Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE @ DIRECT®

Artificial Intelligence 143 (2003) 123–138

Artificial Intelligence

www.elsevier.com/locate/artint

Maximum likelihood bounded tree-width Markov networks [☆]

Nathan Srebro

Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Received 17 December 2001

Abstract

We study the problem of projecting a distribution onto (or finding a maximum likelihood distribution among) Markov networks of bounded tree-width. By casting it as the combinatorial optimization problem of finding a maximum weight hypertree, we prove that it is NP-hard to solve exactly and provide an approximation algorithm with a provable performance guarantee.
© 2002 Elsevier Science B.V. All rights reserved.

Keywords: Markov networks; Markov random fields; Undirected graphical models; Entropy decomposition; Hyper-trees; Tree-width; Hardness

1. Introduction

A fundamental problem of machine learning is that of estimating a joint distribution of many random variables from empirical data. One popular approach (the parametric approach) is to limit attention to a specific "concept class" of distribution models, and to find a good approximation for the empirical distribution within this model class. When the model class consists of distributions with a fixed dependency structure (i.e., when the structure of dependencies and independencies between variables is known in advance), closed form solutions can often be found. However when the dependency structure is not known in advance, and one seeks a simple structure that fits the data well, parametric density estimation remains a difficult tasks. A typical approach to this task involves heuristic search over model structure, usually without performance guarantees. While

[☆] This is an extended version of the paper presented at the 17th Conference on Uncertainty in Artificial Intelligence (UAI-01), Seattle, WA, 2001.

E-mail address: nati@mit.edu (N. Srebro).

model selection among arbitrary directed dependency networks using a Bayesian criterion is known to be hard [5], the hardness of many restricted learning scenarios is not well-understood. The potential for guaranteed approximation algorithms is also unresolved. Here, we focus on the simpler maximum likelihood criterion, in which regularization is attained solely by limiting the models to a concept class of bounded complexity. By casting the maximum likelihood model search problem in purely combinatorial terms, we obtain both hardness results and provable guarantees of approximation accuracy.

In 1968, Chow and Liu [6] provided a rigorous analysis for finding maximum likelihood Markov trees (distributions in which the dependencies follow a tree structure, with only pairwise direct dependencies and no loops), casting it as a problem of finding a maximum-weight tree, and thus providing an efficient and exact algorithm for it. Going beyond Markov trees, one can discuss Markov *networks* [13], in which the dependency structure follows a general undirected graph. We would like to generalize the work of Chow and Liu to the problem of learning a maximum likelihood Markov network of bounded complexity. In Section 2 we discuss how *tree-width* is in many ways the appropriate measure of complexity. Accordingly, we analyze the problem of learning a maximum likelihood Markov network of bounded tree-width.

In fact, we study a somewhat more general problem. The maximum likelihood distribution is a distribution minimizing the information divergence[1] from the empirical distribution. Finding a maximum likelihood distribution can thus be seen as a special case of the problem of *projecting* a target distribution onto a concept class, i.e., finding the distribution from within the class that minimizes the information divergence from the target. Such projections have applications beyond finding the maximum likelihood distribution. In this paper, we use this framework and discuss the problem of projecting a target distribution onto the class of Markov networks of bounded tree-width.

Similarly to the work of Chow and Liu, we are able to formalize the projection problem as a combinatorial optimization problem on graphs. We show that projecting a distribution onto Markov networks of bounded tree-width is equivalent to finding a maximum-weight *hypertree*. This equivalence gives rise to global, integer programming-based approximation algorithms with provable performance guarantees. This contrasts with previously suggested local-search heuristics for the same problem [11]. The equivalence also allows us to study the computational hardness of the learning problem. We show that learning a maximum likelihood Markov network of bounded tree-width is NP-hard, even for tree-width two.

Several other extensions to the work of Chow and Liu have been proposed. Meila [12] studied distributions which are mixtures of tree-shaped Markov networks. Dasgupta [8] suggested poly-tree Bayesian networks (trees with oriented edges), proving the hardness of this problem.

