz. For scoring the datasets memory usage was limited to 64 GB and for structure learning a limit of 128 GB was imposed. For both scoring and learning, a computation time limit of 24 hours was imposed for each instance.

Table 4.2: UCI datasets (*left*, *middle*) and bnlearn Bayesian networks (*right*), where n is the number of variables in the dataset or network, and N is the number of instances in the original UCI dataset.

UCI dataset	n	N
shuttle	10	58,000
census income	14	48,842
letter	17	20,000
online shopping	18	12,330
lymphography	19	148
hepatitis	20	155
parkinsons	23	195
credit card	24	30,000

UCI dataset	n	N
robot navigation	25	5,456
horse colic	27	368
steel	28	1,941
flags	29	194
breast cancer	31	569
soybean	36	683
biodeg	42	1,055
spectf heart	45	267

network	n
sachs	11
child	20
insurance	27
water	32
mildew	35
alarm	37
barley	48
hailfinder	56
heparII	70
win95pts	76

4.6.1 Datasets

To empirically study the scoring functions, we considered a wide selection of datasets from the UCI repository[2] and networks from the bnlearn Bayesian network repository[3] (see Table 4.2). We preprocessed the UCI datasets using a k-nearest neighbor imputation algorithm, with $k = 5$, to fill in missing values and a supervised discretization method [31] based on the MDL principle to discretize continuous variables. For evaluating the scoring functions on the task of density estimation, each UCI dataset was then randomly partitioned to a training set and a test set by a 70% to 30% ratio.

[1] https://www.sharcnet.ca
[2] https://archive.ics.uci.edu/ml
[3] https://www.bnlearn.com/bnrepository/

For evaluating the scoring functions on the task of structure learning, we used a total of 90 ground truth BNs: 10 ground truth BNs came from the bnlearn repository and a further 80 ground truth BNs were constructed following a similar approach to Liu et al. [61] by (i) scoring each of the 16 UCI datasets using each of the five scoring functions AIC, BDeu, BIC, qBDJ, and qNML in turn, (ii) learning an optimal network structure from each scored dataset, and (iii) and fitting the parameters to each structure to give a final Bayesian network. Given the 90 ground truth BNs, we used the logic sampling function `rbn` from the bnlearn R package [81] to generate random samples of sizes $N = 50$, 100, 500, 1,000, 5,000, and 10,000 from the bif files. We collected three samples for each dataset size N, for a total of 18 samples for each ground truth BN. The number of variables n used in our experiments, ranging from 11 to 76, pushes the limits of both the bootstrap and the credible set model averaging approaches, especially when using scoring functions such as BDeu and qNML that have less effective pruning rules.

4.6.2 Scoring and Structure Learning

To evaluate the scoring functions within the bootstrap model averaging framework, we used the implementation available as the function `boot.strength` from the bnlearn R package [81]. We used the default replication factor of 200 and the tabu search algorithm, as in preliminary experiments it performed better than the alternative hill climbing algorithm. Due to score availability in bnlearn, we only consider AIC, BDeu, and BIC in the bootstrap experiments. A total of 4,320 bootstrap experiments were performed.

To evaluate the scoring functions within the credible set model averaging framework, we implemented the scoring functions AIC, BDeu, BIC, qBDJ, and qNML in `Python` to ensure a fair comparison[4]. The code takes a CSV file as input and generates a pruned score file iteratively for each parent set size. Saving the intermediate scoring files that guarantee optimality up to some parent set size is important since we do not limit the size a priori. As we stated above, the pruning rules for AIC and BIC are far more effective than those for other scores since an upper bound on the number of parents can be placed without losing optimality. For other scores, we have to abort the scoring generation at the end of the 24-hour limit. We note that similar experiments in [89, 61] have 20 variables as a computational limit for exact algorithms using scores other than AIC and BIC. Once the score files were generated, we used the `eBNSL` package [58], an extended version of `GOBNILP` [6], for collecting the credible networks. All networks falling within a Bayes factor (BF) of 150 were collected with a counting limit of 100,000. We also set the equivalent

[4] https://github.com/alisterl/hipss/releases/tag/v0.1.0

sample size $\alpha = 1$ for BDeu while the other scores do not have hyperparameters. We use a constant threshold 0.6 to determine whether an edge is present, and we have verified that the score ranking does not change with other reasonable thresholds. A total of 5,940 credible set experiments were performed.

4.6.3 Performance Evaluation Metrics

We evaluated the scoring functions based on their performance on knowledge discovery and density estimation. The former compares the learned structure with the ground truth BN in terms of directed and undirected edges and the latter compares the inference ability of the learned BNs. Each BN is weighted by its scores when the evaluation is conducted on a credible set using model averaging. For scoring functions based on posterior probabilities such as those in our experiments, the difference between two scores is proportional to the logarithm of the BF for the underlying models. The choice of the BF also has implications in model averaging since the worst model in the credible set will have a weight of $\frac{1}{\text{BF}}$ when the best model has a weight of 1.

Table 4.3: Comparison of scoring functions using structural Hamming distance for the bootstrap (*left*) and credible set (*right*) model averaging approaches. At each row, the aggregated Borda count is shown when comparing the scoring functions on a set of experiments that consist of three samples from each ground truth network and dataset sample sizes of $N = 50$, 100, 500, 1,000, 5,000, 10,000.

Ground truth	Scoring function		
	AIC	BDeu	BIC
bnlearn	**259**	145	85
UCI-AIC	225	**266**	96
UCI-BDeu	168	**335**	102
UCI-BIC	172	**248**	121
UCI-qBDJ	222	**226**	121
UCI-qNML	214	**246**	94
Total	1,260	**1,466**	619

Ground truth	Scoring function			
	AIC	BDeu	BIC	qNML
bnlearn	249	209	**270**	235
UCI-AIC	354	234	234	**378**
UCI-BDeu	306	330	249	**376**
UCI-BIC	258	240	300	**339**
UCI-qBDJ	370	188	270	**430**
UCI-qNML	344	184	267	**410**
Total	1,881	1,385	1,590	**2168**

4.7 Experimental Results and Discussion

In this section, we present the results of our experimental study and discuss their implications. The experimental results are aggregated using the Borda count. In the Borda count, in each trial (for a fixed dataset and model averaging method) the scoring functions are ranked according to the performance metric with ties allowed and each scoring function is awarded points corresponding to the number of scoring functions strictly lower in the ranking. Thus, the lowest ranked scoring function always gets 0 points and the highest ranked scoring function gets at most k points (exactly k if there are no ties for highest ranked), where k is the number of scoring functions under consideration. The Borda count was chosen to aggregate the results as it is known to select broadly acceptable options.

We present the results of knowledge discovery using the bootstrap approach in Table 4.3 (*left*) for SHD, in Figure 4.1 for directed multi-class F score, in Figure 4.2 for undirected F score, and in Figure 4.3 for undirected misclassification cost. The figures clearly show that BIC is a high precision low recall score since its Borda count is much higher as β or α decreases from 1. This is consistent with the fact that BIC imposes the most strict penalty on the number of parameters in the network.

In Table 4.3 (*left*), UCI-AIC, for example, refers to the ground truth set of networks that were constructed by using the AIC scoring function to find the optimal structure for each UCI dataset and then fitting the parameters to the structure to obtain a Bayesian network. We can conclude from the table that BDeu dominates both AIC and BIC except for the ground truth BNs from bnlearn. A finer-grained analysis of the SHD results for the bootstrap approach reveals that the values are dominated by missing edges in bnlearn experiments, and thus AIC, with its lower penalty on complexity, produced structures with fewer missing edges.

In Figure 4.1, we show the directed multi-class F score both broken down by sample sizes (a, b, and c) and aggregated across all sample sizes (d). Recall that the β value indicates the tradeoff between recall and precision. We can observe that BDeu dominates both AIC and BIC when the sample size is small (a), AIC tends to perform better given sufficient data (b), and AIC also overfits (with many false positives leading to a terrible precision) given too much data (c). Finally, the aggregated plot (d) shows that BDeu is the best score with varying values of β. We note that observations in Figure 4.2 and Figure 4.3 are similar to those of Figure 4.1.

We present the results of knowledge discovery using the credible set approach in Table 4.3 (*right*) for SHD, in Figure 4.4 for directed multi-class F score, in Figure 4.5 for undirected F score, and in Figure 4.6 for undirected misclassification cost.

The scoring function qBDJ is omitted from the presented results, as in extensive preliminary experiments it was dominated by qNML. In Table 4.3 (*right*), we observe that qNML is the best score on UCI datasets with ground truth generated using all 5 scores, whereas BIC is the best score on the ground truth BNs from bnlearn. A finer-grained analysis of the SHD results for the credible set approach reveals that the values here are dominated by extra edges, and thus BIC comes out ahead due to its heavy penalty on complexity.

In Figure 4.4, we show the directed multi-class F score both broken down by sample sizes (a, b, and c) and aggregated across all sample sizes (d). Recall that the β value indicates the tradeoff between recall and precision. We can observe that qNML dominates all other scores except for the largest sample sizes (c), where BDeu shows similar superior performance. Finally, the aggregated plot (d) shows that qNML is the best score with varying values of β, and that BDeu is better than AIC or BIC which has been shown in the bootstrap approach (Figure 4.1) as well. We note that observations in Figure 4.5 are similar to those of Figure 4.4.

In Figure 4.6, we show undirected misclassification cost both broken down by sample sizes (a, b, and c) and aggregated across all sample sizes (d). Recall that α value indicates the tradeoff between FP and FN. We can observe that the performance is mixed when the data is scarce (a). In particular, BIC tends to perform better when $\alpha < 1$ due to heavier penalty on FP, whereas qNML tends to perform better when $\alpha > 1$ due to heavier penalty on FN. When we bring in more data (b and c) or when we aggregate all sample sizes, we can observe that qNML is the clear winner among all scores.

In both the bootstrap and credible set approaches, our observations are different from those in Liu et al. [61] where the conclusion using only the optimal network leads to BIC being the dominant score. This difference can be attributed to (i) our evaluation is conducted with model averaging and (ii) we use a much more extensive set of datasets both in network sizes and in sample sizes in our experiments.

The results of density estimation are summarized in Table 4.4. Again, we use Borda count to aggregate the results on all datasets from the UCI repository. For the log likelihood based scores (AIC, BIC, and qNML), we learn their parameters using both the sNML method and the smoothed maximum likelihood method with $m = 1$, though the two methods have similar performance on the test data. The credible sets learned by BDeu and qBDJ are parameterized by their assumed Dirichlet distributions with $\alpha = 1$ and $\alpha_{ijk} = 0.5$. Note that the BDeu parameters are equivalent to the smoothed maximum likelihood ones since $m = \alpha = 1$. The negative log likelihood is calculated on a held-out test set from a 70%-30% train-test split ratio and the results indicate that AIC is the clear

winner in inference with qNML trailing slightly behind. AIC's advantage in inference is less apparent when we only consider large BNs or large datasets, but BIC remains the worst performer for inference in almost all cases. This observation suggests that BIC should not be used when density estimation is the intended usage of the learned BN.

Table 4.4: Borda score comparison on inference task using the set of credible networks learned from UCI datasets; e.g., the entry at column (AIC, snml) represents the Borda score for the combination of AIC as scoring function and snml as parameter estimation method.

AIC		BDeu	BIC		qBDJ	qNML	
m	snml	bdeu	m	snml	bdj	m	snml
85	73	49	29	22	66	68	56

The runtime of the structure learning task for each score is reflective of the pruning rules available to each score. In particular, AIC and BIC can complete the scoring task with almost all datasets while the other scoring functions require limits to be set on the maximum number of parents for $n \geq 20$ variables. The advantage of pruning rules for AIC and BIC, however, does not show up in the metrics used for both tasks in our study. When we put a limit on the size of the parent set, all scoring functions have similar runtime, suggesting that such a limit is the defining factor in efficiency. The scale of our experiments push the limits of model averaging approaches for BNSL. Although approximation methods that find a single high-quality network have been extended to thousands of variables [78], in contrast to model averaging approaches, such single-network methods cannot provide confidence measures for knowledge discovery and improved accuracy for density estimation.

4.8 Summary

Scoring functions can conflict in their rankings and previous work has empirically studied their effectiveness with an aim to provide recommendations on their use. However, previous studies on scoring functions are limited by the small number and scale of the instances used in the evaluation and by a focus on learning a single network. We have studied five discrete scoring functions for BNSL, namely AIC, BIC, qNML, BDeu, and qBDJ, scaled our experiments to large BNs using an extension to `GOBNILP`, and evaluated the scores with confidence measures on structure discovery and density estimation. We have addressed previous design limits by considering multiple metrics for structure discovery including the SHD, the F_β-measure, and the misclassification cost. The cost sensitive metrics present a

full picture with varying tradeoffs between precision vs. recall and FP vs. FN. We also evaluated scores on negative log likelihood in density estimation. We used both the ground truth BNs from bnlearn and real world UCI datasets in our structure learning tasks, and we are the first to provide an extensive experimental study of scoring functions in a model averaging framework.

Contrary to previous recommendations in [61], we find that qNML is the best contender for knowledge discovery using the exact credible set approach, and BDeu using bootstrapping, in most real world scenarios. We also find that AIC is best suited for density estimation with qNML trailing slightly behind. Our empirical study provides an insightful look at discrete score functions for BNSL and closes the gap in evaluating BN structures with confidence measures.

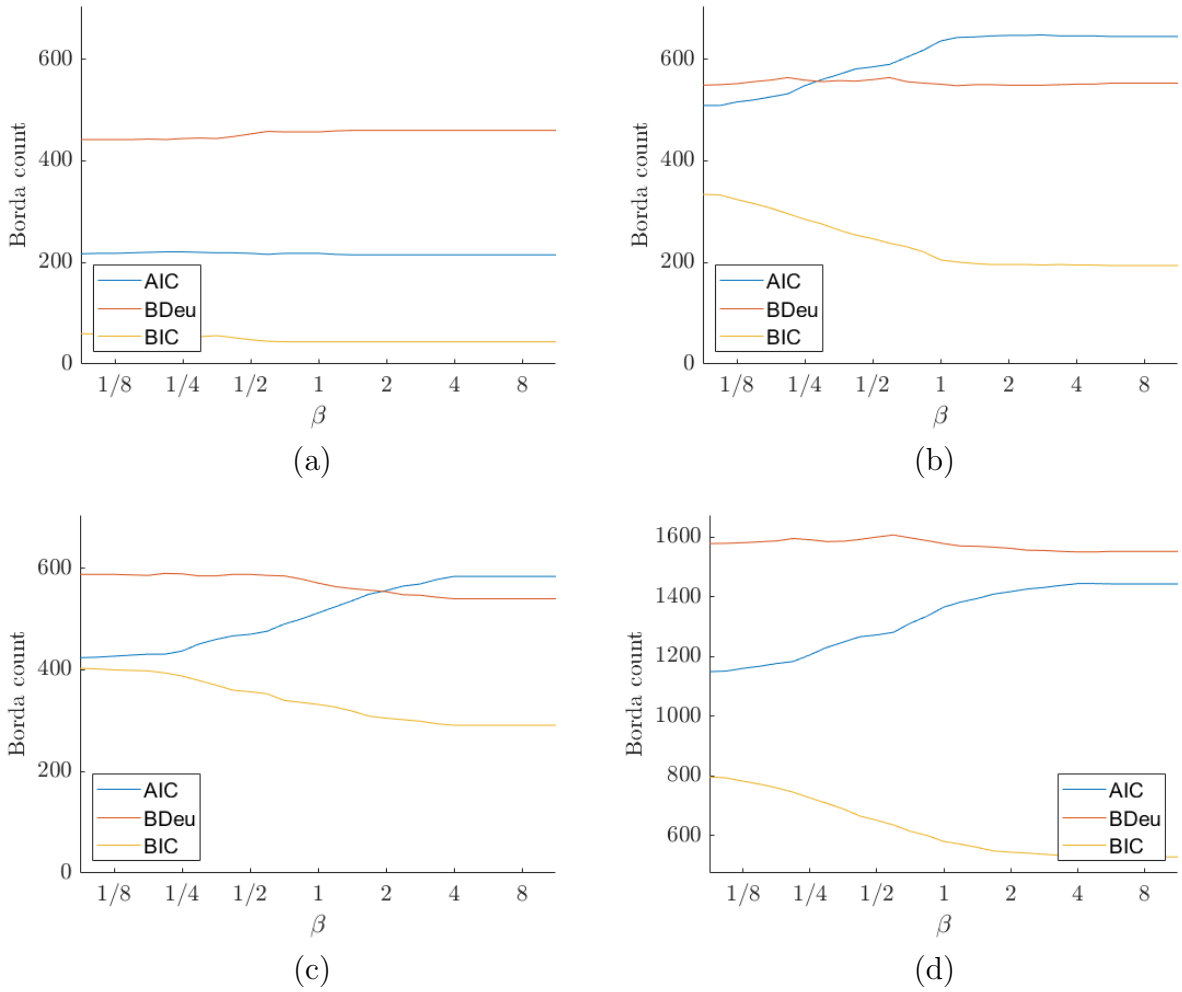


Figure 4.1: *Bootstrapping.* Comparison of scoring functions using weighted multi-class F_β score on *directed* edges. At each β, the aggregated Borda count is shown when comparing the scoring functions on a set of experiments that consist of three samples from each benchmark and dataset sample sizes of: (a) $N = 50, 100$; (b) $N = 500, 1000$; (c) $N = 5000, 10000$; (d) $N = 50, 100, 500, 1000, 5000, 10000$.

