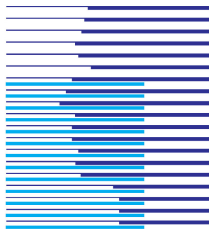


Bayesian Networks, Big Data and Greedy Search

Efficient Implementation with Classic Statistics

Marco Scutari
scutari@idsia.ch
April 3, 2019

USI/SUPSI



IDSIA

Istituto
Dalle Molle
di studi
sull'intelligenza
artificiale

Overview

- Learning the structure of Bayesian networks from data is known to be a **computationally challenging**, NP-hard problem [2, 4, 6].
- Greedy search is the most common score-based heuristic for structure learning, how challenging is it in terms of **computational complexity**?
 - For discrete data;
 - for continuous data;
 - for hybrid (discrete + continuous) data;
 - for big data ($n \gg N$ and/or $n \gg |\Theta|$).
- How are scores computed, and **can we do better** by revisiting learning
 - from **classic statistics**?
 - from a **machine learning** perspective?

Bayesian Networks and Structure Learning

Bayesian Networks: A Graph and a Probability Distribution

A Bayesian network [15, BN] is defined by:

- a **network structure**, a directed acyclic graph \mathcal{G} in which each node $v_i \in \mathbf{V}$ corresponds to a random variable X_i ;
- a **global probability distribution** $P(\mathbf{X})$ with parameters Θ , which can be factorised into smaller **local probability distributions** according to the arcs present in the graph.

The main role of the network structure is to express the **conditional independence** relationships among the variables in the model through **graphical separation**, thus specifying the factorisation of the global distribution:

$$P(\mathbf{X}) = \prod_{i=1}^p P(X_i \mid \Pi_{X_i}; \Theta_{X_i}) \quad \text{where} \quad \Pi_{X_i} = \{\text{parents of } X_i\}.$$

Common Distributional Assumptions

The three most common choices for $P(\mathbf{X})$ in the literature (by far), are:

- **Discrete** BNs [13], in which \mathbf{X} and the $X_i \mid \Pi_{X_i}$ are multinomial:

$$X_i \mid \Pi_{X_i} \sim \text{Mul}(\pi_{ik|j}), \quad \pi_{ik|j} = P(X_i = k \mid \Pi_{X_i} = j).$$

- **Gaussian** BNs [11, GBNs], in which \mathbf{X} is multivariate normal and the $X_i \mid \Pi_{X_i}$ are univariate normals linked by linear dependencies:

$$X_i \mid \Pi_{X_i} \sim N(\mu_{X_i} + \Pi_{X_i} \boldsymbol{\beta}_{X_i}, \sigma_{X_i}^2),$$

which can be equivalently written as a linear regression model

$$X_i = \mu_{X_i} + \Pi_{X_i} \boldsymbol{\beta}_{X_i} + \varepsilon_{X_i}, \quad \varepsilon_{X_i} \sim N(0, \sigma_{X_i}^2).$$

Common Distributional Assumptions

- Conditional linear Gaussian** BNs [17, CLGBNs], in which \mathbf{X} is a mixture of multivariate normals. Discrete $X_i \mid \Pi_{X_i}$ are multinomial and are only allowed to have discrete parents (denoted Δ_{X_i}). Continuous X_i are allowed to have both discrete and continuous parents (denoted Γ_{X_i} , $\Delta_{X_i} \cup \Gamma_{X_i} = \Pi_{X_i}$). Their local distributions are

$$X_i \mid \Pi_{X_i} \sim N \left(\mu_{X_i, \delta_{X_i}} + \Gamma_{X_i} \boldsymbol{\beta}_{X_i, \delta_{X_i}}, \sigma_{X_i, \delta_{X_i}}^2 \right),$$

which can be written as a mixture of linear regressions

$$X_i = \mu_{X_i, \delta_{X_i}} + \Gamma_{X_i} \boldsymbol{\beta}_{X_i, \delta_{X_i}} + \varepsilon_{X_i, \delta_{X_i}}, \quad \varepsilon_{X_i, \delta_{X_i}} \sim N \left(0, \sigma_{X_i, \delta_{X_i}}^2 \right),$$

against the continuous parents with one component for each configuration $\delta_{X_i} \in \text{Val}(\Delta_{X_i})$ of the discrete parents.