Important related work was carried out by Höffgen [9] who studied the PAC learnability of bounded tree-width Markov networks (as well as a more general family of concept classes). In this scenario, one wishes to approximately project an unknown distribution by sampling it. Höffgen's work primarily concerned the required sample size,

[1] Also known as Kullback–Leibler (KL) divergence and as relative entropy [7].

while the computational complexity (efficiency) of learning bounded tree-width Markov networks was not resolved. In studying the efficiency, Höffgen introduced a *directed* entropy decomposition that leads to a minimum-weight *directed* hypertree problem. However, since it is not known if a reverse reduction exists, the NP-hardness of this hypertree problem does not translate to a hardness result on the learning task. In this work, we introduce an *undirected* decomposition, which leads to a simpler (undirected) combinatorial optimization problem. The undirected decomposition is also reversible (i.e., the combinatorial problem can be translated to a learning problem), allowing us to establish hardness results on learning problems.

Details of the combinatorial optimization problem and the algorithm mentioned here can be found in a complementary paper [10].[2]

It should be noted that this work is concerned with finding distributions, and not with reconstructing the "true" structure. Maximum likelihood is generally not appropriate for model selection, as it will always find maximal structures, even when there is no support for some of the edges. In this sense, too, this work is an extension of Chow and Liu's approach.

2. Bounding the complexity of Markov networks

A Markov network over a specified graph G is determined by the marginal distributions over cliques in G, and this representation essentially gives the number of parameters of Markov networks over G. This number is exponential in the clique sizes, and so we would like to keep the clique sizes small.

But other than bounding the number of parameters, we would also like to limit ourselves to tractable computations. Although the clique marginals provide a compact representation for any Markov network with small clique size, there is no generally efficient way of performing exact computations (e.g., of marginal or conditional probabilities) on such graphs. In fact, even calculating the minimum information divergence to Markov networks over the graph (i.e., the maximum likelihood, if the target is the empirical distribution) might be infeasible. Though theoretically possible, it would be extremely optimistic to hope that finding the graph that minimizes this quantity would be easier than calculating it.

In order to work with Markov networks, and in particular to calculate marginal and conditional probabilities, one usually *triangulates* the graph, that is, adds edges to the graph so that all minimal cycles are 3-cycles. On a triangulated graph, such calculations can be performed in time linear in the representation of the clique marginals, i.e., exponential in the size of the cliques. But these are now the cliques of the augmented, triangulated graph. So it is not enough for the Markov network to have small cliques, in order for computations to be tractable, we need the Markov network to have a triangulation with small cliques.

This property is captured by the *tree-width*[3] of a graph:

[2] The algorithms are motivated by the problems discussed here, and some results from Section 3 are quoted in [10].

[3] See [15] for a review of several alternate characterizations of tree-width and related literature.

Definition 1 (*Tree-width*). The tree-width of a graph is the minimum, over all triangulations of the graph, of the maximum clique size in the triangulation, minus one:

$$\mathit{Tree\text{-}width}(G) = \min_{\substack{G' \supset G \\ G' \text{ is triangulated}}} \max_{h \in \mathit{Clique}(G)} |h| - 1.$$

In this work, we study the problem of projecting a distribution onto Markov networks over graphs of tree-width at most k, for some specified k. Graphs of tree-width one constitute the class of acyclic graphs (or forests), and so for $k = 1$ this becomes the problem of projecting onto Markov trees. As the width is increased, more complex Markov networks are allowed, with an exponential dependence on k.

As alluded to earlier, there are two (not necessarily overlapping) reasons for limiting the concept class: regularization (i.e., avoiding a large concept class that might lead to over-fitting) and tractability (i.e., ensuring that calculating with the model will be tractable).

Bounding the tree-width of permitted Markov networks address the tractability consideration almost exactly. Assuming we have other methods of ensuring regularization, or that we already know the distribution exactly, but would like to realize it in a model that would allow efficient computations, then limiting to bounded tree-width Markov networks is a natural thing to do. In fact, exact inference on Bayesian networks essentially involves working with a triangulated (moralized) Markov network. Therefore, the class of bounded tree-width Markov networks is the correct choice when seeking a class of graphical models, either directed or undirected, with the sole goal of efficiency of inference.