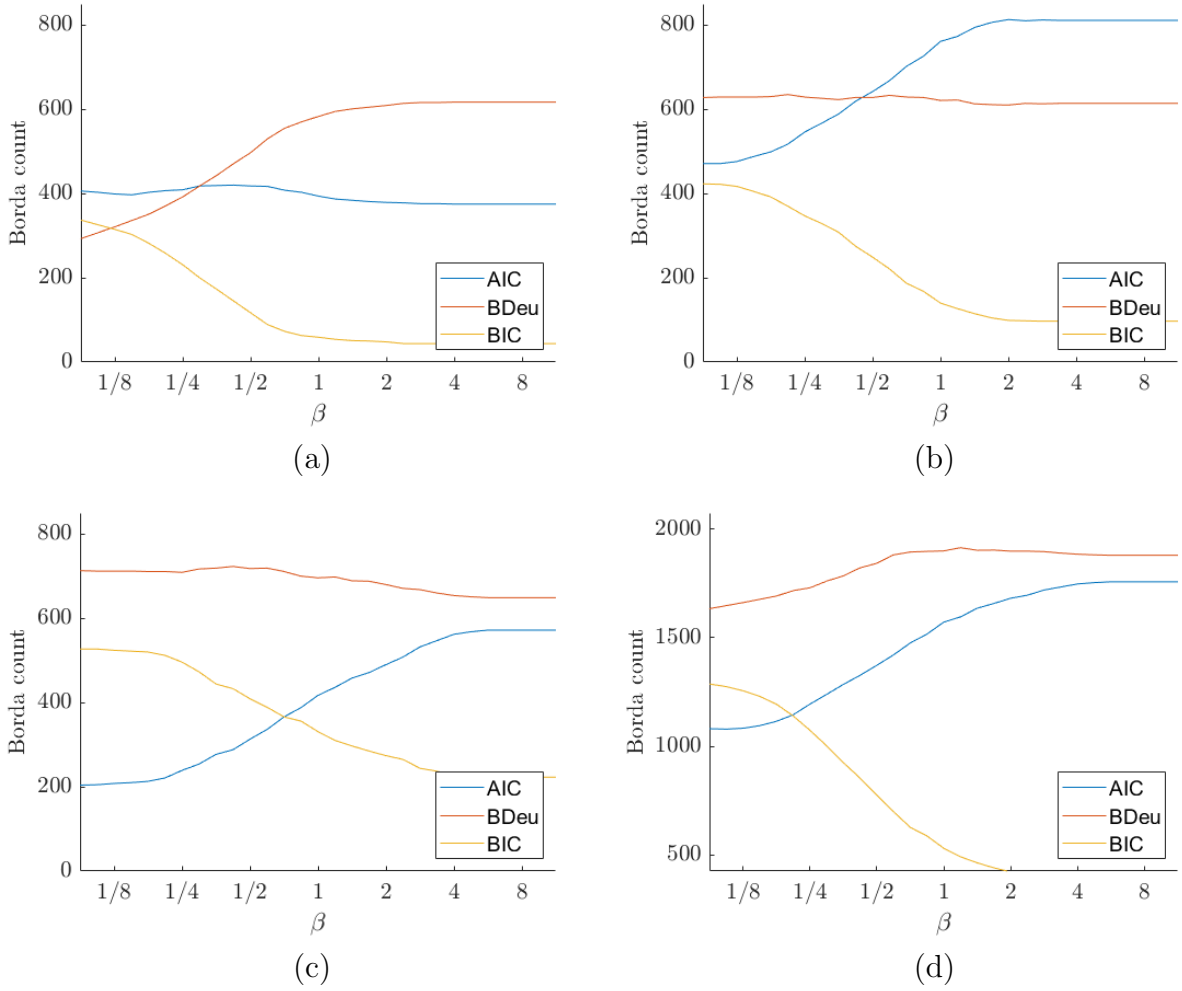


Figure 4.2: *Bootstrapping.* Comparison of scoring functions using F_β score on *undirected* edges. At each β, the aggregated Borda count is shown when comparing the scoring functions on a set of experiments that consist of three samples from each benchmark and dataset sample sizes of: (a) $N = 50, 100$; (b) $N = 500, 1000$; (c) $N = 5000, 10000$; (d) $N = 50, 100, 500, 1000, 5000, 10000$.

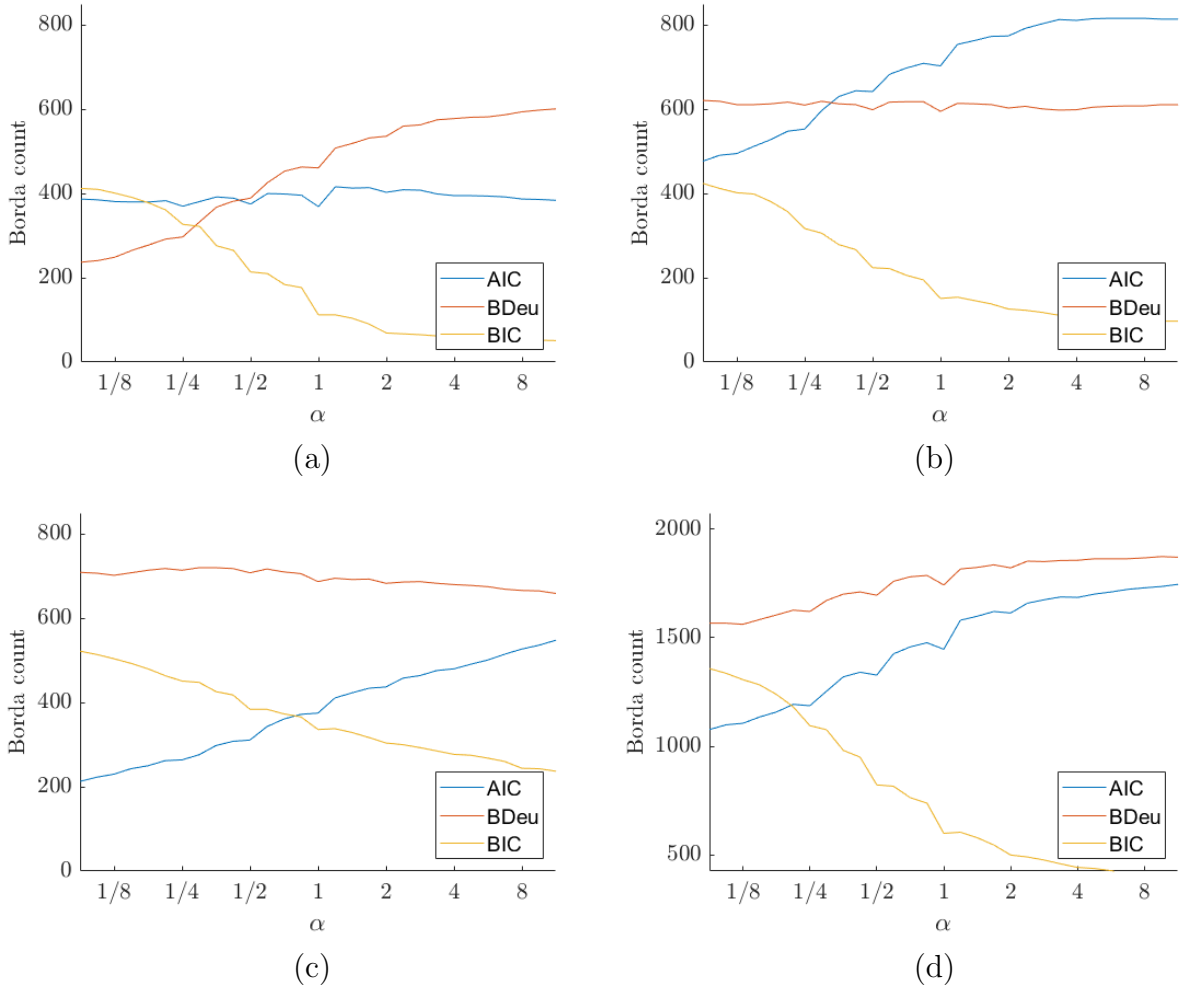


Figure 4.3: *Bootstrapping.* Comparison of scoring functions using misclassification cost on *undirected* edges. At each α, the aggregated Borda count is shown when comparing the scoring functions on a set of experiments that consist of three samples from each benchmark and dataset sample sizes of: (a) $N = 50, 100$; (b) $N = 500, 1000$; (c) $N = 5000, 10000$; (d) $N = 50, 100, 500, 1000, 5000, 10000$.

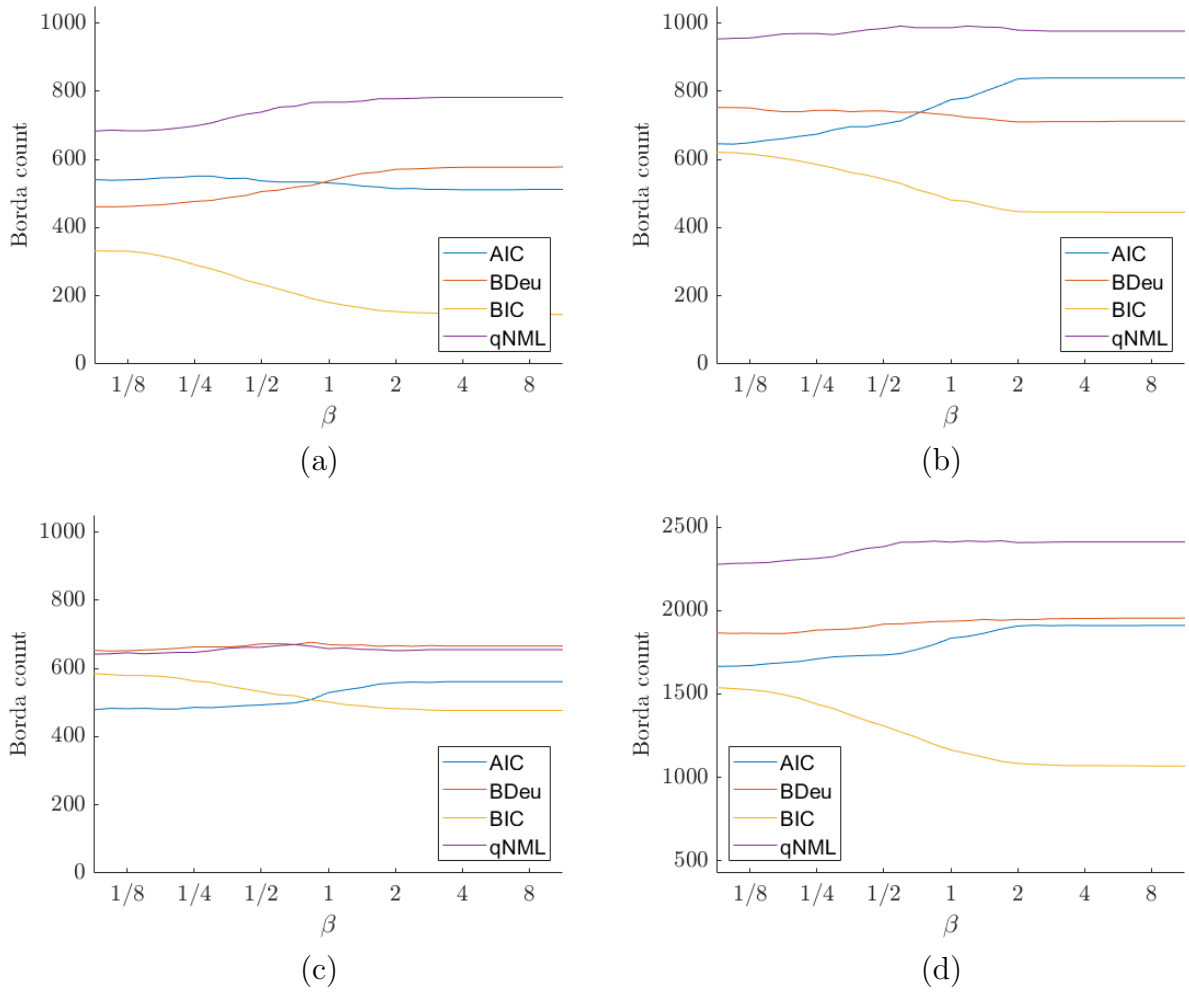


Figure 4.4: *Credible sets.* Comparison of scoring functions using weighted multi-class F_β score on *directed* edges. At each β, the aggregated Borda count is shown when comparing the scoring functions on a set of experiments that consist of three samples from each benchmark and dataset sample sizes of: (a) $N = 50, 100$; (b) $N = 500, 1000$; (c) $N = 5000, 10000$; (d) $N = 50, 100, 500, 1000, 5000, 10000$.

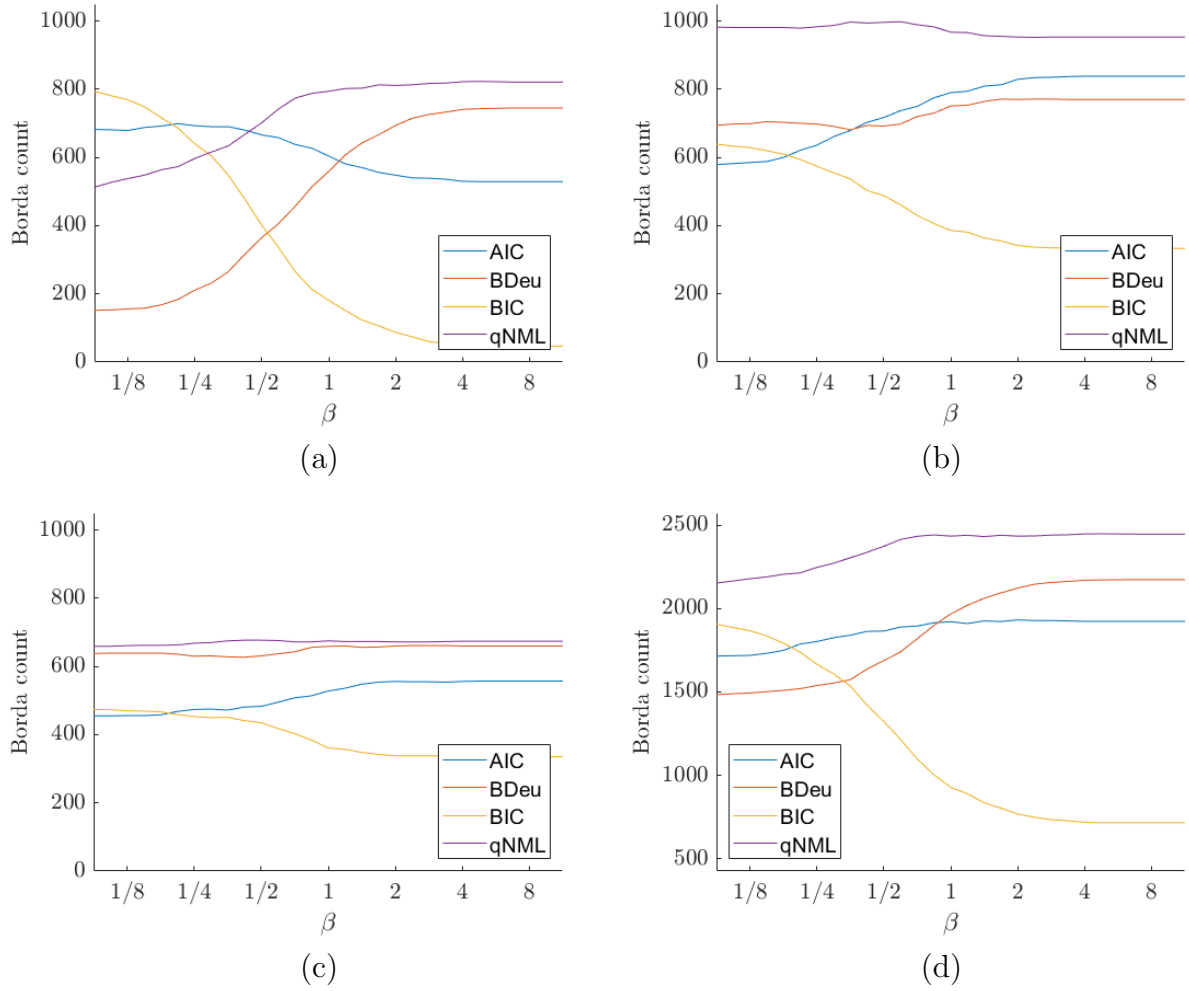


Figure 4.5: *Credible sets.* Comparison of scoring functions using F_β score on *undirected* edges. At each β, the aggregated Borda count is shown when comparing the scoring functions on a set of experiments that consist of three samples from each benchmark and dataset sample sizes of: (a) $N = 50, 100$; (b) $N = 500, 1000$; (c) $N = 5000, 10000$; (d) $N = 50, 100, 500, 1000, 5000, 10000$.

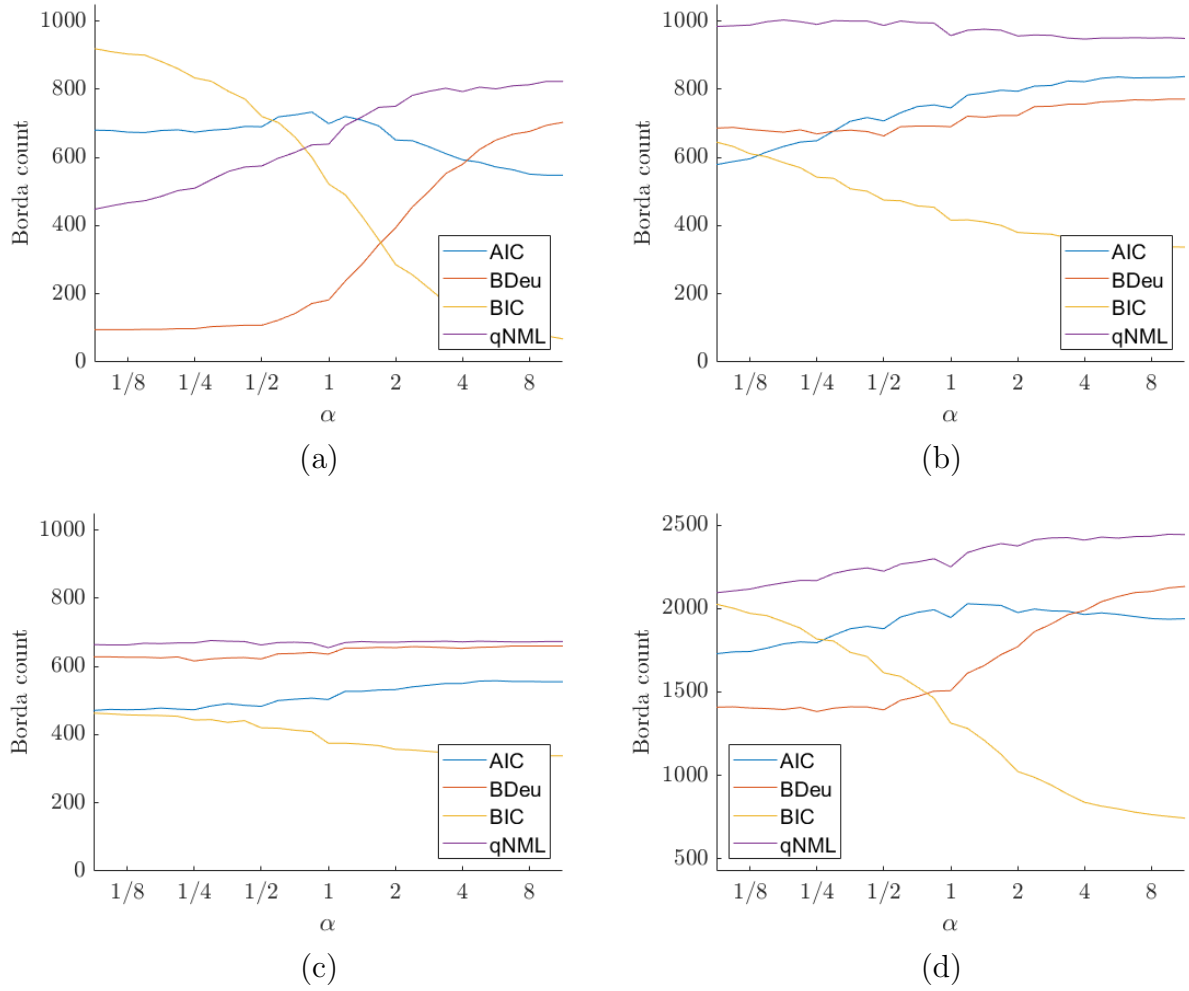


Figure 4.6: *Credible sets.* Comparison of scoring functions using misclassification cost on *undirected* edges. At each α, the aggregated Borda count is shown when comparing the scoring functions on a set of experiments that consist of three samples from each benchmark and dataset sample sizes of: (a) $N = 50, 100$; (b) $N = 500, 1000$; (c) $N = 5000, 10000$; (d) $N = 50, 100, 500, 1000, 5000, 10000$.

