Informative Priors For Learning Graphical Models

E. Aljohani[1] **and J. Cussens** [2]

Abstract Graphical models represent relations of conditional independence between related variables, including, for example, those between various symptoms and causes of a disease. A hot topic in machine learning is the ability to learn such models from data. In some applications, it is crucial to include information not contained in data, prior information. The aim of this research is to provide a formalism to enable users to express prior knowledge in a flexible way such that they can express what they know about the problem efficiently for different problems. This involves creating a system in which the input is prior knowledge connected to a Bayesian learning algorithm. The main purpose is to design prior information ahead of time to speed learning and achieve learning that is more accurate.

1 Introduction

Graphical models represent the conditional independence relationships between related variables. A hot topic in machine learning is the ability to automatically learn graphical models from data [3]. This paper concerns learning graphical models, Bayesian networks specifically, using informative priors.

There are different types of prior knowledge, and different people have different types and levels of knowledge about a particular problem domain. However, because there are a wide variety of problems that need to be solved, the aim of this research is to develop an efficient way in which users can write down what they know about the problem. The main goal is to have an algorithm that takes prior knowledge as input data and builds a Bayesian network that is presumed to be the best available Bayesian network. Moreover, since different people have different prior knowledge, the research also attempts to be as flexible as possible. Thus, including prior knowledge attempts to make learning the Bayesian network much easier. Sometimes working out what the right prior knowledge should be is diffi-

1 University of York, YO10 5GE, UK

2 University of York, YO10 5GE, UK
james.cussens@york.ac.uk

cult, as the goal is to include all the information and somehow incorporate it prior to distribution. This paper discusses what sort of prior knowledge is going to be allowed and the Bayesian structure-learning algorithm will be used.

2 Learning algorithms that use prior knowledge

32.1 developing learning algorithm

The learning algorithm is intended to enable users to express their knowledge of a variety of problems in a straightforward manner. The main objective of this section is to investigate whether the algorithm for prior information developed here is compatible with the learning algorithm and indeed speeds learning. This section discusses the many experiments conducted during the development process in order to generate many results. In this experimental work, we aimed to develop/adapt learning algorithms using different datasets and types of prior knowledge. This experiment was divided into two stages, implementing first the developed learning algorithms and then developing prior knowledge.

In our experiments, we have used the datasets, which were made available by James Cussens at http://www.cs.york.ac.uk/aig/sw/gobnilp/data.

2.1.1 A hill climbing algorithm

Some Bayesian learning algorithms are heuristic search algorithms that try to maximize the score using some data. My current experimental work uses a hill-climbing algorithm, a heuristic search, which explores local moves that lead to the best Bayesian network score. Score-based approaches employ different scores and devote the most effort to maximizing scores without much consideration of prior information [5].

Representation of DAG The structure of a Bayesian networks is a directed acyclic graph (DAG) in which nodes represent the random variables, and edges represent the probabilistic dependence among variables [3]. As part of this experiment, it was necessary to design a data class; the simplest way to represent these classes on a graph was to state the parents of each variable. For example, variable A and its parents, variables B and C, are represented as A <- [B,C].

Cycle Checking As the DAG is the main graphical representation of a Bayesian network, the graphs have no cycles. Thus, no path starts and ends at the same node. Therefore, each time a hill-climbing algorithm makes a change, a method must be used to check for a cycle, which entails an additional move. The simplest approach is to update all child-parent sets [2].

BDe Score There are a number of ways of scoring; in this learning algorithm, the Bayesian Dirichlet likelihood equivalence (BDe) scoring method is used. In a BDe score, for each variable, we look to see what parents it has in the graph. The BDe score is the probability of the observed data conditional on the graph structure (marginal) [4].

Local Move The goal is to develop an algorithm that enables construction of a network with many possible parents. Any Bayesian network has one or more total orderings, and the crucial step is to create a Bayesian network from a collection of variables [2].

- The algorithm takes all the variables V = {x1, x2, . . . , xn} from the dataset and places them in an arbitrary order. First, variables need to be designated as a child and parents. Each child is selected by separating the variables in the ordering set from all the other variables V. The remainder of the variables, except for the child, are the possible parents and can be added as a parent in any move. The objective is to examine each variable and find the best possible parent sets.
- At the start, each child variable has a current and an empty parent set $A \leftarrow [\,]$, and for every iteration of the algorithm, the parent sets becomes larger. Although one variable is added at each move, more than one parent could result.
- The algorithm considered that adding one parent at a time could eventually lead to several parents. The number of possible parents that may be added in a single move is limited to a certain, adjustable number. Therefore, the algorithm needs to set a parameter to set that limit.
- The algorithm computes the local score of the possible parent sets for each child. Thus, the algorithm does not have to compute everything, merely each additional move.
- The algorithm considers all the possible sets and chooses the best one. However, if any given local score is better than the best score, the set is assigned to the best score.