Other, less common options: copulas [9], truncated exponentials [18].

Bayesian Network Structure Learning

Learning a BN $\mathcal{B} = (\mathcal{G}, \Theta)$ from a data set \mathcal{D} is performed in two steps:

$$\underbrace{P(\mathcal{B} | \mathcal{D}) = P(\mathcal{G}, \Theta | \mathcal{D})}_{\text{learning}} = \underbrace{P(\mathcal{G} | \mathcal{D})}_{\text{structure learning}} \cdot \underbrace{P(\Theta | \mathcal{G}, \mathcal{D})}_{\text{parameter learning}} .$$

In a Bayesian setting **structure learning** consists in finding the DAG with the best $P(\mathcal{G} | \mathcal{D})$ (BIC [20] is a common alternative) with some search algorithm. We can decompose $P(\mathcal{G} | \mathcal{D})$ into

$$P(\mathcal{G} | \mathcal{D}) \propto P(\mathcal{G}) P(\mathcal{D} | \mathcal{G}) = P(\mathcal{G}) \int P(\mathcal{D} | \mathcal{G}, \Theta) P(\Theta | \mathcal{G}) d\Theta$$

where $P(\mathcal{G})$ is the **prior distribution over the space of the DAGs** and $P(\mathcal{D} | \mathcal{G})$ is the **marginal likelihood** of the data given \mathcal{G} averaged over all possible parameter sets Θ ; and then

$$P(\mathcal{D} | \mathcal{G}) = \prod_{i=1}^N \left[\int P(X_i | \Pi_{X_i}, \Theta_{X_i}) P(\Theta_{X_i} | \Pi_{X_i}) d\Theta_{X_i} \right] .$$

where Π_{X_i} are the parents of X_i in \mathcal{G} .

Structure Learning Algorithms

Structure learning algorithms fall into one three classes:

- **Constraint-based algorithms** identify conditional independence constraints with **statistical tests**, and link nodes that are not found to be independent. PC [7], HITON-PC [1].
- **Score-based algorithms** are applications of general optimisation techniques; each candidate network is assigned a **score** to maximise as the objective function. Heuristics [19], MCMC [16], exact [22]
- **Hybrid algorithms** have a *restrict* phase implementing a constraint-based strategy to reduce the space of candidate networks; and a *maximise* phase implementing a score-based strategy to find the optimal network in the restricted space. MMHC [23], H²PC [10].

Greedy Search is the Most Common Baseline

Here we concentrate on score-based algorithms and in particular **greedy search** because

- it is one of the **most common** algorithms in practical applications;
- when used in combination with **BIC**, it has the appeal of being simple to reason about;
- there is evidence **it performs well** compared to constraint-based and score-based algorithms [21].

We apply greedy search to modern data which can be

- with a **large sample size**, but not necessarily a large number of variables ($n \gg N$) or parameters ($n \gg |\Theta|$); and
- **heterogeneous**, with both discrete and continuous variables.

Computational Complexity of Greedy Search

Pseudocode for Greedy Search

Input: a data set \mathcal{D} , an initial DAG \mathcal{G} , a score function $\text{Score}(\mathcal{G}, \mathcal{D})$.

Output: the DAG \mathcal{G}_{max} that maximises $\text{Score}(\mathcal{G}, \mathcal{D})$.