The regularization constraint is addressed less tightly, since the number of parameters in a Markov network is determined just by the size and number of cliques, and is not dependent on network being triangulated. It is interesting to note, though, that bounding the tree-width imposes a uniform constraint on the number of parameters, since not only the clique size is bounded, but also the number of cliques. All maximal networks of bounded tree-width k over n variables have exactly $n - k$ cliques of size $k + 1$ [15]. In fact, the number of cliques of any size is fixed, and consequently, the number of parameters is fixed (assuming a fixed number of states for all variables).

3. Decomposing the information divergence

Chow and Liu [6] showed that, for Markov trees, the reduction in information divergence, relative to the empty graph, can be additively decomposed to edges. The contribution of each edge of the tree is independent of the structure of the tree, and is the relative information between its nodes. We show a similar decomposition for "wider" triangulated networks. A key point of this decomposition is that the contributions of local elements are *independent* of the graph structure.

Recall that a Markov network can always be factored over its cliques [3, Hammersley–Clifford Theorem]. That is, any distribution P_X that is a Markov network over some graph G can be written as:

$$P_X(x) = \prod_{h \in \mathit{Clique}(G)} \phi_h(x_h) \tag{1}$$

where ϕ_h depends only on the outcome x_h inside the clique h.

In the general case, the clique factors ϕ_h might have a very complex dependence on the distribution. However, when G is triangulated, the factor can be chosen such that the factor of a clique depends only on the marginal distribution inside the clique. Moreover, the clique factors can be calculated explicitly and directly from the clique marginals.

We will concentrate on a specific explicit factorization, given by:

$$\phi_h(x_h) = \frac{P_h(x_h)}{\prod_{h' \subset h} \phi_{h'}(x_{h'})}, \qquad \phi_\emptyset(\cdot) = 1 \tag{2}$$

where the product in (1) is taken over all, not necessarily maximal, cliques. That is, we refer here to any complete subgraph of G as a *clique*. Factors corresponding to non-maximal cliques can of course be subsumed into some containing maximal clique factor. However, this leads to clique factors that are dependent on the graph structure. The factors given by (2) are unique in that a clique's factor does not depend on the graph G, except for the fact that G includes the clique.

More precisely: consider mappings $P_h \mapsto \phi_h$ from marginal distributions over subsets of variables, to factors over the subset. The mapping given in (2) is the only such mapping, such that (1) holds for *every* triangulated graph G and *every* Markov network P over G. A clique's factor depends *only* on the marginal inside the clique, and is completely oblivious to the distribution outside the clique, or even to the structure of the graph outside the clique. This very strong locality property will be essential later.

For a specific triangulated graph G, the projection of a target distribution $P^{\mathbf{T}}$ onto Markov networks over G can be calculated explicitly. Following (2), and since the projected Markov network is the one in which the clique marginals agree with $P^{\mathbf{T}}$ [16], the projection $\widehat{P}_G$ is given by:

$$\widehat{P}_G(x) = \prod_{h \in Clique(G)} \hat{\phi}_h(x_h), \quad \hat{\phi}_h(x_h) = \frac{{P^{\mathbf{T}}}_h(x_h)}{\prod_{h' \subset h} \hat{\phi}_{h'}(x_{h'})} \tag{3}$$

where again, the product is over all, not necessarily maximal, cliques.

What we would like to do is to project $P^{\mathbf{T}}$ onto Markov networks over *any* graph of tree-width at most k. The core problem is finding the *projected graph* itself (i.e., the graph over which the projected Markov network is achieved). We can associate with every graph G its information divergence from $P^{\mathbf{T}}$—that is, the minimum, over all Markov networks over G, of the information divergence from $P^{\mathbf{T}}$:

$$D(P^{\mathbf{T}} \| G) = \min_{\substack{P \text{ is a} \\ \text{Markov net over } G}} D(P^{\mathbf{T}} \| P) = D(P^{\mathbf{T}} \| \widehat{P}_G) \tag{4}$$

The projected graph is the bounded tree-width graph minimizing the information divergence from G.

Note that adding edges to a graph can only decrease the information divergence to it, since any distribution that was a Markov network on the sparser graph is certainly also a Markov network on the augmented one. So, the projected graph can always be taken to be triangulated—if it is not triangulated, it can be triangulated by adding edges without increasing the tree-width, yielding an acceptable triangulated graph with lower or equal divergence.