Chapter 5

Threshold Selection

Bayesian networks (BNs) are widely used as a data analysis tool in diverse areas, including finance, medicine, and sports. A standard data analysis methodology is to use the well-known score-and-search approach to learn a set of possible Bayesian networks and then to perform model averaging with thresholding to identify features such as edges between variables with high confidence. A fundamental step in the methodology is to select the threshold, as the value selected has broad implications for the success of the analysis. However, the problem of selecting a good threshold in BNSL has received limited attention in the literature. In this chapter, we identify an important shortcoming in a widely used threshold selection method. We then propose a simple transfer learning approach for maximizing target metrics and selecting a threshold that can be generalized from proxy datasets to the target dataset and show on an extensive set of benchmarks that it can perform significantly better than previous approaches.

5.1 Introduction

As already noted, a BN can be learned from data using the well-known *score-and-search* approach. However, selecting a single model (i.e., the best-scoring BN) is often not the best choice. When one is learning a BN from limited data, selecting a single model may be misleading as there may be many other BNs that have scores that are close to optimal, and the posterior probability of even the best-scoring BN is often close to zero. A more preferred alternative to committing to a single model is to perform some form of Bayesian or frequentist model averaging (e.g., [34, 58, 63, 95]).

Our interest here is in Bayesian networks as a knowledge discovery or data analysis tool. In the context of knowledge discovery, model averaging allows one to estimate, for example, the posterior probability or degree of confidence $\hat{P}(e \mid \mathcal{D})$ that an edge e is present in the true network structure describing the dependence structure in the dataset $\mathcal{D}$, rather than just knowing whether the edge is present in the best-scoring network. A widely used data analysis methodology is to: (i) learn a set of plausible networks that fit the data $\mathcal{D}$, (ii) perform model averaging to obtain $\hat{P}(e \mid \mathcal{D})$ for each edge e, and (iii) select a threshold c and report all edges e with $\hat{P}(e \mid \mathcal{D}) > c$. In this manner, a representative network can be constructed from the edges that are deemed significant that can then be examined for probabilistic dependencies and possible cause-effect relations.

Table 5.1: Representative data and causal analyses using Bayesian networks, where n is the number of random variables, N is the number of instances in the dataset, and c is the threshold for determine whether an edge is significant.

Area	n	N	c	Description
Banking	18	1,796	0.50	Contagion interactions between credit issuers following a sovereign default [4].
Biology	12	1,900	0.50†	Factors that directly impact red tide species occurrences and concentrations [32].
Medicine	11	79	0.65	Biophysical interactions of pneumonitis due to radiation therapy in lung cancer [62].
Medicine	17	120	0.30	Pathological interactions between diabetes mellitus and tuberculosis [73].
Medicine	26	408	0.85‡	Interactions between symptoms of obsessive-compulsive disorder and depression [64].
Safety	27	3,640	0.50†	Relationships between interstate motor carrier characteristics and safety performance [45].
Software	21	12,630	0.65	Interactions between code review measures and prevalence of post-release defects [54].
Sports	22	377	0.85	Relationships between psychological features and team performance in football [35].

† Unspecified and assumed to be the default value of the software package used in the analysis.
‡ Also used the threshold from Scutari and Nagarajan [82], which we will show to be equivalent to $c = 0.5$.

However, the problem of selecting a good threshold c for determining the significant

edges has received limited attention in the BNSL literature, in spite of the importance and widespread use of Bayesian networks for data analysis. Most data analyses use an ad hoc threshold, with little explicit discussion of the rationale for the value selected (e.g., see Table 5.1). Broom et al. [7] provide an algorithmic approach for identifying the significant edges in an instance-specific manner based on permutation testing. Scutari and Nagarajan [82] provide an optimization method for selecting a threshold in an instance-specific manner based on minimizing an L_1 distance. The method is the default threshold selection method in the widely used bnlearn package [81]. Gross et al. [39] analytically derive an instance-independent threshold for identifying the significant *directed* edges by modeling edge occurrences as a stochastic process. Little is known about the relative performance of these methods as previous empirical studies are limited by the small number and scale of the instances used in the evaluation (at most four networks were used in each study).

In this chapter, we make the following three contributions.

- We prove that the optimization method of Scutari and Nagarajan [82] is in fact *equivalent* to a fixed threshold of $c = 0.5$ for all instances; i.e., although their method selects different thresholds depending on the instance under consideration, the significant edges identified upon thresholding are exactly the same as if a cutoff of $c = 0.5$ was used.

- We propose a simple machine learning approach based on transfer learning for selecting a threshold. The thresholds learned apply to all instances and can be readily applied in practice. Existing methods for automatically selecting a threshold implicitly or explicitly assume that the cost of a false positive (an extra edge) and of a false negative (a missing edge) are equal. However, the relative costs are application dependent and in many domains (e.g., see Table 5.1) this assumption does not hold. We address the imbalance in costs by considering as loss functions in our machine learning framework (i) the cost-sensitive F_β measure with varying tradeoffs between precision and recall, (ii) the misclassification cost with varying tradeoffs between false positives and false negatives, as well as (iii) the cost-insensitive structural Hamming distance.

- We perform an extensive empirical evaluation of our proposed method against previously proposed methods. On a broad set of ground truth Bayesian networks and real-world datasets, our approach gave significantly better performance in almost all scenarios and competitive performance on the others.

5.2 Collect a Set of Networks

In BNSL using model averaging, the goal of the structure learning is to estimate the posterior probability or degree of confidence $\hat{P}(e_j \mid \mathcal{D})$ that an edge e_j is present in the true network structure. Among the model averaging methods available to collect a set of BNs to estimate the empirical probabilities, two have been shown to scale the best: the bootstrap method [34] and the credible set method [58]. Both methods use the well-known *score-and-search* approach, where a scoring function such as BIC [80, 55], BDeu [8, 42], or qNML [89] is used to evaluate the fit of a proposed BN to the data, and the space of directed acyclic graphs is searched for the best-scoring BN.

The bootstrap method [34] collects M samples $\mathcal{D}_m$ of the same length as the original dataset $\mathcal{D}$ via random sampling with replacement, for some given M. For each sample $\mathcal{D}_m$, it uses search to find a locally optimal BN $G_m = (\mathcal{X}, E_m)$. The empirical probability of some edge $P(e_j)$ is then approximated by simply counting its occurrence in the M BNs, i.e., $\hat{P}(e_j) = \frac{1}{M}\sum_{m=1}^{M} e_{jm}$, where $e_{jm} = 1$ if the edge e_j is present in E_m. Due to its simplicity, this method has been a popular choice in the application literature (see Table 5.1), and most use the bnlearn package [81] to estimate the empirical probability of the edges.

The credible set method [58] collects all networks that are optimal or near-optimal in score using exact search. Note that the optimization problem defined by a scoring function and a dataset is to find the maximum-score BN. Let OPT be the score of the optimal BN. The set of networks learned from a dataset, denoted the *credible set*, is given by,

$$\mathcal{G} = \{G \mid score(G) \geq OPT - \log \mathrm{BF}\},$$

where the difference between the optimal score and the score of a network under consideration is proportional to the logarithm of some given BF, a well-known criteria for selecting between two models. Each network in the credible set can then be aggregated to form a combined structure weighted by its score, where the scores of the networks in the credible set are normalized to sum to 1 and the best model has the highest weight. Alternatively, as with the bootstrap method, the networks can be equally weighted when averaged, yielding $\hat{P}(e_j) = \frac{1}{M}\sum_{m=1}^{M} e_{jm}$, where $M = |\mathcal{G}|$ is the number of optimal and near-optimal networks learned from the dataset and $e_{jm} = 1$ if the edge e_j is present in E_m.

5.3 Identify Edges: Distance Measure Approach

In this section, we perform a theoretical evaluation of the instance-specific optimization method of Scutari and Nagarajan [82]. The method is the default threshold selection

method in the widely used bnlearn package [81]. In particular, we prove that their optimization method is in fact equivalent to a fixed threshold of $c = 0.5$ for all instances.

5.3.1 Setup

We begin by presenting Scutari and Nagarajan's method (Algorithm 5.1 [82, Section 2]). Recall that the empirical probability of an edge over a set of M BNs is defined as $\hat{P}(e_j) = \frac{1}{M}\sum_{m=1}^{M} e_{jm}$, where $e_{jm} = 1$ if the edge e_j is present in E_m. Denote the vector of all such empirical probabilities as $\hat{\mathbf{p}} = \{\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_k\}$, where $k = \binom{|V|}{2}$, and consider the order statistic $\hat{\mathbf{p}}_{(\cdot)}$ (line 8 of Algorithm 5.1) derived from $\hat{\mathbf{p}}$:

$$\hat{\mathbf{p}}_{(\cdot)} = \left(\hat{p}_{(1)}, \cdots, \hat{p}_{(k)}\right), \quad \hat{p}_{(1)} \leq \cdots \leq \hat{p}_{(k)}.$$

Intuitively, the first elements of $\hat{\mathbf{p}}_{(\cdot)}$ are more likely to be associated with non-significant edges while the last elements are more likely to be associated with significant edges.

Let $G_0 = (V, E_0)$ be the (unknown) true network structure. The ideal configuration $\tilde{\mathbf{p}}_{(\cdot)}$ of $\hat{\mathbf{p}}_{(\cdot)}$ is of the form $\{0, \ldots, 0, 1, \ldots, 1\}$ where

$$\tilde{p}_{(i)} = \begin{cases} 1 & e_{(i)} \in E_0 \\ 0 & e_{(i)} \notin E_0 \end{cases}.$$

This configuration characterizes every edge as either significant or non-significant without any uncertainty. The empirical CDFs of $\hat{\mathbf{p}}_{(\cdot)}$ and $\tilde{\mathbf{p}}_{(\cdot)}$ are given by[1]

$$F_{\hat{\mathbf{p}}_{(\cdot)}}(x) = \frac{1}{k}\sum_{i=1}^{k} \mathbb{1}_{\{\hat{p}_{(i)} \leq x\}}, \quad x \in [0, 1];$$

$$F_{\tilde{\mathbf{p}}_{(\cdot)}}(x) = \begin{cases} 0 & x \in (-\infty, 0) \\ t & x \in [0, 1) \\ 1 & x \in [1, \infty) \end{cases}.$$

Note the scalar $t \in [0, 1]$ corresponds to the fraction of elements of $\tilde{\mathbf{p}}_{(\cdot)}$ equal to zero and is a measure of the fraction of non-significant edges in the true structure G_0. This provides a threshold for separating the elements of $\tilde{\mathbf{p}}_{(\cdot)}$, namely:

$$e_{(i)} \in E_0 \iff \tilde{p}_{(i)} > F^{-1}_{\tilde{\mathbf{p}}_{(\cdot)}}(t), \tag{5.1}$$

where $F^{-1}_{\tilde{\mathbf{p}}_{(\cdot)}}(t) = \inf_{x \in \mathbb{R}}\{F_{\tilde{\mathbf{p}}_{(\cdot)}}(x) \geq t\}$ is the quantile function.

[1] The indicator function in $F_{\hat{\mathbf{p}}_{(\cdot)}}$ should be $\mathbb{1}_{\{p_i \leq x\}}$ instead of $\mathbb{1}_{\{p_i < x\}}$. This was a typo in the original paper.

5.3.2 Estimating Threshold

The threshold t can be estimated by approximating the ideal, asymptotic empirical CDF $F_{\tilde{\mathbf{p}}_{(\cdot)}}$ with its finite sample estimate $F_{\hat{\mathbf{p}}_{(\cdot)}}$. Scutari and Nagarajan [82] proposed to use the L_1 norm given by,

$$L_1(t; \hat{\mathbf{p}}_{(\cdot)}) = \int \left| F_{\hat{\mathbf{p}}_{(\cdot)}}(x) - F_{\tilde{\mathbf{p}}_{(\cdot)}}(x; t) \right| \, dx.$$

Since $F_{\hat{\mathbf{p}}_{(\cdot)}}$ is piecewise constant, changing value only at $\hat{p}_{(i)}$'s, we can write

$$L_1(t; \hat{\mathbf{p}}_{(\cdot)}) = \sum_{x_i \in \{\{0\} \cup \hat{\mathbf{p}}_{(\cdot)} \cup \{1\}\}} |F_{\hat{\mathbf{p}}_{(\cdot)}}(x_i) - t|(x_{i+1} - x_i).$$

Scutari and Nagarajan [82] propose to use linear programming to find the $\hat{t} \in [0, 1]$ that minimizes $L_1(t; \hat{\mathbf{p}}_{(\cdot)})$ (line 9 of Algorithm 5.1). The full procedure of Scutari and Nagarajan [82] is summarized in Algorithm 5.1. The threshold $\hat{t}$ can then be used as in rule (5.1) to identify significant edges. We now present our main result (Theorem 5.1) on the deterministic value of $\hat{t}$ below.

Algorithm 5.1: The threshold selection method from Scutari and Nagarajan [82].

Input: $e_j \in \{0, 1\}$ indicating whether edge e_j is present in the ground truth network.

Input: $\hat{e}_{jm} \in \{0, 1\}$ indicating whether edge e_j is present in the collected m-th network.

Output: The threshold $\hat{t}$.

1 **for** $j \in \{1, 2, \cdots, n(n-1)/2\}$ **do**
2 $\quad O_{e_j} = 0$; `/* Initialize the occurrences of edge` e_j. `*/`
3 $\quad$ **for** $m \in \{1, 2, \cdots, M\}$ **do**
4 $\quad\quad O_{e_j} = O_{e_j} + \hat{e}_{jm}$;
5 $\quad$ **end**
6 $\quad \hat{P}(e_j) = \frac{O_{e_j}}{M}$;
7 **end**
8 Sort $\hat{P}(e_j)$ s.t. $\hat{\mathbf{p}}_{(\cdot)} = \left(\hat{p}_{(1)}, \cdots, \hat{p}_{(j)}\right), \quad \hat{p}_{(1)} \leq \cdots \leq \hat{p}_{(j)}$;
9 Find $\hat{t} \in [0, 1]$ that minimizes $L_1(t; \hat{\mathbf{p}}_{(\cdot)})$.

Theorem 5.1. *$L_1(t; \hat{\mathbf{p}}_{(\cdot)})$ attains its global minimum at $\hat{t} = F_{\hat{\mathbf{p}}_{(\cdot)}}(x = 0.5)$.*

Proof. Let $Q := \{0\} \cup \hat{\mathbf{p}}_{(\cdot)} \cup \{1\}$ and $\Delta x_i := x_{i+1} - x_i$ for $i \in \{0, 1, \ldots, k\}$. By the definition of $L_1(t; \hat{\mathbf{p}}_{(\cdot)})$,

$$\begin{aligned} L_1(F_{\hat{\mathbf{p}}_{(\cdot)}}(0.5); \hat{\mathbf{p}}_{(\cdot)}) &= \sum_{x_i \in Q} \mid F_{\hat{\mathbf{p}}_{(\cdot)}}(x_i) - F_{\hat{\mathbf{p}}_{(\cdot)}}(0.5) \mid \Delta x_i \\ &= \sum_{x_i \in \{q_i \mid q_i < 0.5\}} (-F_{\hat{\mathbf{p}}_{(\cdot)}}(x_i) + F_{\hat{\mathbf{p}}_{(\cdot)}}(0.5)) \Delta x_i \\ &\quad + \sum_{x_i \in \{q_i \mid q_i > 0.5\}} (F_{\hat{\mathbf{p}}_{(\cdot)}}(x_i) - F_{\hat{\mathbf{p}}_{(\cdot)}}(0.5)) \Delta x_i. \end{aligned}$$

For any $\delta \in \mathbb{R}^+$, observe that

$$\begin{aligned} & L_1(F_{\hat{\mathbf{p}}_{(\cdot)}}(0.5+\delta); \hat{\mathbf{p}}_{(\cdot)}) - L_1(F_{\hat{\mathbf{p}}_{(\cdot)}}(0.5); \hat{\mathbf{p}}_{(\cdot)}) \\ = \; & (F_{\hat{\mathbf{p}}_{(\cdot)}}(0.5+\delta) - F_{\hat{\mathbf{p}}_{(\cdot)}}(0.5)) \\ & \cdot \left(\sum_{x_i \in \{q_i \mid q_i < 0.5+\delta\}} \Delta x_i - \sum_{x_i \in \{q_i \mid q_i > 0.5+\delta\}} \Delta x_i \right) \\ = \; & (F_{\hat{\mathbf{p}}_{(\cdot)}}(0.5+\delta) - F_{\hat{\mathbf{p}}_{(\cdot)}}(0.5)) \\ & \cdot \left(\sup_Q \{q_i \mid q_i < 0.5+\delta\} - (1 - \inf_Q \{q_i \mid q_i > 0.5+\delta\}) \right). \end{aligned}$$

If $F_{\hat{\mathbf{p}}_{(\cdot)}}(0.5+\delta) = F_{\hat{\mathbf{p}}_{(\cdot)}}(0.5)$, then the difference is zero. If $F_{\hat{\mathbf{p}}_{(\cdot)}}(0.5+\delta) > F_{\hat{\mathbf{p}}_{(\cdot)}}(0.5)$, i.e., $\{q_i \mid 0.5 < q_i \leq 0.5+\delta\} \neq \emptyset$, it follows that $\sup_Q\{q_i \mid q_i < 0.5+\delta\} > 0.5$ and $1 - \inf_Q\{q_i \mid q_i > 0.5+\delta\} < 0.5$, and the difference is positive. Therefore, $L_1(F_{\hat{\mathbf{p}}_{(\cdot)}}(0.5+\delta); \hat{\mathbf{p}}_{(\cdot)}) \geq L_1(F_{\hat{\mathbf{p}}_{(\cdot)}}(0.5); \hat{\mathbf{p}}_{(\cdot)})$. A mirror argument proves that $L_1(F_{\hat{\mathbf{p}}_{(\cdot)}}(0.5-\delta); \hat{\mathbf{p}}_{(\cdot)}) \geq L_1(F_{\hat{\mathbf{p}}_{(\cdot)}}(0.5); \hat{\mathbf{p}}_{(\cdot)})$. Combining the two inequalities, we conclude that $L_1(F_{\hat{\mathbf{p}}_{(\cdot)}}(0.5); \hat{\mathbf{p}}_{(\cdot)})$ is the global minimum. □

Following Theorem 5.1, the optimal cutoff value for empirical edge probabilities in terms of the L_1 norm is 0.5, formally $e_{(i)} \in E_0 \iff \hat{p}_{(i)} > 0.5$. This suggests that using linear programming to find the $\hat{t} \in [0, 1]$ that minimizes $L_1(t; \hat{\mathbf{p}}_{(\cdot)})$ would yield results equivalent to $\hat{t} = F_{\hat{\mathbf{p}}_{(\cdot)}}(x = 0.5)$. See Example 5.1 for a demonstration.