1. Compute the score of \mathcal{G} , $S_{\mathcal{G}} = \text{Score}(\mathcal{G}, \mathcal{D})$.
2. Set $S_{max} = S_{\mathcal{G}}$ and $\mathcal{G}_{max} = \mathcal{G}$.
3. **Hill climbing:** repeat as long as S_{max} increases:
 - 3.1 for every valid arc addition, deletion or reversal in \mathcal{G}_{max} :
 - 3.1.1 compute the score of the modified DAG \mathcal{G}^* , $S_{\mathcal{G}^*} = \text{Score}(\mathcal{G}^*, \mathcal{D})$;
 - 3.1.2 if $S_{\mathcal{G}^*} > S_{max}$ and $S_{\mathcal{G}^*} > S_{\mathcal{G}}$, set $\mathcal{G} = \mathcal{G}^*$ and $S_{\mathcal{G}} = S_{\mathcal{G}^*}$.
 - 3.2 if $S_{\mathcal{G}} > S_{max}$, set $S_{max} = S_{\mathcal{G}}$ and $\mathcal{G}_{max} = \mathcal{G}$.
4. **Tabu search:** for up to t_0 times:
 - 4.1 repeat step 3 but choose the DAG \mathcal{G} with the highest $S_{\mathcal{G}}$ that has not been visited in the last t_1 steps regardless of S_{max} ;
 - 4.2 if $S_{\mathcal{G}} > S_{max}$, set $S_0 = S_{max} = S_{\mathcal{G}}$ and $\mathcal{G}_0 = \mathcal{G}_{max} = \mathcal{G}$ and restart the search from step 3.
5. **Random restart:** for up to r times, perturb \mathcal{G}_{max} with multiple arc additions, deletions and reversals to obtain a new DAG \mathcal{G}' and:
 - 5.1 set $S_0 = S_{max} = S_{\mathcal{G}}$ and $\mathcal{G}_0 = \mathcal{G}_{max} = \mathcal{G}$ and restart the search from step 3;
 - 5.2 if the new \mathcal{G}_{max} is the same as the previous \mathcal{G}_{max} , stop and return \mathcal{G}_{max} .

Computational Complexity

The following assumptions are standard in the literature:

1. Estimating each local distribution is $O(1)$; that is, the overall computational complexity of an algorithm is measured by the **number of estimated local distributions**.
2. Model comparisons are assumed to **always add, delete and reverse arcs correctly** with respect to the underlying true model, since marginal likelihoods and BIC are globally and locally consistent [3].
3. The **true DAG is sparse** and contains $O(cN)$, $c \in [1, 5]$ arcs.

The resulting expression for the computational complexity is:

$$\begin{aligned}
 O(g(N)) &= O\left(\underbrace{cN^3}_{\text{steps 1-3}} + \underbrace{t_0N^2}_{\text{step 4}} + \underbrace{r_0(r_1N^2 + t_0N^2)}_{\text{step 5}} \right) \\
 &= O\left(cN^3 + (t_0 + r_0(r_1 + t_0))N^2 \right).
 \end{aligned}$$

Caching Local Distributions

Caching local distributions **reduces the leading term to $O(cN^2)$** because

- Adding or removing an arc only alters a single $P(X_i | \Pi_{X_i})$.
- Reversing an arc $X_j \rightarrow X_i$ to $X_i \rightarrow X_j$ alters both $P(X_i | \Pi_{X_i})$ and $P(X_j | \Pi_{X_j})$.

Hence, we can keep a cache of the score values of the N local distributions for the current \mathcal{G}_{max} , and of the $N^2 - N$ differences

$$\Delta_{ij} = \text{Score}_{\mathcal{G}_{max}}(X_i, \Pi_{X_i}^{\mathcal{G}_{max}}, \mathcal{D}) - \text{Score}_{\mathcal{G}^*}(X_i, \Pi_{X_i}^{\mathcal{G}^*}, \mathcal{D}), i \neq j;$$

so that **we only have to estimate N or $2N$ local distributions** for the nodes whose parents changed in the previous iteration (instead of N^2).

Are They Really All the Same?

Estimating a local distribution in a **discrete BN** requires a **single pass** over the n samples for X_i and the Π_{X_i} (taken to have l levels each):

$$O(f_{\Pi_{X_i}}(X_i)) = O\left(\underbrace{n(1 + |\Pi_{X_i}|)}_{\text{counts}} + \underbrace{l^{1+|\Pi_{X_i}|}}_{\text{probabilities}}\right).$$

In a **GBN**, a local distribution is essentially a linear regression model and thus is usually estimated by applying a **QR decomposition** on $[1 \ \Pi_{X_i}]$:

$$O(f_{\Pi_{X_i}}(X_i)) = \underbrace{O(n(1 + |\Pi_{X_i}|)^2)}_{\text{QR decomposition}} + \underbrace{O(n(1 + |\Pi_{X_i}|))}_{\text{computing } \mathbf{Q}^T X_i} + \underbrace{O((1 + |\Pi_{X_i}|)^2)}_{\text{backwards substitution}} + \underbrace{O(n(1 + |\Pi_{X_i}|))}_{\text{computing } \hat{x}_i} + \underbrace{O(3n)}_{\text{computing } \hat{\sigma}_{X_i}^2}.$$

Are They Really All the Same?