Hill climbing with random restart

This algorithm does not ensure that it will find the optimal network because it may become stuck at a local maximum, which could be many. Moreover, at each step, the algorithm considers many different possibilities and, for each variable, adds parents for each additional move, up to a certain number. A run in hill climbing typically does not make improvements. In this situation, it is best to start the search again rather than continuing. Random-restart hill climbing is used to overcome becoming stuck on local maxima. This method randomly generates the initial state by conducting a series of hill climbing research. However, when using restarts, hill climbing is suitable to find a local optimum but is not guaranteed to identify the best possible solution in the search space [3]. This method iteratively performs hill climbing, each time with a random ordering **V**. The best total score of the ordering is kept; if a new iteration of hill climbing produces a better total score, that new score replaces the retained total score and returns the network with the high score.

3 Developing the file inputs of prior knowledge

This section incorporates different prior knowledge into the developed learning algorithms. It assesses which algorithms work well and the types of prior knowledge that should be used in different algorithms. There are many types of prior knowledge, such as the knowledge that a node **A** is not a parent of node **B** or that a node **A** is a parent of node **B**; known ordering; and, most challengingly and the main subject of this paper, ancestor relation and conditional independencies.

3.1 Prior knowledge

Prior information is any information that people have, other than data, that helps to obtain a good model. Hard information reduces the probability of some network structures to zero. In other words, to use prior information, one must take hard information and express it in the prior distribution to give certain networks zero probability. If the prior probability is zero, no matter what sort of data is available, the posterior probability will still be zero. In contrast, soft prior information gives nonzero probability to some possible network structures, and some will get higher probabilities than others will. When we set a prior probability to a small number along with enough supporting data, the posterior probability could be large. uniform priors knowledge can be used if we do not have any information, as it represents a lack of information. Such that, each structure has equal prior probability of being true [1].

3.1.1 Arrows do not to be there A/B

The user can input **"A/B"** to represent the prior knowledge that **A** must not be a parent of **B**. This hard constraint can easily be incorporated as prior knowledge by eliminating all violating parent sets from the datasets. Thus, this rules out that the parent set does not fit with this prior knowledge.

3.1.2 Arrows have to be there A->B or A<-B

Under this prior knowledge, an arrow can be added to a particular child based on the given parents. First, a child is picked and then which parents to add. Hill-climbing algorithms permit learning the network for many possible parents. The algorithm limits the number of parents that may be added. However, the limit of extra parents considered by the algorithm can be adjusted. For example, take **A<-B**, where **B** must be a parent of **A.** Assume that **A** is the specified child and that B is the specified parent. A constraint is needed to help obtain a network that meets this prior knowledge.
- If the current child equals the specified child, all possible parent sets that do contain the specified parent are chosen, and those that do not contain the specified parent are rejected. If the score is high, more parents that are possible are added.
- **A** cannot be the ancestor of **B**.

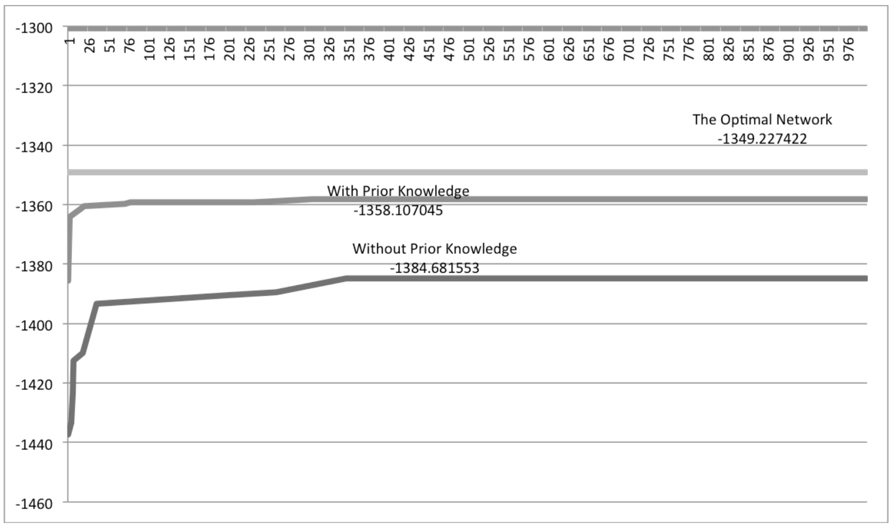

Figure 1 without and with prior knowledge

3.1.3 Ancestor relation

The algorithm works by considering a number of possibilities at various points. The greedy algorithm does not consider every possible network but makes and sticks with several choices. For each current child, the score for each possible parent is calculated, and the one with the highest score is chosen so that the network meets the ancestor relation constraint. The ancestor relation is difficult to check, because it might be found to happen at the last edge.