Example 5.1 (Threshold selection from Scutari and Nagarajan [82])**.** *Suppose we have a probability vector* $\hat{\mathbf{p}}_{(\cdot)} = \{0.1, 0.2, 0.4, 0.8\}$*. Following Scutari and Nagarajan [82], we want*

to optimize the fraction of insignificant edges t s.t. $L_1(t; \hat{\mathbf{p}}_{(\cdot)})$ is minimized, i.e.,

$$\underset{t}{\operatorname{argmin}}\, L_1(t; \hat{\mathbf{p}}_{(\cdot)}) = \sum_{x_i \in \{0,0.1,0.2,0.4,0.8,1\}} |F_{\hat{\mathbf{p}}_{(\cdot)}}(x_i) - t|(x_{i+1} - x_i)$$

We can calculate the values for $L_1(t; \hat{\mathbf{p}}_{(\cdot)})$ at each cutoff value c that alters t (See Table 5.2). As we have proven in Theorem 5.1, the global minimum of $L_1(t; \hat{\mathbf{p}}_{(\cdot)})$ is attained at $\hat{t} = F_{\hat{\mathbf{p}}_{(\cdot)}}(x = 0.5) = 0.75$, which corresponds to any $c \in [0.4, 0.8)$. This is why the optimization method would yield a cutoff value that is equivalent to the constant $c_S = 0.5$.

Table 5.2: Comparison of the cutoff value c and its corresponding fraction of insignificant edges t and the L_1 norm. Note that the L_1 norm is minimized when $t = 0.75$, which corresponds to $c \in [0.4, 0.8)$.

c	$t = F_{\hat{\mathbf{p}}_{(\cdot)}}(x = c)$	$L_1(t; \hat{\mathbf{p}}_{(\cdot)})$
0	0	0.625
0.1	0.25	0.545
0.2	0.5	0.425
0.4	0.75	0.225
0.5	0.75	0.225
0.8	1	0.315

5.4 Identify Edges: Our Learning Approach

In this section, we present our simple transfer learning approach for selecting a threshold that directly optimizes a desired metric.

The result of learning a set of plausible networks from a dataset and performing model averaging is an estimate $\hat{P}(e_j)$, for each possible edge e_j, representing the probability or the confidence that the edge e_j is present in the true network structure. Note that in contrast to the more standard methodology of supervised learning where a dataset contains a distinguished class variable, BNSL with model averaging builds this predictive model in an *unsupervised* manner. However, the key to our learning approach for determining effective thresholds is to setup threshold selection as a *supervised* machine learning task. To do this we need to specify how to obtain labeled data and which loss function is to be used in learning.

To obtain labeled data for our supervised approach, we start with ground truth Bayesian networks that are either existing human-constructed BNs or learned using a scoring function on a dataset. These ground truth networks are then sampled to generate datasets of various sizes from which sets of plausible networks are learned using a variety of scoring functions and model averaging methods. Model averaging is then used to obtain the estimates $\hat{P}(e_j)$ and the ground truth networks are used to determine whether the edge is present. We refer to a particular combination of ground truth network, dataset, scoring function, and model averaging method as an *instance*.

Algorithm 5.2: Determine the optimal threshold from proxy instances.

Input: $\{e_j\}_i$ indicating whether edge $e_j \in \{0, 1\}$ is present in the ground truth network of instance i.

Input: $\{\hat{e}_{jm}\}_i$ indicating whether edge $e_j \in \{0, 1\}$ is present in the collected m-th network of instance i.

Output: The optimal threshold c^*.

$c^* \leftarrow 0.0$;
$\text{metric}^* \leftarrow 0.0$; `/* To maximize the metric. */`
for $c \in \{0.05, 0.10, \cdots, 0.95\}$ **do**
  for $i \in \{1, 2, \cdots, N\}$ **do**
    for $j \in \{1, 2, \cdots, n(n-1)/2\}$ **do**
      $O_{e_j} = 0$; `/* Initialize the occurrences of edge` e_j. `*/`
      for $m \in \{1, 2, \cdots, M\}$ **do**
        $O_{e_j} = O_{e_j} + \hat{e}_{jm}$;
      end
      $\hat{e_j} = \frac{O_{e_j}}{M} > c$;
    end
    $\text{metric}_i = \text{Metric}(\{e_j\}_i, \{\hat{e_j}\}_i)$;
    if $metric_i > metric^*$ **then**
      $\text{metric}^* = \text{metric}_i$;
      $c^* = c$;
    end
  end
end

As loss functions or performance metrics for the effectiveness of a threshold on an instance, we use both the cost-insensitive SHD (H), and cost-sensitive F_β scores and mis-

classification cost C_α, and all of them are defined in Section 2.4.

For each instance, the value of the metrics can be determined by a threshold c that defines the set of significant edges, i.e., $\hat{e}_j = 1 \iff \hat{p}_j > c$. We cast the threshold learning problem as a transfer learning problem that aims to provide a threshold value for any target dataset using the value learned from a wide range of proxy datasets. Denote our threshold as $c^*(\beta)$ that determines the set of significant edges for all proxy instances at a given value of β. Let $F_\beta^* = \max_c F_\beta(c)$ be the optimal F_β value for each instance from the proxy datasets. The value of $c^*(\beta)$ can then be determined by minimizing the mean absolute error as follows,

$$c^*(\beta) = \underset{c}{\operatorname{argmin}} \frac{1}{K} \sum_{i=1}^{K} \mid [F_\beta^*]_i - [F_\beta(c)]_i \mid,$$

where K is the number of proxy instances. Similarly, the value of $c^*(\alpha)$ and $c^*(H)$ for misclassification cost and SHD, respectively, can be determined by minimizing the mean relative error,

$$c^*(\alpha) = \underset{c}{\operatorname{argmin}} \frac{1}{K} \sum_{i=1}^{K} \frac{\mid [C_\alpha^*]_i - [C_\alpha(c)]_i \mid}{n_i(n_i - 1)},$$

$$c^*(H) = \underset{c}{\operatorname{argmin}} \frac{1}{K} \sum_{i=1}^{K} \frac{\mid [H^*]_i - [H(c)]_i \mid}{n_i(n_i - 1)},$$

where n_i is the number of variables. We minimize over $c \in \{0.0, 0.05, ..., 0.95\}$ to avoid making finer distinctions that are not justified by the data and to make our recommendations for suitable thresholds more applicable in practice. We found that the values of $c^*(\beta)$, $c^*(\alpha)$, and $c^*(H)$ stay the same or almost the same if we minimize mean squared error instead of the mean absolute error, suggesting that the thresholds we learned are robust.

Finally, we evaluate the learned threshold values on a target dataset. The key to learning an effective threshold is to minimize over all proxy instances—combinations of ground truth network, dataset, scoring function, and search algorithm—rather than over individual edges to avoid having the contributions of larger networks in the training set overwhelm the contributions of smaller networks. We learned thresholds that are dependent on dataset size N, scoring function, and model averaging method as it was found that these additional features significantly altered the values of the threshold. The number of variables n in the dataset did not significantly alter the values of the threshold and was not included as an additional feature.

Table 5.3: UCI datasets (*left*, *middle*) and bnlearn Bayesian networks (*right*), where n is the number of variables in the dataset or network, and N is the number of instances in the original UCI dataset.

UCI dataset	n	N
shuttle	10	58,000
census income	14	48,842
letter	17	20,000
online shopping	18	12,330
lymphography	19	148
hepatitis	20	155
parkinsons	23	195
credit card	24	30,000

UCI dataset	n	N
robot navigation	25	5,456
horse colic	27	368
steel	28	1,941
flags	29	194
breast cancer	31	569
soybean	36	683
biodeg	42	1,055
spectf heart	45	267

network	n
sachs	11
child	20
insurance	27
water	32
mildew	35
alarm	37
barley	48
hailfinder	56
heparII	70
win95pts	76

In the next section, we experimentally compare our approach to other approaches. In the remainder of this chapter, we cast previous approaches into a common notation. Recall that the threshold in Broom et al. [7] uses a permutation test to determine the number of false positives. For the i-th instance, let $[G_p]_i = (\mathcal{X}, [E_p(c)]_i)$ be the learned BN in the permutation test and $[G_b]_i = (\mathcal{X}, [E_b(c)]_i)$ from the original dataset, where the significant edges in both networks are identified by the threshold c. The threshold seeks to maximize the gap between $[E_p(c)]_i$ and $[E_b(c)]_i$, formally $[c_B]_i = \text{argmax}_c\{\sum_{e_j \in [E_b(c)]_i} e_j - \sum_{e_j \in [E_p(c)]_i} e_j\}$. Similarly, the threshold in Gross et al. [39] can be defined as $[c_G]_i = 1/3 + \sqrt{2/M_i}$, where M_i is the size of the set of BNs collected for instance i. Since we proved that the optimization method in Scutari and Nagarajan [82] yields a fixed probability threshold for all instances, we can define such a threshold as $c_S = 0.5$.

As a baseline point of comparison, we also consider the special threshold values $c_{F_\beta} = 1/(1+\beta^2)$. Let F_β^* be the optimal F_β score achievable by a model. Then the optimal threshold $c_{F_\beta}^*$ is given by $c_{F_\beta}^* = F_\beta^*/(1+\beta^2)$ assuming that the probabilities are perfectly calibrated (see [60, 53, 101, 103]). Of course, in practice one does know F_β^*. However, for any instance we know that $F_\beta^* \leq 1$. Thus, the optimal threshold is always less than or equal to $1/(1+\beta^2)$ (but can be much less than $1/(1+\beta^2)$). In contrast to cost-sensitive classification (where the decision rule is directly derivable from the costs), the decision rule (i.e., the threshold) for F_β is not directly derivable and only bounds are known so far.

The threshold in the F_β case will be some function of training set size, as in general the maximum attainable F_β value increases as the amount of data increases. Besides c_{F_β}, we also use $c_{C_\alpha} = 1/(1+\alpha)$ as the optimal threshold for C_α assuming that the probabilities are perfectly calibrated (see, e.g., [33] and references therein). Note that c_B and c_G are unique for each instance whereas c_S, $c^*(\beta)$, $c^*(\alpha)$, c_{F_β}, c_{C_α} and $c^*(H)$ are applicable to all instances.

5.5 Meta Ensembles

In Chapter 4, we compared a broad range of scoring functions and examined which one is the best *single* scoring function to use under two ensemble frameworks, namely bootstrap (bagging) and (Bayesian) model averaging using the credible set. Although we find that qNML is the best contender for knowledge discovery using the exact credible set approach, and BDeu using bootstrapping, in most real world scenarios, there still exist many instances where the recommended score is outperformed by others. Since the threshold selection task is aimed at producing a prediction on whether the edge exists or not, we can use another layer of ensemble methods in the supervised learning scenario to boost the performance over using a single scoring function, and we call this layers of ensemble methods meta ensembles. Many ensemble generation techniques are applicable in combining information from different scores/network sets, and we consider two approaches to demonstrate the effectiveness of meta ensembles — majority/weighted voting over the results from all scoring functions, and gradient boosted trees for algorithmic combination of predictions (also called stacking).

As the name suggests, majority voting would consider each score as an equal party, and the prediction would be that the edge exists if more than half of the scores predicted its existence. Weighted voting goes a step further and allow us to give more credit to the scores that are more reliable than others. Despite our effort at trying out different weights, we are unable to produce better results than using a single score (qNML for the credible set and BDeu for bootstrap). Upon further examination, we note that unlike normal ensemble settings, we have a rather dominant learner (score) comparing to all learners, and so a linear combination of predictions without considering the context does not really work. We omit the voting results below and only focus on the non-linear gradient boosted tree.

To utilize predictions from each score with extra information about the context, we use the sample sizes and β or α values as the score invariant information, and the probability of the edge, the deviation from the threshold, and the predictions as the score specific

information in the final ensemble model. Then we feed all information to sklearn's GradientBoostingClassifier with default hyperparameters as an illustrative combiner algorithm. The final model also builds the training data using stratified sampling to prevent larger networks from dominating the learning process.

5.6 Experimental Evaluation

The structure learning and threshold selection experiments were conducted on a shared server with 346 GB RAM and Intel Xeon Gold 6148 at 2.4 GHz. For scoring the datasets memory usage was limited to 64 GB and for structure learning a limit of 128 GB was imposed. For both scoring and learning, a computation time limit of 24 hours was imposed for each instance. Properties of the datasets are summarized in Table 5.3.

Table 5.4: Comparison of threshold selection methods when the performance metric is SHD for the bootstrap and credible set model averaging approaches. At each row, the aggregated Borda count is shown when comparing the selection methods on a set of experiments that consist of three samples from each ground truth network and dataset sample sizes of $N = 50$, 100, 500, 1000, 5000, 10000.

Method	Scutari	Broom	Gross	ML
Bootstrap	5,568	2,275	5,864	**6,166**
Credible set	6,468	3,210	4,336	**6,853**

5.6.1 Setup

To empirically study the threshold selection methods, we considered a wide selection of datasets from the UCI repository[2] and networks from the bnlearn Bayesian network repository[3] (see Table 5.3). We preprocessed the UCI datasets using a k-nearest neighbor imputation algorithm, with $k = 5$, to fill in missing values and a supervised discretization method based on the MDL principle to discretize continuous variables. We used a total of 90 ground truth BNs: 10 ground truth BNs came from the bnlearn repository and

[2] https://archive.ics.uci.edu/ml
[3] https://www.bnlearn.com/bnrepository/

Table 5.5: Comparison of threshold selection methods when the performance metric is multi-class F_β score on *directed* edges. At each β, the aggregated Borda count is shown when comparing the selection methods on a set of experiments that consist of three samples from each ground truth network and dataset sample sizes of $N = 50$, 100, 500, 1000, 5000, 10000.

	β	Scutari	Broom	Gross	c_{F_β}	ML
Bootstrap	1/4	8,777	5,526	8,532	4,698	**9,034**
	1/2	8,169	6,201	8,612	4,614	**9,516**
	1	4,634	7,316	7,427	4,634	**9,117**
	2	5,071	8,092	6,447	8,347	**9,193**
	4	4,479	9,204	5,966	**9,288**	8,815
Credible set	1/4	**11,196**	7,034	8,513	6,336	10,728
	1/2	**10,130**	7,243	8,225	7,119	9,897
	1	5,932	**7,114**	6,307	5,932	6,460
	2	7,946	7,178	7,158	7,664	**8,841**
	4	7,684	7,540	6,883	8,552	**9,667**

a further 80 ground truth BNs were constructed following a similar approach to Liu et al. [61] by (i) scoring each of the 16 UCI datasets using each of the five scoring functions AIC [2], BIC/MDL [80, 55], BDeu [8, 42], qBDJ [93], and qNML [89] in turn, (ii) learning an optimal network structure from each scored dataset, and (iii) and fitting the parameters to each structure to give a final Bayesian network. Given the 90 ground truth BNs, we used the logic sampling function `rbn` from the bnlearn R package [81] to generate random samples of sizes $N = 50$, 100, 500, 1,000, 5,000, and 10,000 from the bif files. We collected three samples for each dataset size N, for a total of 18 samples for each ground truth BN.

To evaluate the threshold selection methods within the bootstrap approach to model averaging, we used the implementation available as the function `boot.strength` from the bnlearn R package [81]. We used the default replication factor of 200, the default equivalent sample size for BDeu of 1, and the tabu search algorithm, as in preliminary experiments the tabu algorithm performed better than the alternative hill climbing algorithm. Due to score availability in bnlearn, we only consider AIC, BDeu, and BIC in the bootstrap experiments. A total of 4,320 bootstrap experiments were performed.

To evaluate the scoring functions within the credible set model averaging framework, we implemented the scoring functions AIC, BDeu, BIC, and qNML in `Python` and used the

Table 5.6: Comparison of threshold selection methods when the performance metric is F_β score on undirected edges. At each β, the aggregated Borda count is shown when comparing the selection methods on a set of experiments that consist of three samples from each ground truth network and dataset sample sizes of $N = 50$, 100, 500, 1000, 5000, 10000.

	β	Scutari	Broom	Gross	c_{F_β}	ML
Bootstrap	1/4	8,328	2,265	6,855	8,474	**10,306**
	1/2	7,969	3,596	7,316	7,234	**9,060**
	1	4,583	5,883	6,629	4,583	**7,950**
	2	4,790	7,910	5,903	8,226	**8,944**
	4	4,134	8,942	5,218	**9,356**	9,030
Credible set	1/4	7,027	3,866	4,546	**8,823**	8,684
	1/2	6,981	4,760	5,068	**7,814**	7,763
	1	4,321	5,719	5,500	4,321	**7,031**
	2	5,394	6,261	6,322	7,535	**8,618**
	4	5,038	6,298	5,849	8,924	**9,753**

`eBNSL` package [58], an extended version of `GOBNILP` [6], for collecting the credible networks. For AIC and BIC, eBNSL sets the limit on the size of the parent set based on the pruning rule that guarantees optimality, whereas for other scores there is no known explicit limit. Then all networks falling into a Bayes factor (BF) of 150 were collected with a counting limit of 100,000. Since the scores under consideration are all score equivalent [18], i.e., BNs of the same equivalence class have the same score, the choice of exact structure learning algorithm does not affect the learned set of credible networks. We also set the equivalent sample size for BDeu to 1, while other scores do not have hyperparameters. A total of 5,940 credible set experiments were performed.

The values for c_G, c_S, $c^*(\beta)$, $c^*(\alpha)$, and $c^*(H)$ can be determined directly from the empirical probabilities. The values for c_{F_β} and c_{C_α} can be determined for specific β or α. For the method of Broom et al. [7], the threshold c_B requires a separate permutation test, where all columns of the dataset are randomized to create permuted datasets. The process is repeated 60 times as in the original paper since we want to replicate the original results. The threshold is then determined by the optimization step mentioned above. The method of Gross et al. [39] is a threshold for identifying *directed* edges. However, in their evaluation they adapt it to identify undirected edges and for C_α and F_β, we used their adaptation as well.

Table 5.7: Comparison of threshold selection methods when the performance metric is misclassification cost on undirected edges; i.e., $\alpha \times \text{FN} + \text{FP}$. At each α, the aggregated Borda count is shown when comparing the selection methods on a set of experiments that consist of three samples from each ground truth network and dataset sample sizes of $N = 50, 100, 500, 1000, 5000, 10000$.