In a **CLGBN**, the local distribution of a continuous node with Γ_{X_i} continuous parents and Δ_{X_i} discrete parents is a **mixture of linear regressions**, each estimated with QR:

$$\begin{aligned} O(f_{\Pi_{X_i}}(X_i)) &= \\ &= O\left((n + l^{|\Delta_{X_i}|})(1 + |\Gamma_{X_i}|)^2\right) + O(2n(1 + |\Gamma_{X_i}|)) + O(3n). \end{aligned}$$

The local distribution of a discrete node is computed in the same way as in a discrete BN.

It is clear that the computational complexity of estimating local distributions is very different under different distributional assumptions, so **the $O(1)$ assumption does not hold**. What does that mean for greedy search?

A Realistic Computational Complexity: Discrete BNs

Replacing $O(1)$ in the computational complexity of greedy search with that of the local distributions in **discrete BNs** we get

$$\begin{aligned}
 O(g(N, \mathbf{d})) &= \sum_{i=1}^N \sum_{j=1}^{|\Pi_{X_i}|+1} \sum_{k=1}^{N-1} O(n(1+j) + l^{1+j}) \\
 &= O\left(ncN^2 + nN \sum_{i=1}^N \frac{|\Pi_{X_i}|^2}{2} + Nl^2 \sum_{i=1}^N \frac{l^{|\Pi_{X_i}|+1} - 1}{l-1}\right).
 \end{aligned}$$

This implies that if \mathcal{G} is **sparse** ($|\Pi_{X_i}| \leq b$) complexity is $O(nN^2)$:

$$O(g(N, \mathbf{d})) = O\left(N^2 \left[nc + n\frac{b^2}{2} + l^2 \frac{l^{b+1} - 1}{l-1}\right]\right);$$

and $O(nN^2l^N)$ if \mathcal{G} is **dense** ($|\Pi_{X_i}| = O(N)$):

$$O(g(N, \mathbf{d})) = O\left(N^2 \left[nc + n\frac{N^3}{2} + l^2 \frac{l^N - 1}{l-1}\right]\right).$$

A Realistic Computational Complexity: GBNs and CLGBNs

The corresponding computational complexity in the case of **GBNs** is

$$O(g(N, \mathbf{d})) = O\left(nN \sum_{i=1}^N \frac{|\Pi_{X_i}|^3}{3}\right),$$

which is **polynomial even if \mathcal{G} is dense**. For **CLGBNs** with M continuous nodes and $N - M$ discrete nodes, we have a complicated expression that combines the previous two. It tells us that:

- $O(g(N, \mathbf{d}))$ is always **linear in the sample size**;
- unless the number of discrete parents is bounded for both discrete and continuous nodes, $O(g(N, \mathbf{d}))$ is again **more than exponential**;
- if the proportion of discrete nodes is small, we can assume that $M \approx N$ and $O(g(N, \mathbf{d}))$ is always **polynomial**.

Revisiting from Classic Statistics and Machine Learning

Greedy Search and Low-Order Regressions

If we assume that \mathcal{G} is sparse, most nodes will have a small number of parents and **the vast majority of the local distributions we estimate will be low-dimensional**. If we start the search from an empty DAG, we need to estimate local distributions

- with $j = 0, 1$ for all nodes;
- those with $j = 2$ for all non-root nodes;
- a **vanishingly small number with $j > 3$** .

Hence optimising how we estimate local distributions with $j = 0, 1, 2$ parents can have an important impact on the overall computational complexity; especially in the case of GBNs and CLGBNs which do not scale linearly in N .