Taking this into account, it is enough to search over all triangulated graphs of tree-width at most k for the graph minimizing the information divergence from $P^{\mathbf{T}}$. But for triangulated graphs, the projected distribution $\widehat{P}_G$ is given by (3), and so the information divergence can be calculated as:

$$\begin{aligned} D\big(P^{\mathbf{T}}\|G\big) &= D\big(P^{\mathbf{T}}\|\widehat{P}_G\big) \\ &= \mathbf{E}_{X\sim P^{\mathbf{T}}}\left[\log \frac{P^{\mathbf{T}}(X)}{\prod_{h\in Clique(G)}\hat{\phi}_h(X_h)}\right] \\ &= \mathbf{E}_{P^{\mathbf{T}}}\big[\log P^{\mathbf{T}}\big] - \sum_{h\in Clique(G)} \mathbf{E}_{X\sim P^{\mathbf{T}}}\big[\log \hat{\phi}_h(X_h)\big]. \end{aligned}$$

Recall the strong locality of the projected clique factors $\hat{\phi}_h$, i.e., that they depend only on the marginals of $P^{\mathbf{T}}$. Consequently, for each candidate clique h, the term $\mathbf{E}_{P^{\mathbf{T}}}[\log \hat{\phi}_h(X_h)]$ in the sum depends only on the marginal inside the clique, and not on the structure of the graph G.

We can now describe a weight function over candidate cliques, such that $w(h) = \mathbf{E}_{P^{\mathbf{T}}}[\log \hat{\phi}_h(X_h)]$. This weight function can be calculated from the target distribution $P^{\mathbf{T}}$ alone, and is independent of the graph. After calculating the weight function for every possible clique, the information divergence to any triangulated graph is given by:

$$D\big(P^{\mathbf{T}}\|G\big) = \mathbf{E}_{P^{\mathbf{T}}}\big[\log P^{\mathbf{T}}\big] - \sum_{h\in Clique(G)} w(h) \tag{5}$$

where again, the sum is over all, not necessarily maximal, cliques in G. In fact, the summation always includes all singleton cliques, i.e., sets of a single vertex. For an empty graph $G = \emptyset$, the only cliques are singleton cliques, and so the information divergence to an empty graph (i.e., fully independent model) is exactly captured by the singleton cliques. Thus, separating out the singletons, we can rewrite (5) as:

$$D\big(P^{\mathbf{T}}\|G\big) = D\big(P^{\mathbf{T}}\|\emptyset\big) - \sum_{h\in Clique(G),\ |h|>1} w(h). \tag{6}$$

Eq. (6) expresses the reduction in the information divergence versus a simple base model, as a sum of weights (derived from the target distribution) of all non-trivial cliques that appear in the graph. Minimizing the information divergence is thus equivalent to maximizing this sum of weights.

This is illustrated in Fig. 1. The figure represents the entropy of the target distribution, and the entropies of its projection onto a specific triangulated graph G, and onto the empty graph $\emptyset$ (i.e., onto the set of fully independent distributions). Note that the clique marginals of the projection $\widehat{P}_G$ agree with $P^{\mathbf{T}}$, and so

$$\begin{aligned} \mathbf{E}_{P^{\mathbf{T}}}\big[\log \widehat{P}_G\big] &= -\sum_h \mathbf{E}_{P^{\mathbf{T}}}\big[\log \hat{\phi}_h\big] = -\sum_h \mathbf{E}_{\widehat{P}_G}\big[\log \hat{\phi}_h\big] \\ &= -\mathbf{E}_{\widehat{P}_G}\big[\log \widehat{P}_G\big] = -H\big(\widehat{P}_G\big). \end{aligned}$$

Similarly $\mathbf{E}_{P^{\mathbf{T}}}[\log \widehat{P}_G] = H(\widehat{P}_\emptyset)$, and we have $D(P^{\mathbf{T}}\|G) = H(\widehat{P}_G) - H(P^{\mathbf{T}})$ and $D(P^{\mathbf{T}}\|\emptyset) = H(\widehat{P}_\emptyset) - H(P^{\mathbf{T}})$.