	α	Scutari	Broom	Gross	c_{C_α}	ML
Bootstrap	1/4	7,785	1,925	6,307	9,144	**9,746**
	1/2	7,250	2,630	6,254	7,858	**8,335**
	1	5,277	3,433	5,931	5,277	**7,273**
	2	6,364	4,927	6,805	7,064	**7,774**
	4	5,503	6,777	6,504	8,119	**8,584**
Credible set	1/4	6,878	3,468	4,182	8,982	**9,540**
	1/2	6,403	3,613	4,056	7,587	**8,399**
	1	4,688	3,906	4,027	4,688	**6,798**
	2	6,098	5,158	4,843	5,112	**6,750**
	4	5,904	5,992	6,503	7,193	**7,808**

5.6.2 Results and Discussion

Comparison of Thresholds

The results from comparison of the threshold selection methods are summarized in Table 5.4 for SHD, Table 5.5 for multi-class F_β^{CPDAG}, Table 5.6 for binary F_β, and Table 5.7 for misclassification cost C_α. The experimental results are aggregated using the Borda count. For the SHD and misclassification cost C_α performance metrics, our proposed threshold is the clear winner. Recall that when $\beta < 1$, the F_β measure gives more weight to precision and vice versa. In Table 5.6 (F_β), the special threshold c_{F_β} has some advantages in a high recall case ($\beta = 8$) when the networks come from bootstrap, and in a high precision scenarios ($\beta < 1$) when the networks are collected using the credible set. In Table 5.5 (F_β^{CPDAG}), the special threshold c_{F_β} also maintains a slight lead in a high recall case ($\beta = 8$) when the networks come from bootstrap. The winners on the credible set are less clear with the constant threshold (Scutari) taking the high precision scenarios ($\beta < 1$), the permutation test (Broom) taking the F_1, and our approach (ML) taking the high recall scenarios ($\beta > 1$).

We present the values of the recommended thresholds for SHD in Table 5.8, for multi-

Table 5.8: Recommended thresholds for model averaging using the bootstrap and credible set approaches when the performance measure is the structural Hamming distance, for various dataset sizes N and scoring functions.

	Bootstrap			Credible set			
N	AIC	BDeu	BIC	AIC	BDeu	BIC	qNML
50, 100	0.35	0.50	0.25	0.60	0.95	0.50	0.75
500, 1000	0.45	0.30	0.25	0.50	0.50	0.50	0.50
5000, 10000	0.45	0.40	0.45	0.50	0.50	0.50	0.50

class F_β^{CPDAG} in Figure 5.1, for binary F_β score in Figure 5.2, and for misclassification cost C_α in Figure 5.3. In Table 5.8, the preferred thresholds using bootstrap are always below 0.5, regardless of the scoring functions. On the other hand, using credible set we can often leave the threshold at 0.5 given a moderate amount of data. Recall that when $\beta < 1$, the F_β measure gives more weight to precision and vice versa. As expected, the threshold shifts lower as we give more weight to recall and less to precision, i.e., from $\beta = 1/8$ to $\beta = 8$. As the dataset size grows, the threshold curve tends to shift higher, indicating the presence of more spurious edges. Note that with sufficient data and when the networks are collected using credible set (bottom right, Figure 5.1), the preferred threshold stays at 0.5 for all values of β under consideration. This is consistent with our observations on SHD because both metrics operate on CPDAG. On misclassification cost (Figure 5.3), we can observe the same trends as when $\alpha < 1$, the C_α puts more pressure on reducing FP (which leads to better precision), and vice versa. Given a data analysis scenario, an application-specific tradeoff between precision and recall can be used to determine the recommended threshold from the given plots.

As is standard, our final recommended thresholds were learned using all of the instances. However, when evaluating and comparing to previous proposals, we used leave-one-benchmark-out cross-validation. For example, if we want to get the F_β values on *alarm*, we use all other benchmarks such as *insurance* and *barley* to learn the threshold $c^*(\beta)$ and then identify significant edges in *alarm* with such a threshold.

Our examination of the application literature (see Table 5.1) indicates a strong preference for precision over recall; i.e., $\beta < 1$. For example, the thresholds in the two medicine studies [62, 64] are consistent with our recommended values for $\beta = 1/8$, whereas the other medicine study [73] used a threshold corresponding to $\beta = 1/2$. Our approach allows one to explicitly incorporate domain knowledge and target metrics in setting the threshold,

and thus removes a potentially confounding factor from a data analysis.

Bootstrap vs. Credible Set

The results from comparison of bootstrap vs. the credible set are summarized in Figure 5.4 for SHD and multi-class F_β^{CPDAG}, and in Figure 5.5 for binary F_β and misclassification cost C_α. The curves in the plot represents the cumulative metrics values, possibly on log scale. The SHD plot (top, Figure 5.4) shows that credible set with the qNML score has a clear advantage over all other combinations (lower is better). The same conclusion can be drawn from the multi-class F_β^{CPDAG} plot (bottom, Figure 5.4) where higher is better, and from the misclassification cost C_α plot (top, Figure 5.5) where lower is better. The binary F_β plot (bottom, Figure 5.5), however, shows some mixed results with bootstrap with the BDeu score taking the lead for a third of the instances, and credible set with the qNML score picking up with more instances.

The results suggest that the credible set approach produces better structure reconstruction results than the bootstrap approach in most cases, and that the qNML score works very well with the credible set to achieve great network reconstruction quality in the model averaging paradigm.

5.6.3 Ensemble Results

In this context, ensemble generation relies heavily on being able to score efficiently, so the two problems work together.

The results from comparison of bootstrap meta ensemble with bootstrap model averaging method are summarized in Figure 5.6 for SHD and multi-class F_β^{CPDAG}, and in Figure 5.7 for binary F_β and misclassification cost C_α. The curves in the plot represents the cumulative metrics values, possibly on log scale. The SHD plot (top, Figure 5.6) shows that bootstrap meta ensemble is on par with the bootstrap model averaging method with BDeu (lower is better), and the same can be observed also in the misclassification cost C_α plot (top, Figure 5.7) where lower is better. On the other hand, both the multi-class F_β^{CPDAG} plot (bottom, Figure 5.6) and the binary F_β plot (bottom, Figure 5.7) show that the ensemble outperforms other scores.

The results from comparison of credible set meta ensemble with credible set model averaging method are summarized in Figure 5.8 for SHD and multi-class F_β^{CPDAG}, and in Figure 5.9 for binary F_β and misclassification cost C_α. The curves in the plot represents

the cumulative metrics values, possibly on log scale. The SHD plot (top, Figure 5.8) shows that credible set meta ensemble has mixed results with the credible model averaging method with the best scores (qNML and BDeu) (lower is better), and the same can be observed also the multi-class F_β^{CPDAG} plot (bottom, Figure 5.8), where higher is better. The misclassification cost C_α plot (top, Figure 5.9) where lower is better, and the binary F_β plot (bottom, Figure 5.9), where higher is better, show that the ensemble clearly outperforms other scores.

When we combine multiple scores in a specific network collection approach, the binary F_β always reflects superior performance over results from using a single score, whereas the SHD has more mixed results that are on par with the best choice of scores within each framework. Since there is an uneven improvement brought by combining scores within each framework, by further combining results from both frameworks, we expect that all four metrics should have some improvements over the previous best results using a single score in each framework.

The results from comparison of meta ensemble using combined features with bootstrap and credible set ensemble methods are summarized in Figure 5.10 for SHD and multi-class F_β^{CPDAG}, and in Figure 5.11 for binary F_β and misclassification cost C_α. The curves in the plot represents the cumulative metrics values, possibly on log scale. The SHD plot (top, Figure 5.8) shows perhaps the least amount of improvement over bootstrap or credible set model averaging method with the best scores (qNML and BDeu) (lower is better), which is consistent with the earlier observation that SHD is on par with the best score within each framework. The other 3 metrics show that the meta ensemble combining both scores and frameworks clearly outperforms other approaches using a single score.

We also explored whether building separate models based on the dataset size would improve the results, and we note that building three separate models based on the sample size groups (50, 100), (500, 1,000), and (5,000, 10,000) does not make a difference over building a single model with sample size as a feature.

5.7 Summary

A fundamental step in the data analysis methodology using BNs is to identify significant edges from a set of BNs learned with the well-known score-and-search approach. Selecting a reasonable threshold has broad implications for the success of the analysis. However, the problem of selecting a good threshold has received limited attention in the literature. In this chapter, we identified an important shortcoming in a widely used threshold selection

method. In particular, we proved that the optimization method of Scutari and Nagarajan [82] is in fact equivalent to a fixed threshold of $c = 0.5$ for all instances. We then proposed a simple transfer learning approach for maximizing a target metric and selecting a threshold that can be generalized from proxy datasets to the target dataset. We addressed the imbalance classification problem of edges by considering both the structural Hamming distance and the cost-sensitive F_β and C_α measures with varying tradeoffs between precision/FP and recall/FN. We also pushed the boundaries of performance by using meta ensembles on top of the existing model averaging methods. In our experimental evaluation on a broad set of benchmarks from bnlearn and UCI datasets, the proposed threshold performs significantly better than previous approaches in almost all scenarios and performs competitively on the others. We also demonstrated that the credible set approach produces better structure reconstruction results than the bootstrap approach in most cases, and that the qNML score works very well with the credible set to achieve great network reconstruction quality in the model averaging paradigm. In addition, the meta ensembles combining across scores and model averaging frameworks further improve all metrics over using a single score in a framework. Considering the bootstrap, credible set, and the meta ensembles, we conclude that performance wise the ranking would be meta ensembles $>$ credible set $>$ bootstrap, whereas resource wise it would be the reverse. Our results suggest that one needs to make a more intricate and informed decision of the threshold on edges no matter which sampling methods or scores are used in learning the structures.

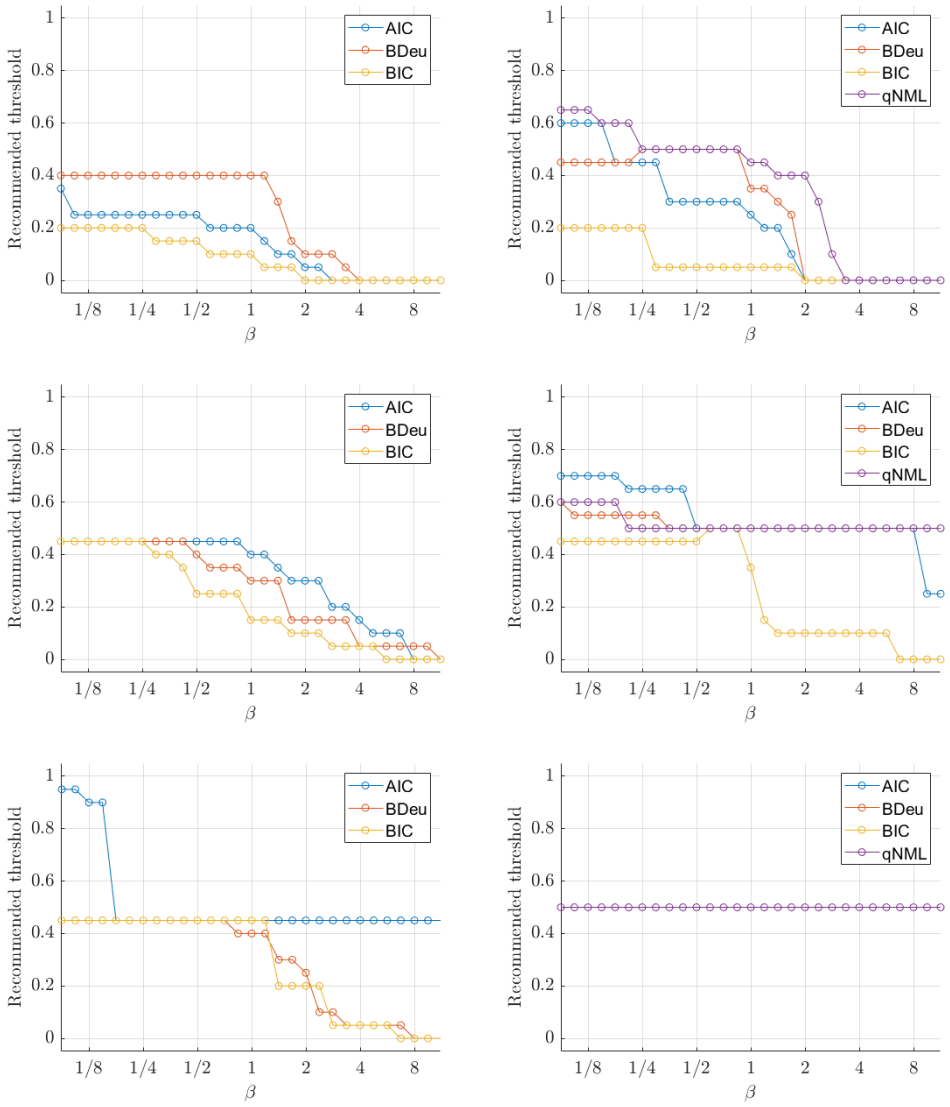


Figure 5.1: Recommended thresholds for model averaging using the bootstrap (lhs) and credible set (rhs) approaches when the performance measure is the multi-class F_β score for predicting *directed* edges, for various β, dataset sizes N, and scoring functions; (top) $N = 50, 100$; (middle) $N = 500, 1000$; (bottom) $N = 5000, 10000$. Note that the four curves in the bottom right plot perfectly overlap.

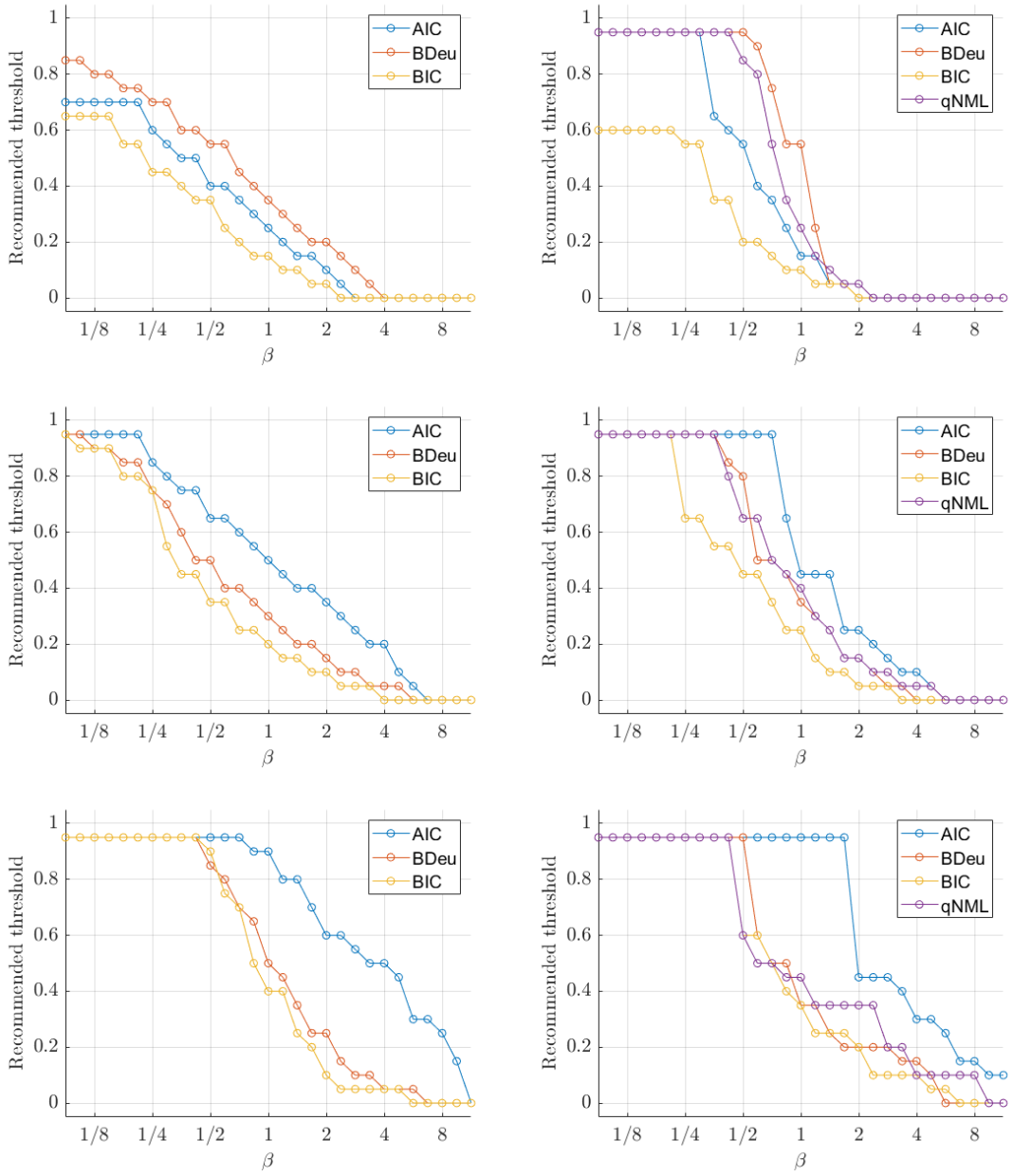


Figure 5.2: Recommended thresholds for model averaging using the bootstrap (lhs) and credible set (rhs) approaches when the performance measure is the F_β score for predicting undirected edges, for various β, dataset sizes N, and scoring functions; (top) $N = 50, 100$; (middle) $N = 500, 1000$; (bottom) $N = 5000, 10000$;

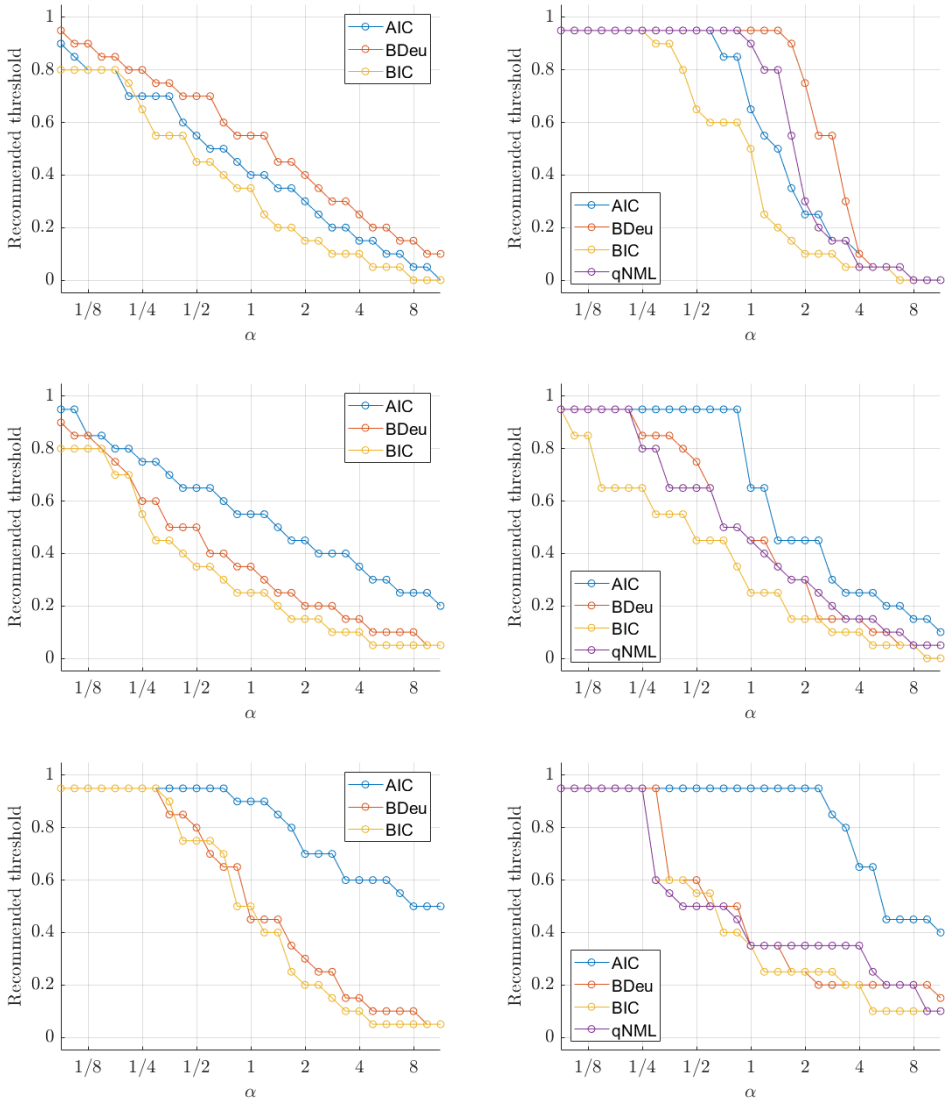


Figure 5.3: Recommended thresholds for model averaging using the bootstrap (lhs) and credible set (rhs) approaches when the performance measure is the misclassification cost $\alpha \times \text{fn} + \text{fp}$ for predicting *undirected* edges, for various α, dataset sizes N, and scoring functions; (top) $N = 50, 100$; (middle) $N = 500, 1000$; (bottom) $N = 5000, 10000$.