Linear Regressions with Zero, One and Two Parents

- $j = 0$ gives the trivial regression $X_i = \mu_{X_i} + \varepsilon_{X_i}$.
- $j = 1$ gives the simple regression with:

$$\hat{\beta}_{X_j} = \frac{\text{COV}(X_i, X_j)}{\text{VAR}(X_j)}.$$

- $j = 2$ gives a regression with two explanatory variables and [25]

$$\hat{\beta}_{X_j} = \frac{1}{d} [\text{VAR}(X_k) \text{COV}(X_i, X_j) - \text{COV}(X_j, X_k) \text{COV}(X_i, X_k)],$$

$$\hat{\beta}_{X_k} = \frac{1}{d} [\text{VAR}(X_j) \text{COV}(X_i, X_k) - \text{COV}(X_j, X_k) \text{COV}(X_i, X_j)];$$

with $d = \text{VAR}(X_j) \text{VAR}(X_k) - \text{COV}(X_j, X_k)$.

In all cases we can compute closed-form estimators from variances and covariances, which are faster to compute (and to cache) than QR decompositions.

How Much Faster?

for GBNs:

j	with QR	closed-form
0	$O(6n)$	$O(4.5n)$
1	$O(9n)$	$O(7n)$
2	$O(16n)$	$O(10.5n)$

for CLGBNs:

j	with QR	closed-form
0	$O(6n + l^{D_{X_i}})$	$O(4.5n)$
1	$O(11n + 4l^{D_{X_i}})$	$O(7n)$
2	$O(18n + 9l^{D_{X_i}})$	$O(10.5n)$

Predictions as Scores, the Machine Learning Way

Chickering and Heckerman suggested [5] using **predictive posterior probability as the score function** to select the optimal DAG,

$$\text{Score}(\mathcal{G}, \mathcal{D}) = \log P(\mathcal{D}^{test} \mid \mathcal{G}, \Theta, \mathcal{D}^{train}), \quad \mathcal{D} = \mathcal{D}^{train} \cup \mathcal{D}^{test};$$

effectively maximising the negative cross-entropy between the “correct” posterior distribution of \mathcal{D}^{test} and that determined by $\mathcal{G}, \mathcal{D}^{train}$. This is called the **engineering criterion**.

As is the case for many machine learning models [12, e.g., deep neural networks], **prediction is computationally much cheaper than estimation** because it does not involve solving an optimisation problem.

Prediction vs Estimation

The computational complexity of prediction is:

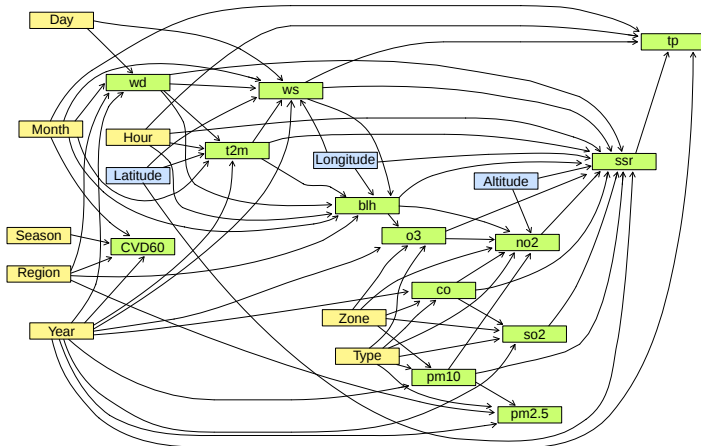
- $O(N|\mathcal{D}^{test}|)$ for discrete BNs, because we just have to perform an $O(1)$ look-up to collect the relevant conditional probability for each node and observation;
- $O(cN|\mathcal{D}^{test}|)$ for GBNs and CLGBNs, because for each node and observation we need to compute $\prod_{X_i}^{(n+1)} \hat{\beta}_{X_i}$.

In contrast **the computational complexity of estimating local distributions is higher than $O(N)$ for both GBN and CLGBNs**, while it is the same for discrete BNs.

Hence the proportion of \mathcal{D} used as \mathcal{D}^{test} will control the computational complexity of scoring nodes, since the per-node cost of prediction is smaller than that of estimation.

Can We Do Better?