Intelligent Backtrack approach

Sometimes a network that does not meet a user's prior knowledge is generated. This network does not present a problem if the user's prior knowledge is simple enough to deal with edges. If the prior knowledge deals with an ancestor relation, however, an illegal network can go undetected until extremely late in the process, which creates problems. The primary challenge is to decide what action to take when a branch of the search fails or reaches dead ends. The intelligent backtrack approach can overcome this problem by returning to a related point that might fix the problem. Therefore, if the generated network does not meet the user's prior knowledge, the algorithm will perform intelligent backtracking. The simplest backtrack approach is to return to the most recent changes [2].

However, if the generated network does not meet the user's prior knowledge, the greedy algorithm backtracks to the ancestors of the specified user's child. If the algorithm generated an illegal network such as some missing variables, it backtracks to the beginning of the process. When the algorithm selects a random number as the ancestor of the specified child user's prior knowledge, all the variables after that number are re-computed. It also retains records of the most visited parent sets to avoid repetition later in the search. However, a limited portion of the most visited parent sets is retained.

Backtrack point If the produced network is inconsistent with prior knowledge, then the process backtracks to the most recent point of the ancestor of the specified child user's prior knowledge. If the child variable has no legal value to assign, this will produce an illegal network structure. Therefore, it backtracks to the beginning of the network structure, which is the first child variable for the perspective ordering **V**.

Restarts and Intelligent Backtrack approach

In this method, we employ a greedy search until we hit a local maximum. Then we randomly change the ordering of the variable of the network structure and repeat the process for some number of iterations. Additionally, using several random restarts may be a better strategy if the number of nodes is very large and the optimization is more likely to be complex. In contrast, intelligent backtracking often applies when a network structure is inconsistent with a constraint [2]. One of the means of intelligent backtracking is to backtrack to the child variable that might solve the problem through the most recent changes and then attempt to find different values. For any a single random restart, if the produced network is inconsistent with a user's prior knowledge then a backtrack method is applied.

For example, take **B**-**A**, where **A** must be an ancestor of **B**. Assume that **B** is the specified child and that **A** is the specified ancestor. If the generated network does not meet the prior knowledge, the algorithm will perform a backtracking by picking a random variable from the ancestor of the user's prior knowledge. If an illegal network is generated, the algorithm will perform a backtracking from the

beginning of the process. A constraint is needed to generate a network that meets this prior knowledge:
- Child **B** cannot have an empty parent.
- B cannot be ancestor of **A**.

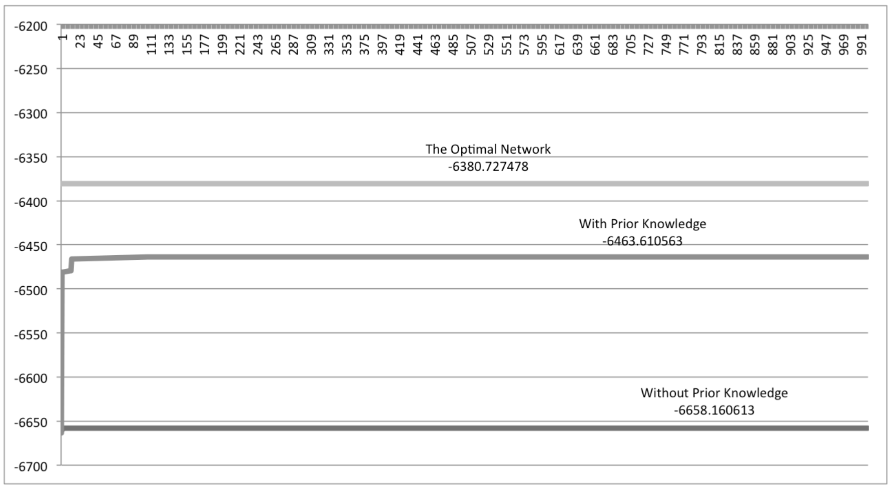

Figure 1 without and with prior knowledge

References

1. Friedman, Nir, Murphy, K., & Russell, S. (1998). Learning the Structure of Dynamic Probabilistic Networks, *in* `Proc. Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI '98)'.
2. Russell, S., Noving, P., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach* (third edit.). prentice hell.
3. Koller, Daphne, & Friedman, Nir. (2009). Probabilistic graphical models: principles and techniques.
4. Ueno, M. (2002). Robust learning Bayesian networks for prior belief.
5. Markowetz, F., & Spang, R. (2007). Inferring cellular networks--a review. BMC bioinformatics, 8 Suppl 6, S5. doi:10.1186/1471-2105-8-S6-S5.