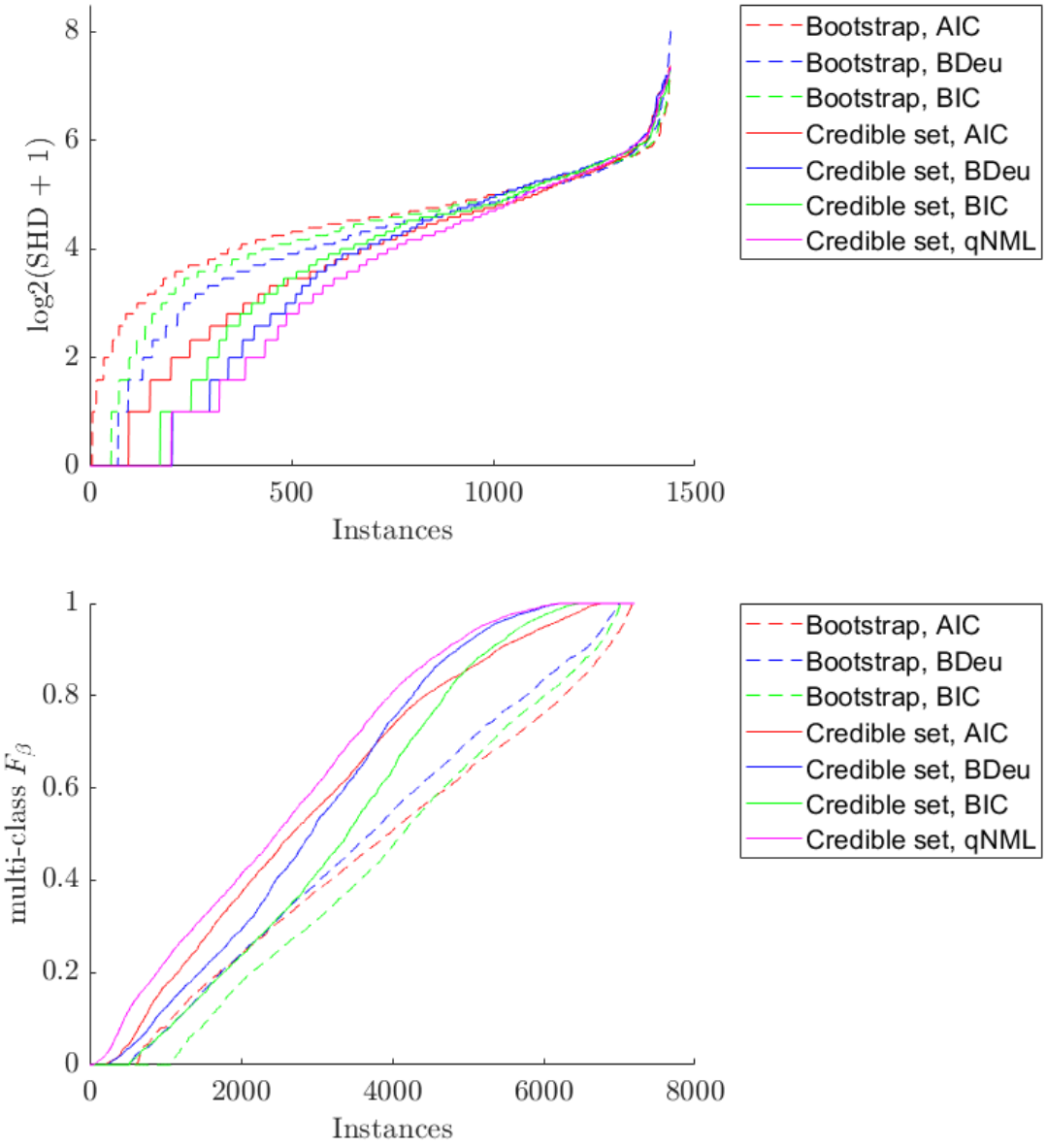


Figure 5.4: Comparison of bootstrap and credible set model averaging methods, for various scoring functions and performance measures for directed edges: Structural Hamming distance (top); multi-class F_β score (bottom). All methods used machine learned thresholds.

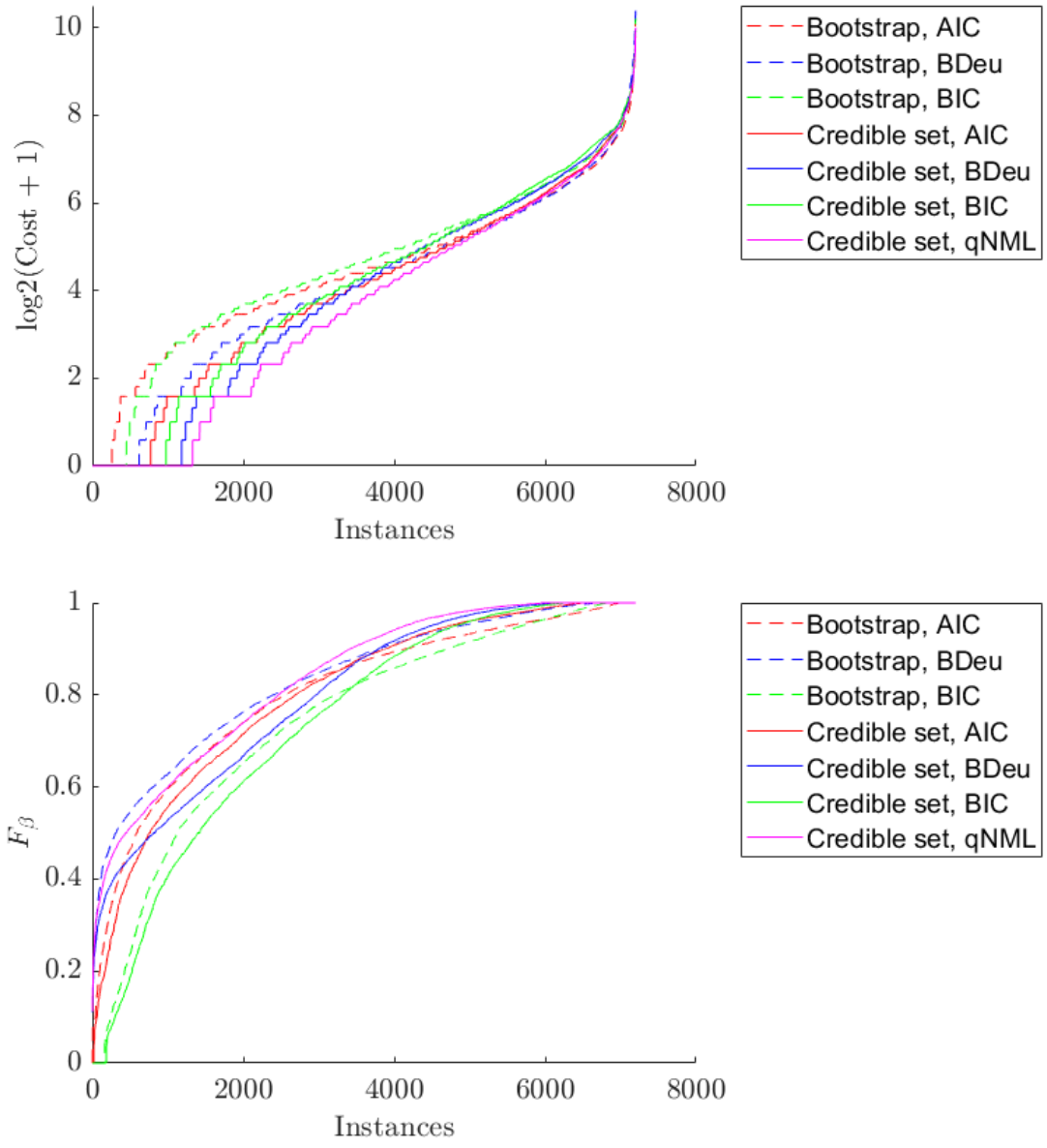


Figure 5.5: Comparison of bootstrap and credible set model averaging methods, for various scoring functions and performance measures for undirected edges (skeleton): Misclassification cost (top); F_β score (bottom). All methods used machine learned thresholds.

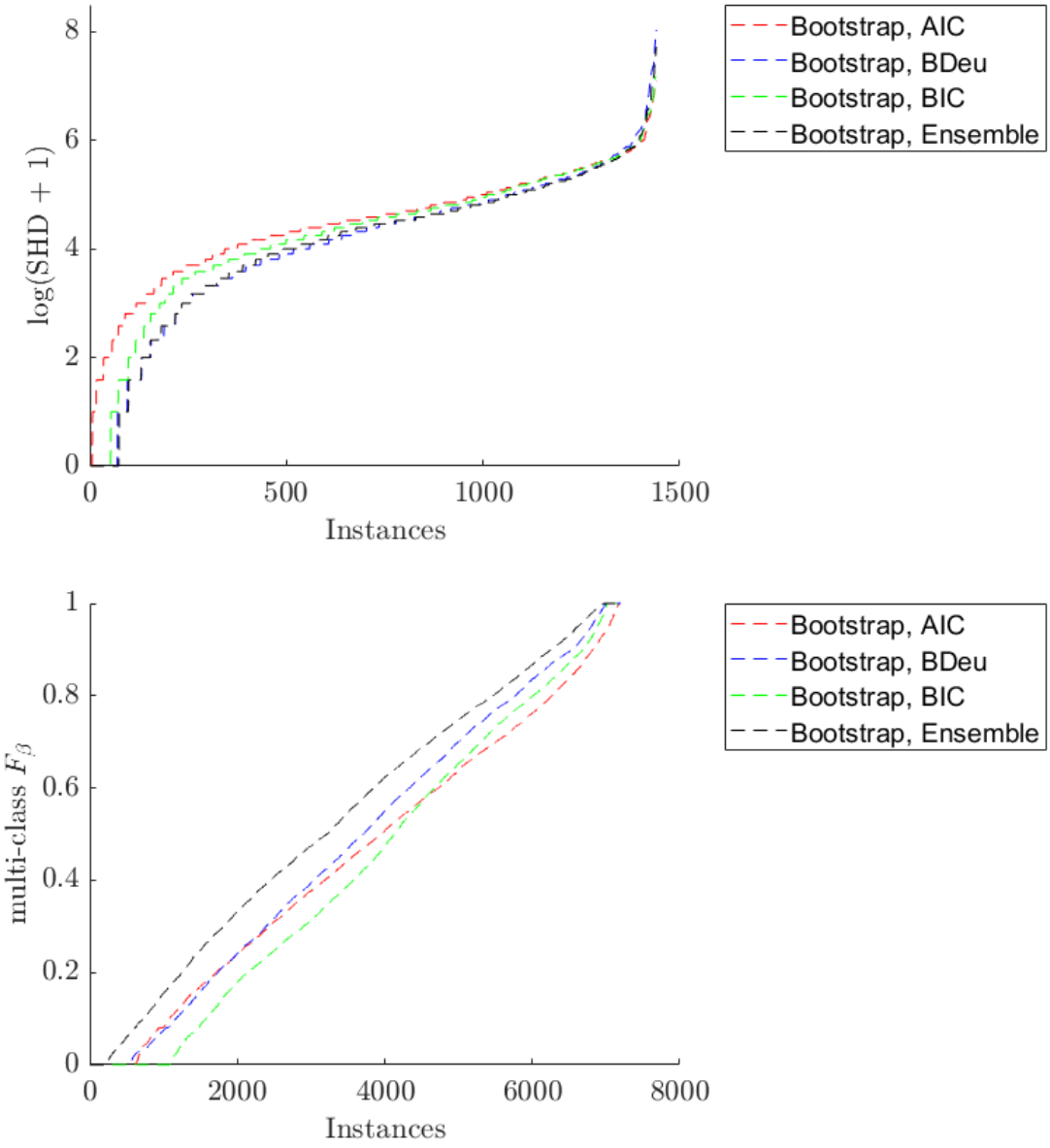


Figure 5.6: Comparison of bootstrap meta ensemble with bootstrap model averaging method, for various scoring functions and performance measures for directed edges: Structural Hamming distance (top); multi-class F_β score (bottom). All methods used machine learned thresholds.

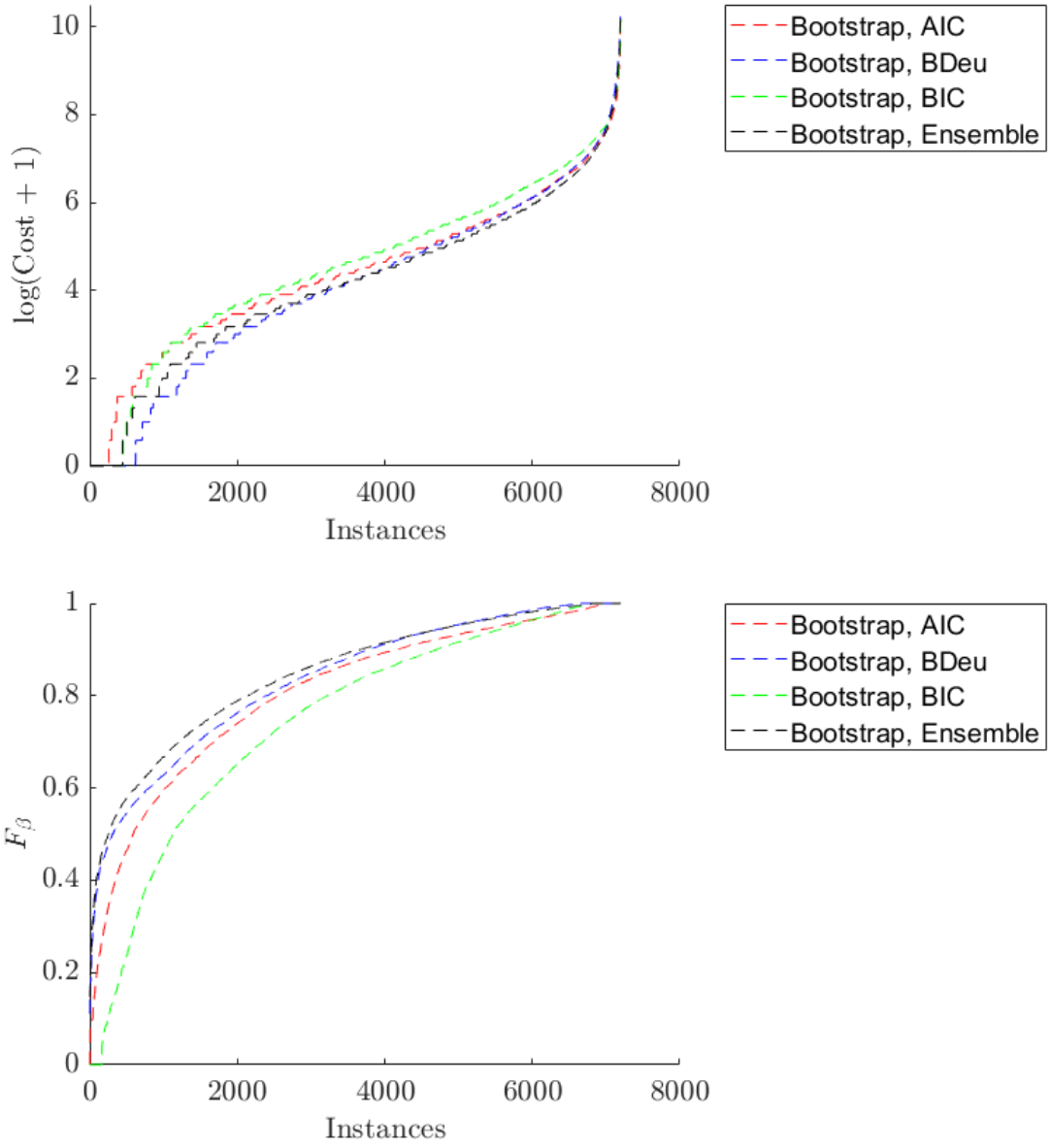


Figure 5.7: Comparison of bootstrap meta ensemble with bootstrap model averaging method, for various scoring functions and performance measures on undirected edges (skeleton): Misclassification cost (top); F_β score (bottom). All methods used machine learned thresholds.