Simulations from the MEHRA Network

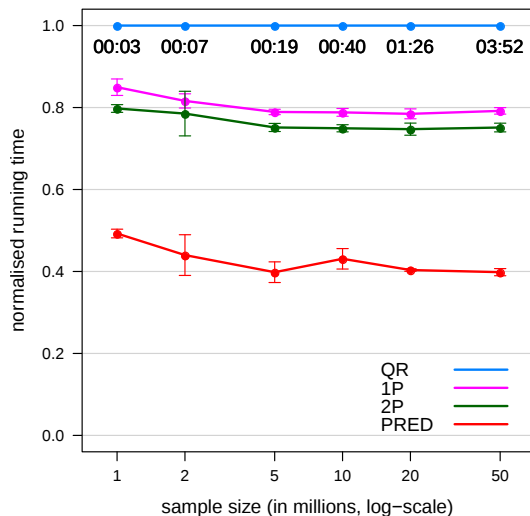


MEHRA [24]: 24 variables, 50 million observations to explore the interplay between environmental factors, exposure levels to outdoor air pollutants, and health outcomes in the English regions of the United Kingdom between 1981 and 2014.

Simulation Setting

1. We consider **sample sizes** of 1, 2, 5, 10, 20 and 50 millions;
2. For each sample size, we generate **5 data sets** from the CLGBN learned from the MEHRA data set;
3. For each sample, we learn back the structure of the BN using greedy search in combination with **various optimisations**:
 - **QR**: estimating all Gaussian and conditional linear Gaussian local distributions using the QR decomposition, and BIC as the score function;
 - **1P**: using the closed-form estimates for the local distributions that involve 0 or 1 parents, and BIC as the score function;
 - **2P**: using the closed-form estimates for the local distributions that involve 0, 1 or 2 parents, and BIC as the score functions;
 - **PRED**: using the closed-form estimates for the local distributions that involve 0, 1 or 2 parents for learning the local distributions on 75% of the data and estimating posterior predictive probabilities on the remaining 25%.

Running Times and Structural Errors



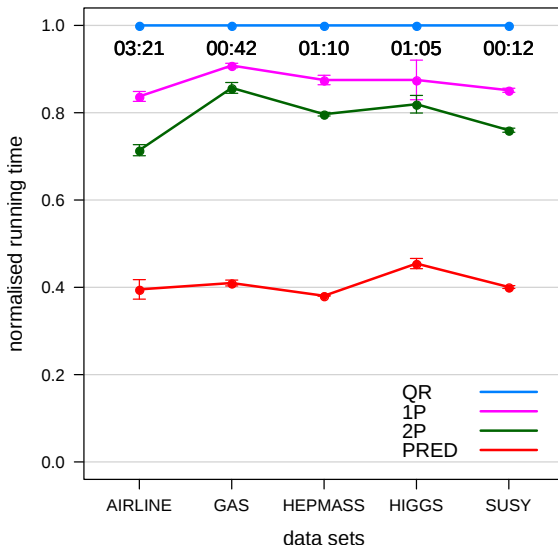
n	SHD	
	BIC	PRED
1	11	2
2	2	1
5	0	1
10	0	0
20	0	0
50	0	0

Simulations from Reference Data Sets

We confirm the improvements in running times on **5 reference data sets** from the UCI Machine Learning Repository [8] and from the repository of the Data Exposition Session of the Joint Statistical Meetings [14, JSM].

Data	sample size	discrete nodes	continuous nodes
AIRLINE	53.6×10^6	9	19
GAS	4.2×10^6	0	37
HEPMASS	10.5×10^6	1	28
HIGGS	11.0×10^6	1	28
SUSY	5.0×10^6	1	18

Running Times on the Reference Data Sets



Conclusions

Conclusions

- The assumption that estimating local distributions can be treated as an $O(1)$ operation, regardless of the number of parents and distributional assumptions, **is violated in practice**.
- The computational complexity of greedy search **is markedly different** for different distributional assumptions and graph sparsity.
- In light of this, we can revisit how we score nodes **using foundational results from classic statistics** and speed up learning of both GBNs and CLGBNs.
- And **taking a machine learning perspective** on scoring we can further speed up structure learning for all types of BNs by using predictive posterior probabilities as network scores.

Thanks!

References

References I

 C. F. Aliferis, A. Statnikov, I. Tsamardinos, S. Mani, and X. D. Xenofon.

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, 11:171–234, 2010.