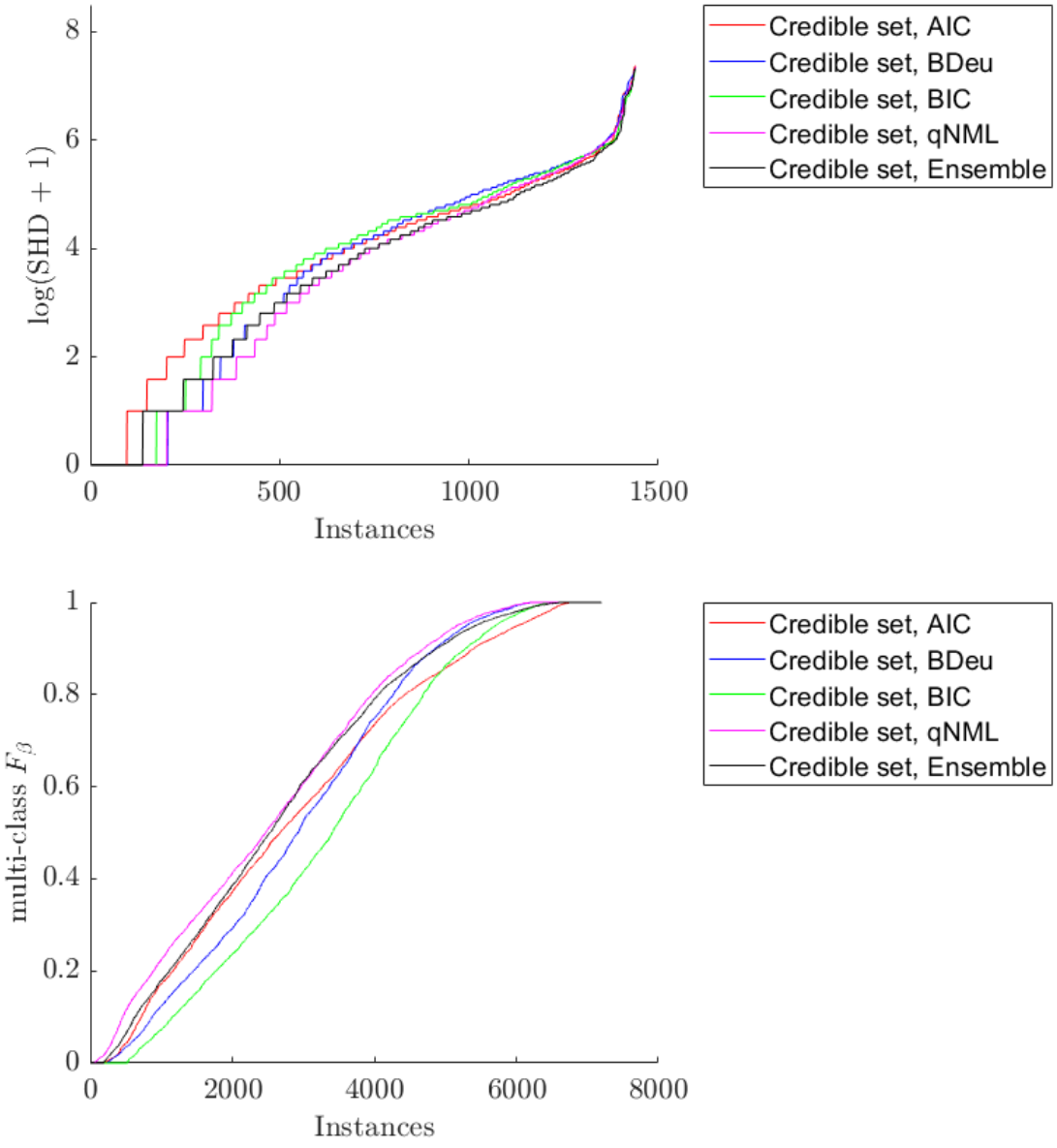


Figure 5.8: Comparison of credible set meta ensemble with credible set model averaging method, for various scoring functions and performance measures for directed edges: Structural Hamming distance (top); multi-class F_β score (bottom). All methods used machine learned thresholds.

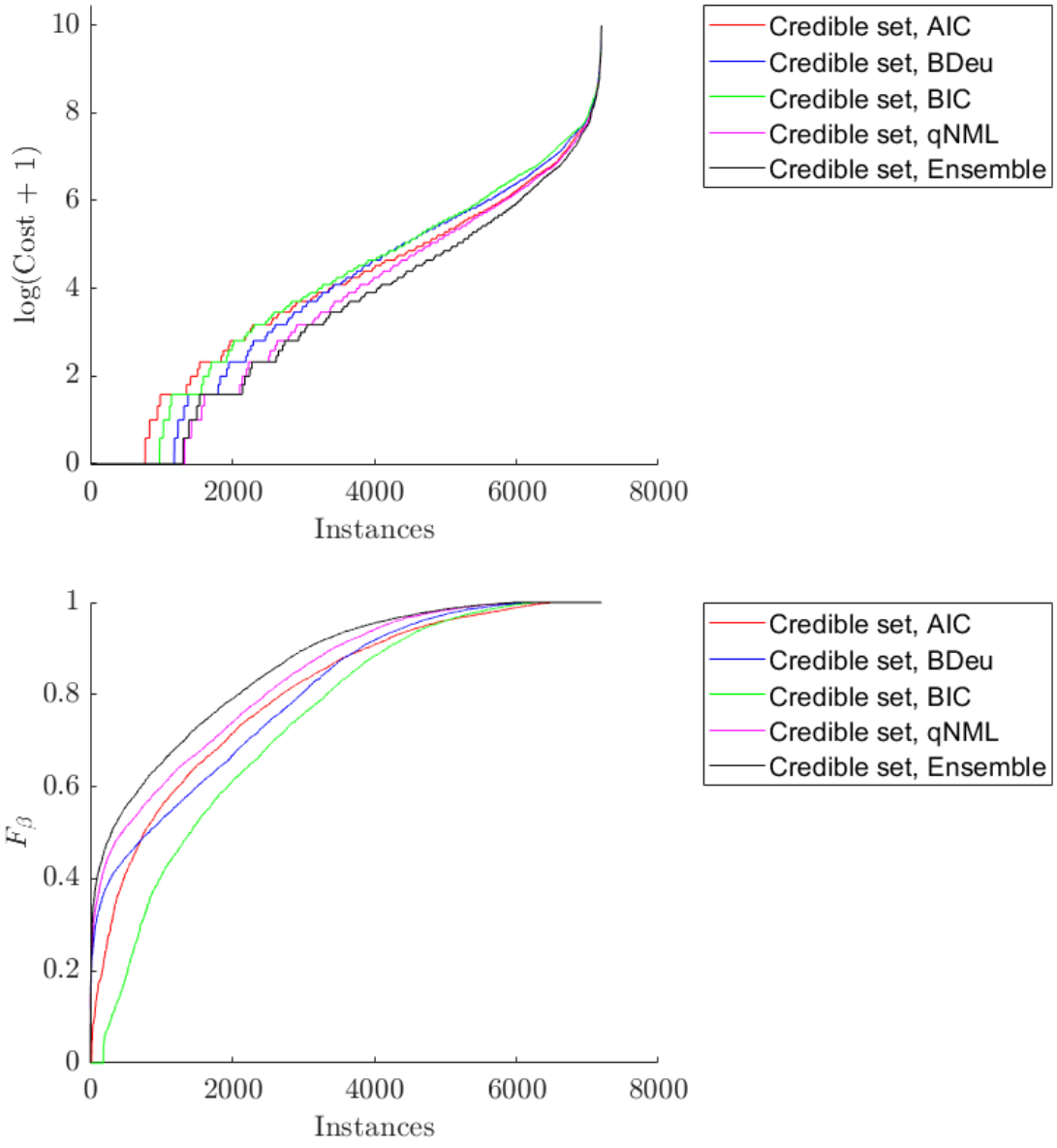


Figure 5.9: Comparison of credible set meta ensemble with credible set model averaging method, for various scoring functions and performance measures on undirected edges (skeleton): Misclassification cost (top); F_β score (bottom). All methods used machine learned thresholds.

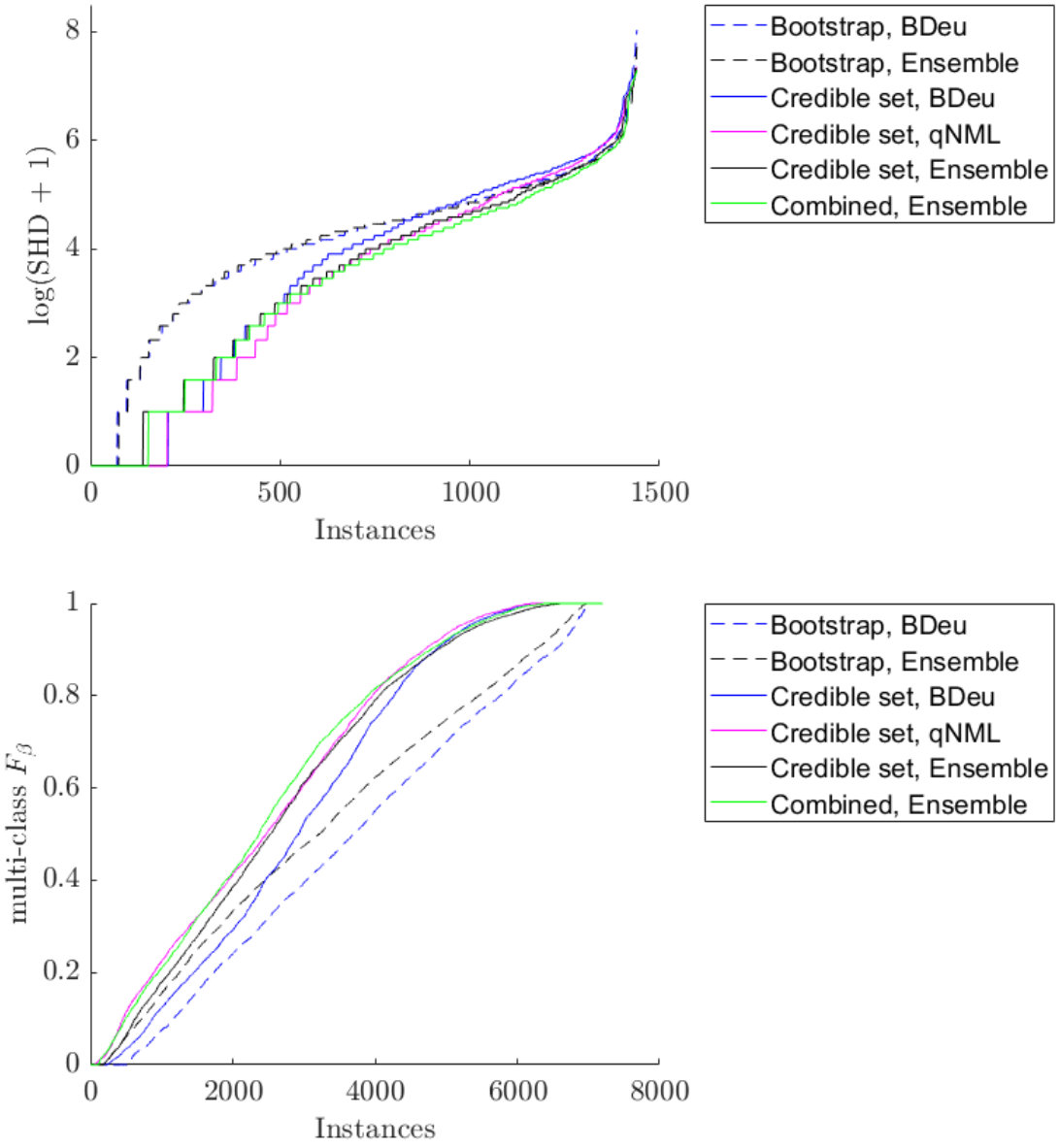


Figure 5.10: Comparison of meta ensemble using combined features with bootstrap and credible set ensemble methods, for various performance measures for directed edges: Structural Hamming distance (top); multi-class F_β score (bottom). All methods used machine learned thresholds.

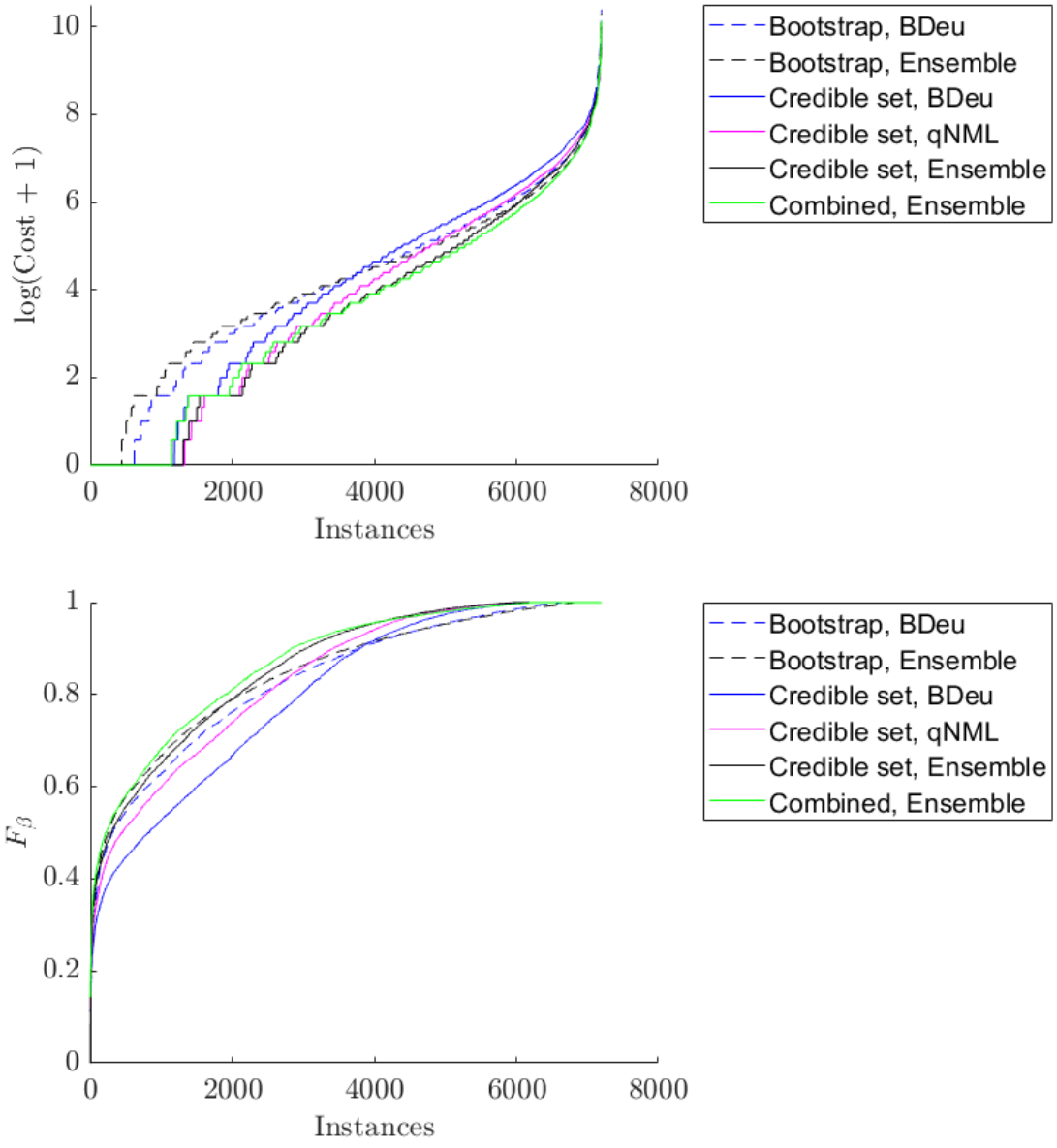


Figure 5.11: Comparison of meta ensemble using combined features with bootstrap and credible set ensemble methods, for various performance measures on undirected edges (skeleton): Misclassification cost (top); F_β score (bottom). All methods used machine learned thresholds.

Chapter 6

Conclusions

This thesis presents a few improvements to BNSL in the model averaging paradigm. Existing exact approaches for model averaging for BNSL either severely restrict the structure of the Bayesian network or have only been shown to scale to networks with fewer than 30 random variables. In Chapter 3, we proposed the credible set approach to model averaging inspired by performance guarantees in approximation algorithms that considers all networks within a factor of optimal. Our approach has two primary advantages. First, our approach only considers *credible* models in that they are optimal or near-optimal in score. Second, our approach is significantly more efficient and scales to much larger Bayesian networks than existing exact approaches. We modified `GOBNILP` [6], a state-of-the-art method for finding an optimal Bayesian network, to implement our generalized pruning rules and to find all *near*-optimal networks. Our experimental results demonstrate that the modified `GOBNILP` scales to significantly larger networks without resorting to restricting the structure of the Bayesian networks that are learned.

Although the credible set approach is score agnostic, scoring functions can conflict in their rankings and previous work has empirically studied their effectiveness with an aim to provide recommendations on their use. However, previous studies on scoring functions are limited by the small number and scale of the instances used in the evaluation and by a focus on learning a single network. To provide more practical insights about the scores, we studied five discrete scoring functions for BNSL in Chapter 4, namely AIC, BIC, qNML, BDeu, and qBDJ. The credible set approach allow us to scale our experiments to large BNs using an extension to `GOBNILP`, and to evaluate the scores with confidence measures on structure discovery. We have addressed previous design limits by considering multiple metrics for structure discovery including the SHD, the F-beta-measure, and the misclassification cost. These cost sensitive metrics present a full picture with varying

tradeoffs between precision vs. recall and FP vs. FN. We used both the ground truth BNs from bnlearn and real world UCI datasets in our structure learning tasks, and we are the first to provide an extensive experimental study of scoring functions in a model averaging framework. Contrary to previous recommendations in [61], we find that qNML is the best contender for knowledge discovery using the exact credible set approach, and BDeu using bootstrapping, in most real world scenarios. Our empirical study provides an insightful look at discrete score functions for BNSL and closes the gap in evaluating BN structures with confidence measures.

With the desired scoring function and the model averaging framework, one can obtain strength measures on all potential edges in a BN. A fundamental step in the data analysis methodology using BNs is to identify significant edges from a set of BNs learned with the well-known score-and-search approach. Selecting a reasonable threshold has broad implications for the success of the analysis. However, the problem of selecting a good threshold has received limited attention in the literature. In Chapter 5, we identified an important shortcoming in a widely used threshold selection method. In particular, we proved that the optimization method of Scutari and Nagarajan [82] is in fact equivalent to a fixed threshold of $c = 0.5$ for all instances. We then proposed a simple transfer learning approach for maximizing a target metric and selecting a threshold that can be generalized from proxy datasets to the target dataset. We addressed the imbalanced classification problem of edges by considering both the structural Hamming distance and the cost-sensitive F_β and C_α measures with varying tradeoffs between precision/FP and recall/FN. We also pushed the boundaries of performance by using meta ensembles on top of the existing model averaging methods. In our experimental evaluation on a broad set of benchmarks from bnlearn and UCI datasets, the proposed threshold performs significantly better than previous approaches in almost all scenarios and performs competitively on the others. We also demonstrated that the credible set approach produces better structure reconstruction results than the bootstrap approach in most cases, and that the qNML score works very well with the credible set to achieve great network reconstruction quality in the model averaging paradigm. In addition, the meta ensembles combining across scores and model averaging frameworks further improve all metrics over using a single score in a framework. Considering the bootstrap, credible set, and the meta ensembles, we conclude that performance wise the ranking would be meta ensembles $>$ credible set $>$ bootstrap, whereas resource wise it would be the reverse. Our results suggest that one needs to make a more intricate and informed decision of the threshold on edges no matter which sampling methods or scores are used in learning the structures.

These results will be of interest to all researchers interested in using BN as a data analysis tool in practice. We have seen studies using BN in banking [4], biology [12, 32],

medicine [62, 64, 73], safety [45], software [54], and sports [35]. Future studies using the same data analysis framework could benefit from the improvements made in this thesis, which allows researchers to draw conclusions from their observations in a principled way with proper confidence measures.

References

[1] Silvia Acid, Luis M. de Campos, and Javier G. Castellano. Learning Bayesian network classifiers: Searching in a space of partially directed acyclic graphs. *Machine Learning*, 59(3):213–235, 2005.

[2] Hirotugu Akaike. Information theory and the maximum likelihood principle. In *Proceedings of the Second International Symposium on Information Theory*, pages 267–281, 1973.

[3] Constantin F. Aliferis, Alexander Statnikov, Ioannis Tsamardinos, Subramani Mani, and Xenofon D. Koutsoukos. Local causal and Markov blanket induction for causal discovery and feature selection for classification. Part I: Algorithms and empirical evaluation. *Journal of Machine Learning Research*, 11(1), 2010.

[4] Ioannis Anagnostou, Javier Sanchez, Sumit Sourabh, and Drona Kandhai. Contagious defaults in a credit portfolio: A Bayesian network approach. *Journal of Credit Risk*, 16:1–26, 2020.

[5] Xue Bai, Rema Padman, Joseph Ramsey, and Peter Spirtes. Tabu search-enhanced graphical models for classification in high dimensions. *INFORMS Journal on Computing*, 20(3):423–437, 2008.

[6] Mark Bartlett and James Cussens. Advances in Bayesian network learning using integer programming. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pages 182–191, 2013.

[7] Bradley M. Broom, Kim-Anh Do, and Devika Subramanian. Model averaging strategies for structure learning in Bayesian networks with limited data. *BMC Bioinformatics*, 13(Suppl 13), 2012.

[8] Wray L. Buntine. Theory refinement of Bayesian networks. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 52–60, 1991.

[9] Kenneth P. Burnham and David R. Anderson. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer, 2nd edition, 2002.

[10] Cory Butz, Jhonatan S Oliveira, and Robert Peharz. Sum-product network decompilation. In *International Conference on Probabilistic Graphical Models*, pages 53–64. PMLR, 2020.

[11] Cory J. Butz, André E. dos Santos, Jhonatan S. Oliveira, and Christophe Gonzales. On a simple method for testing independencies in Bayesian networks. *Computational Intelligence*, 34(3):789–801, 2018.

[12] Cory J. Butz, André E. dos Santos, Jhonatan S. Oliveira, and John Stavrinides. Efficient examination of soil bacteria using probabilistic graphical models. In *International Conference on Industrial, Engineering and other Applications of Applied Intelligent Systems*, pages 315–326. Springer, 2018.

[13] Alexandra M. Carvalho. Scoring functions for Bayesian networks. INESC-ID Tech. Rep. 54, 2009.

[14] Eunice Yuh-Jie Chen, Arthur Choi, and Adnan Darwiche. Learning Bayesian networks with non-decomposable scores. In *Proceedings of the Fourth IJCAI Workshop on Graph Structures for Knowledge Representation and Reasoning*, pages 50–71, 2015. Available as: LNAI 9501.

[15] Eunice Yuh-Jie Chen, Arthur Choi, and Adnan Darwiche. Enumerating equivalence classes of Bayesian networks using EC graphs. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Statistics*, pages 591–599, 2016.

[16] Eunice Yuh-Jie Chen, Adnan Darwiche, and Arthur Choi. On pruning with the MDL score. *International Journal of Approximate Reasoning*, 92:363–375, 2018.

[17] Yetian Chen and Jin Tian. Finding the k-best equivalence classes of Bayesian network structures for model averaging. In *Proceedings of the Twenty-Eighth Conference on Artificial Intelligence*, pages 2431–2438, 2014.

[18] David M. Chickering. Learning equivalence classes of Bayesian network structures. *Journal of Machine Learning Research*, 2:445–498, 2002.

[19] David M. Chickering, Christopher Meek, and David Heckerman. Large-sample learning of Bayesian networks is NP-hard. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pages 124–133, 2003.

[20] David M. Chickering, David Heckerman, and Christopher Meek. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5: 1287–1330, 2004.

[21] Gerda Claeskens and Nils Lid Hjort. *Model Selection and Model Averaging*. Cambridge University Press, 2008.

[22] Diego Colombo and Marloes H. Maathuis. Order-independent constraint-based causal structure learning. *Journal of Machine Learning Research*, 15(1):3741–3782, 2014.

[23] Gregory F. Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Journal of Machine Learning Research*, 9:309–347, 1992.

[24] James Cussens. GOBNILP: Learning Bayesian network structure with integer programming. In *International Conference on Probabilistic Graphical Models*, pages 605–608, 2020.

[25] James Cussens and Mark Bartlett. GOBNILP 1.2 user/developer manual. *University of York, York*, 2012.

[26] Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

[27] Denver Dash and Gregory F. Cooper. Model averaging for prediction with discrete Bayesian networks. *Journal of Machine Learning Research*, 5:1177–1203, 2004.

[28] Cassio P. de Campos and Qiang Ji. Efficient structure learning of Bayesian networks using constraints. *Journal of Machine Learning Research*, 12:663–689, 2011.

[29] Cassio P. de Campos, Mauro Scanagatta, Giorgio Corani, and Marco Zaffalon. Entropy-based pruning for learning Bayesian networks using BIC. *Artificial Intelligence*, 260:42–50, 2018.

[30] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

[31] Usama Fayyad and Keki Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1022–1029, 1993.

[32] Wafa Feki-Sahnouna, Asma Hamzaa, Hasna Njahb, Mabrouka Barrajd, Nouha Mahfoudia, Ahmed Rebaie, and Malika Bel Hassend. A Bayesian network approach to determine environmental factors controlling Karenia selliformis occurrences and blooms in the Gulf of Gabès, Tunisia. *Harmful Algae*, 63:119–132, 2017.

[33] Peter A. Flach. Classifier calibration. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning and Data Mining*. Springer, 2016.

[34] Nir Friedman, Moises Goldszmidt, and Abraham Wyner. Data analysis with Bayesian networks: A bootstrap approach. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 196–205, 1999.

[35] Pilar Fuster-Parra, Alexandre García-Mas, Francisco Javier Ponseti Verdaguer, and Francisco Miguel Leo. Team performance and collective efficacy in the dynamic psychology of competitive team: A Bayesian network analysis. *Human Movement Science*, 40:98–118, 2015.

[36] José A. Gámez, Juan L. Mateo, and José M. Puerta. Learning Bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood. *Data Mining and Knowledge Discovery*, 22(1-2):106–148, 2011.

[37] Maxime Gasse, Alex Aussem, and Haytham Elghazel. A hybrid algorithm for Bayesian network structure learning with application to multi-label learning. *Expert Systems with Applications*, 41(15):6755–6772, 2014.

[38] Ambros Gleixner, Michael Bastubbe, Leon Eifler, Tristan Gally, Gerald Gamrath, Robert Lion Gottwald, Gregor Hendel, Christopher Hojny, Thorsten Koch, Marco E. Lübbecke, Stephen J. Maher, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Franziska Schlösser, Christoph Schubert, Felipe Serrano, Yuji Shinano, Jan Merlin Viernickel, Matthias Walter, Fabian Wegscheider, Jonas T. Witt, and Jakob Witzig. The SCIP Optimization Suite 6.0. Technical report, Optimization Online, July 2018. URL http://www.optimization-online.org/DB_HTML/2018/07/6692.html.

[39] Tadeu Junior Gross, Michel Bessani, Willian Darwin Junior, Renata Bezerra Araújo, Francisco Assis Carvalho Vale, and Carlos Dias Maciel. An analytical threshold for combining Bayesian networks. *Knowledge-Based Systems*, 175:36–49, 2019.

[40] Ru He, Jin Tian, and Huaiqing Wu. Structure learning in Bayesian networks of moderate size by efficient sampling. *Journal of Machine Learning Research*, 17:1–54, 2016.

[41] David Heckerman. A tutorial on learning Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, 1995.

[42] David Heckerman, Dan Geiger, and David M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20: 197–243, 1995.

[43] Jennifer A. Hoeting, David Madigan, Adrian E. Raftery, and Chris T. Volinsky. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–401, 1999.

[44] Estevam R. Hruschka Jr. and Nelson F. F. Ebecken. Towards efficient variables ordering for Bayesian networks classifier. *Data & Knowledge Engineering*, 63(2): 258–269, 2007.

[45] Steven Hwang, Linda Ng Boyle, and Ashis G. Banerjee. Identifying characteristics that impact motor carrier safety using Bayesian networks. *Accident Analysis and Prevention*, 128:40–45, 2019.

[46] Nathalie Japkowicz and Mohak Shah. *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press, 2011.

[47] Sir Harold Jeffreys. *Theory of Probability: 3d Ed.* Clarendon Press, 1967.

[48] Markus Kalisch and Peter Bühlman. Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *Journal of Machine Learning Research*, 8(3), 2007.

[49] Robert E. Kass and Adrian E. Raftery. Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795, 1995.

[50] Mikko Koivisto and Kismat Sood. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5:549–573, 2004.

[51] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[52] Kevin B. Korb and Ann E. Nicholson. *Bayesian Artificial Intelligence*. CRC press, 2010.

[53] Oluwasanmi Koyejo, Nagarajan Natarajan, Pradeep Ravikumar, and Inderjit S. Dhillon. Consistent binary classification with generalized performance metrics. In *Advances in Neural Information Processing Systems*, pages 2744–2752, 2014.

[54] Andrey Krutauz, Tapajit Dey, Peter C. Rigby, and Audris Mockus. Do code review measures explain the incidence of post-release defects? *Empirical Software Engineering*, pages 1–34, 2020.

[55] Wai Lam and Fahiem Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10:269–293, 1994.

[56] Wai Lam and Fahiem Bacchus. Using new data to refine a Bayesian network. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 383–390, 1994.

[57] Colin Lee and Peter van Beek. Metaheuristics for score-and-search Bayesian network structure learning. In *Canadian Conference on Artificial Intelligence*, pages 129–141. Springer, 2017.

[58] Zhenyu A. Liao, Charupriya Sharma, James Cussens, and Peter van Beek. Finding all Bayesian network structures within a factor of optimal. In *Proceedings of the Thirty-Third Conference on Artificial Intelligence*, volume 33, pages 7892–7899, 2019.

[59] Charles X. Ling and Victor S. Sheng. Cost-sensitive learning. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning and Data Mining*. Springer, 2016.

[60] Zachary C. Lipton, Charles Elkan, and Balakrishnan Naryanaswamy. Optimal thresholding of classifiers to maximize F1 measure. In *Proceedings of Machine Learning and Knowledge Discovery in Databases*, pages 225–239, 2014.

[61] Zhifa Liu, Brandon Malone, and Changhe Yuan. Empirical evaluation of scoring functions for Bayesian network model selection. *BMC Bioinformatics*, 13(Suppl 15): S14, 2012.

[62] Yi Luo, Issam El Naqa, Daniel L. McShan, Dipankar Ray, Ines Lohse, Martha M. Matuszak, Dawn Owen, Shruti Jolly, Theodore S. Lawrence, Feng-Ming Kong, and Randall K. Ten Haken. Unraveling biophysical interactions of radiation pneumonitis in non-small-cell lung cancer via Bayesian network analysis. *Radiotherapy and Onconology*, 123:85–92, 2017.

[63] David Madigan and Adrian E. Raftery. Model selection and accounting for uncertainty in graphical models using Occam's window. *Journal of the Amercian Statistical Association*, 89:1535–1546, 1994.

[64] Richard J. McNally, Patrick Mair, Beth Reeder, and Bradley Riemann. Co-morbid obsessive-compulsive disorder and depression: A Bayesian network approach. *Psychological Medicine*, 47:1204–1214, 2017.

[65] Christopher Meek. Causal inference and causal explanation with background knowledge. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 403–410, 1995.

[66] Marina Meilă and Tommi Jaakkola. Tractable Bayesian learning of tree belief networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 380–388, 2000.

[67] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[68] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[69] Judea Pearl. *Causality*. Cambridge university press, 2009.

[70] Franz Pernkopf and Jeff A. Bilmes. Efficient heuristics for discriminative structure learning of Bayesian network classifiers. *Journal of Machine Learning Research*, 11 (Aug):2323–2360, 2010.

[71] Franz Pernkopf and Michael Wohlmayr. Stochastic margin-based structure learning of Bayesian network classifiers. *Pattern recognition*, 46(2):464–471, 2013.

[72] Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 689–690. IEEE, 2011.

[73] Cesar A. Prada-Medina, Kiyoshi F. Fukutani, Nathella Pavan Kumar, Leonardo Gil-Santana, Subash Babu, Flávio Lichtenstein, Kim West, Shanmugam Sivakumar, Pradeep A. Menon, Vijay Viswanathan, Bruno B. Andrade, Helder I. Nakaya, and Hardy Kornfeld. Systems immunology of diabetes-tuberculosis comorbidity reveals signatures of disease complications. *Scientific Reports*, 7:1–16, 2017.

[74] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[75] Brian D. Ripley. Pattern recognition and neural networks. *Cambridge University Press*, 1996.

[76] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[77] Jorma Rissanen and Teemu Roos. Conditional NML universal models. In *2007 Information Theory and Applications Workshop*, pages 337–341. IEEE, 2007.

[78] Mauro Scanagatta, Cassio P. de Campos, Giorgio Corani, and Marco Zaffalon. Learning Bayesian networks with thousands of variables. In *Advances in Neural Information Processing Systems*, pages 1864–1872, 2015.

[79] Mauro Scanagatta, Giorgio Corani, and Marco Zaffalon. Improved local search in Bayesian networks structure learning. In *Advanced Methodologies for Bayesian Networks*, pages 45–56, 2017.

[80] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6: 461–464, 1978.

[81] Marco Scutari. Learning Bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3):1–22, 2010.

[82] Marco Scutari and Radhakrishnan Nagarajan. Identifying significant edges in graphical models of molecular networks. *Artificial Intelligence in Medicine*, 57:207–217, 2013.

[83] Marco Scutari, Phil Howell, David J. Balding, and Ian Mackay. Multiple quantitative trait analysis using bayesian networks. *Genetics*, 198(1):129–137, 2014.

[84] Marco Scutari, Catharina Elisabeth Graafland, and José Manuel Gutiérrez. Who learns better Bayesian network structures: Accuracy and speed of structure learning algorithms. *International Journal of Approximate Reasoning*, 115:235–253, 2019.

[85] Basilio Sierra, Nicolas Serrano, Pedro Larrañaga, Eliseo Plasencia, Iñaki Inza, Juan Jiménez, Pedro Revuelta, and Melfy Mora. Using Bayesian networks in the construction of a bi-level multi-classifier. a case study using intensive care unit patients data. *Artificial Intelligence in Medicine*, 22:233–48, 07 2001.

[86] Tomi Silander and Petri Myllymäki. A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pages 445–452, 2006.

[87] Tomi Silander, Petri Kontkanen, and Petri Myllymäki. On sensitivity of the MAP Bayesian network structure to the equivalent sample size parameter. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, 2007.

[88] Tomi Silander, Teemu Roos, and Petri Myllymäki. Learning locally minimax optimal Bayesian networks. *International Journal of Approximate Reasoning*, 51(5):544–557, 2010.

[89] Tomi Silander, Janne Leppä-aho, Elias Jääsaari, and Teemu Roos. Quotient normalized maximum likelihood criterion for learning Bayesian network structures. In *Proceedings of the Twenty-First Conference on Artificial Intelligence and Statistics*, 2018.

[90] Peter Spirtes, Clark N. Glymour, Richard Scheines, and David Heckerman. *Causation, prediction, and search*. MIT press, 2000.

[91] M. Stone. An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion. *Journal of the Royal Statistical Society Series B*, 39:44–47, 1977.

[92] Joe Suzuki. A theoretical analysis of the BDeu scores in Bayesian network structure learning. *Behaviormetrika*, 44(1):97–116, 2017.

[93] Joe Suzuki and Jun Kawahara. Branch and bound for regular Bayesian network structure learning. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence*, 2017.

[94] Marc Teyssier and Daphne Koller. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 548–549, 2005.

[95] Jin Tian, Ru He, and Lavanya Ram. Bayesian model averaging using the k-best Bayesian network structures. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pages 589–597, 2010.

[96] Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65 (1):31–78, 2006.

[97] Tim Van Allen and Russell Greiner. Model selection criteria for learning belief nets: An empirical comparison. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1047–1054, 2000.

[98] Peter van Beek and Hella-Franziska Hoffmann. Machine learning of Bayesian networks using constraint programming. In *Proceedings of the 21st International Conference on Principles and Practice of Constraint Programming*, pages 428–444, 2015.

[99] Thomas Verma and Judea Pearl. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 220–227, 1990.

[100] Shulin Yang and Kuo-Chu Chang. Comparison of score metrics for Bayesian network learning. *IEEE Transactions on Systems, Man and Cybernetics*, 32:419–428, 2002.

[101] Nan Ye, Kian Ming A. Chai, Wee Sun Lee, and Hai Leong Chieu. Optimizing F-measures: A tale of two approaches. In *Proceedings of the International Conference on Machine Learning*, 2012.

[102] Changhe Yuan, Brandon Malone, and Xiaojian Wu. Learning optimal Bayesian networks using A* search. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pages 2186–2191, 2011.

[103] Ming-Jie Zhao, Narayanan Edakunni, Adam Pocock, and Gavin Brown. Beyond Fano's inequality: bounds on the optimal F-score, BER, and cost-sensitive risk and their implications. *Journal of Machine Learning Research*, 14:1033–1090, 2013.

[104] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Abbreviations

ϵBNSL ϵ — Bayesian network structure learning 20–24, 36

AIC Akaike information criterion 35–38, 41, 43, 45–48, 86

BDJ Bayesian Dirichlet score based on Jeffreys' prior 39

BDeu likelihood-equivalence Bayesian Dirichlet score with uniform priors 5, 8–10, 19, 21–24, 26, 28, 30–36, 39–41, 43–48, 58, 66, 72, 73, 86, 87

BF Bayes factor 19, 22–25, 27–33, 41, 58

BIC Bayesian information criterion 4, 8–10, 19–25, 27–30, 35–38, 41, 43, 45–47, 58, 86

BNSL Bayesian network structure learning 3, 5–11, 13, 18, 20–22, 31, 36, 37, 39–41, 47, 48, 55, 56, 58, 62, 86, 87

BN Bayesian Network 1–12, 14, 16, 18, 19, 24, 28, 31, 34–37, 41, 43–48, 55, 58, 59, 62, 65, 67, 68, 73, 86, 87

CPDAG completed partially directed acyclic graph 11–15, 70–73

DAG directed acyclic graph 1, 3, 4, 6–12, 18, 19, 21, 22, 28, 30, 32

MEC Markov equivalence class 11, 12, 27, 29, 30, 32

SHD structural Hamming distance 11–13, 36, 45–47, 63, 64, 67, 70–73, 86

fNML factorized normalized maximum likelihood 38

qBDJ quotient Bayesian Dirichlet score based on Jeffreys' prior 35, 36, 38, 39, 43, 46, 47, 86