 D. M. Chickering.

Learning Bayesian networks is NP-Complete.





In D. Fisher and H. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag, 1996.

 D. M. Chickering.




Optimal Structure Identification With Greedy Search.

Journal of Machine Learning Research, 3:507–554, 2002.





References II

-  D. M. Chickering and D. Heckerman.
Learning Bayesian networks is NP-hard.
Technical Report MSR-TR-94-17, Microsoft Corporation, 1994.
-  D. M. Chickering and D. Heckerman.
A Comparison of Scientific and Engineering Criteria for Bayesian Model Selection.
Statistics and computing, 10:55–62, 2000.
-  D. M. Chickering, D. Heckerman, and C. Meek.
Large-sample Learning of Bayesian Networks is NP-hard.
Journal of Machine Learning Research, 5:1287–1330, 2004.
-  D. Colombo and M. H. Maathuis.
Order-Independent Constraint-Based Causal Structure Learning.
Journal of Machine Learning Research, 15:3921–3962, 2014.


References III

-  D. Dheeru and E. Karra Taniskidou.
UCI Machine Learning Repository, 2017.
-  G. Elidan.
Copula Bayesian Networks.
In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 559–567, 2010.
-  M. Gasse, A. Aussem, and H. Elghazel.
A Hybrid Algorithm for Bayesian Network Structure Learning with Application to Multi-Label Learning.
Expert Systems with Applications, 41(15):6755–6772, 2014.


References IV

-  D. Geiger and D. Heckerman.
Learning Gaussian Networks.
In Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence, pages 235–243, 1994.
-  I. Goodfellow, Y. Bengio, and A. Courville.
Deep Learning.
MIT Press, 2016.
-  D. Heckerman, D. Geiger, and D. M. Chickering.
Learning Bayesian Networks: The Combination of Knowledge and Statistical Data.
Machine Learning, 20(3):197–243, 1995.
Available as Technical Report MSR-TR-94-09.
-  JSM, the Data Exposition Session.
Airline on-time performance, 2009.




References V

 D. Koller and N. Friedman.
Probabilistic Graphical Models: Principles and Techniques.
MIT Press, 2009.




 J. Kuipers and G. Moffa.
Partition MCMC for Inference on Acyclic Digraphs.
Journal of the American Statistical Association, 112(517):282–299,
2017.

 S. L. Lauritzen and N. Wermuth.
Graphical Models for Associations Between Variables, Some of
which are Qualitative and Some Quantitative.
The Annals of Statistics, 17(1):31–57, 1989.


References VI


-  S. Moral, R. Rumi, and A. Salmerón.
Mixtures of Truncated Exponentials in Hybrid Bayesian Networks.
In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, volume 2143 of *Lecture Notes in Computer Science*, pages 156–167. Springer, 2001.
-  S. J. Russell and P. Norvig.
Artificial Intelligence: A Modern Approach.
Prentice Hall, 3rd edition, 2009.
-  G. Schwarz.
Estimating the Dimension of a Model.
The Annals of Statistics, 6(2):461–464, 1978.

References VII

-  M. Scutari, C. E. Graafland, and J. M. Gutierrez.
Who Learns Better Bayesian Network Structures: Constraint-Based, Score-Based or Hybrid Algorithms?
Proceedings of Machine Learning Research (PGM 2018), 72:416–427, 2018.
-  J. Suzuki and J. Kawahara.
Branch and Bound for Regular Bayesian Network Structure Learning.
In Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence, pages 212–221, 2017.
-  I. Tsamardinos, L. E. Brown, and C. F. Aliferis.
The Max-Min Hill-Climbing Bayesian Network Structure Learning Algorithm.
Machine Learning, 65(1):31–78, 2006.

References VIII

-  C. Vitolo, M. Scutari, M. Ghalaieny, A. Tucker, and A. Russell.
Modelling Air Pollution, Climate and Health Data Using Bayesian
Networks: a Case Study of the English Regions.
Earth and Space Science, 5, 2018.
Submitted.

-  C. E. Weatherburn.
A First Course in Mathematical Statistics.
Cambridge University Press, 1961.