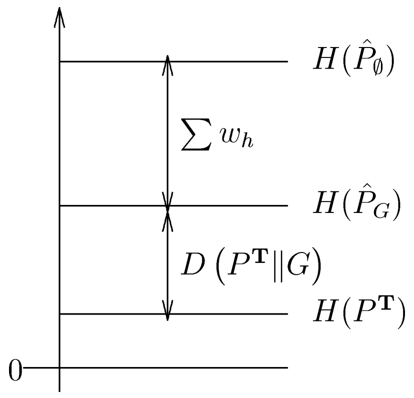

Fig. 1. $D(P^{\mathbf{T}}\|G) = (H(\widehat{P_\emptyset}) - H(P^{\mathbf{T}})) - \sum_{h \in Clique(G),\ |h|>1} w_h$.

Note that this weight decomposition differs from that of Höffgen [9] in two ways. First, Höffgen's weights sum to $H(\widehat{P_G})$, and so are actually complimentary to the weights discussed here (the summed Höffgen weight of a graph is equal to $H(\widehat{P_\emptyset})$ minus the summed weight of the graph as discussed here). More importantly, Höffgen's weights are not on cliques, but rather on directed, or ordered, sets of vertices, and apply to directed (acyclic) graphs rather then undirected graphs. In order to calculate the Höffgen weight of an undirected graph, it must be oriented (to form a DAG). Although the summed Höffgen weights of all orientations of a triangulated graph are equal, the orientation is an additional, non-local, structure that must be globally "agreed upon". Thus, the Höffgen weight cannot be computed locally for an undirected graph.

Before we return to maximizing the sum of weights, let us investigate the structure of these weights.

4. The weights

The weight of each candidate clique is determined by the target distribution and was defined in terms of the projected factor:

$$w(h) = \mathbf{E}_{X \sim P^{\mathbf{T}}}\left[\log \hat{\phi}_h(X_h)\right]. \tag{7}$$

Incorporating the explicit definition of the projected factors (3):

$$\begin{aligned}
&= \mathbf{E}_{P^{\mathbf{T}}}\left[\log \frac{P^{\mathbf{T}}{}_h(X_h)}{\prod_{h' \subset h} \hat{\phi}_{h'}(X_{h'})}\right] \\
&= \mathbf{E}_{P^{\mathbf{T}}}\left[\log P^{\mathbf{T}}{}_h(X_h)\right] - \sum_{h' \subset h} \mathbf{E}_{P^{\mathbf{T}}}\left[\log \hat{\phi}_{h'}(X_{h'})\right] \\
&= -H\left(P^{\mathbf{T}}{}_h(X_h)\right) - \sum_{h' \subset h} w(h')
\end{aligned} \tag{8}$$

where $H(P^{\mathbf{T}}(X_h))$ is the entropy of the marginal distribution of $P^{\mathbf{T}}$ over h. This provides for a simple recursive specification of the weights. Unrolling this recursion, the weight of a candidate hyperedge can also be written as a sum:

$$w(h) = -\sum_{h' \subseteq h} (-1)^{|h|-|h'|} H\big(P^{\mathbf{T}}(X_{h'})\big). \tag{9}$$

Note that the weight $w(\{v\})$ of a singleton $\{v\}$ is the negative entropy $-H(X_v)$ of the single variable. Accordingly, all singleton weights are negative (or at least non-positive). This is not surprising, as these weights sum up to $\mathbf{E}_{P^{\mathbf{T}}}[\log \widehat{P_\emptyset}] \leqslant 0$ and are incorporated in our formulation (6) of the projection problem as part of $D(P^{\mathbf{T}} \| \emptyset)$.

However, as more edges are added to the graph, the admissible distributions are less limited and the projected distribution can become more similar to the target distribution. This means that weights of candidate cliques beyond singletons should generally have a positive contribution, representing the reduction in the information divergence.

In fact, the weight of a vertex pair $\{u, v\}$, i.e., an edge, is (following (9)): $w(\{u, v\}) = -H(X_u, X_v) + H(X_u) + H(X_v) = I(X_u; X_v) \geqslant 0$, in agreement with Chow and Liu. The mutual information across an edge precisely captures the reduction in information divergence attained by taking into account the dependence between the endpoints.

Now consider the weight of some candidate three-clique $\{1, 2, 3\}$. If the three variables X_1, X_2, X_3 are pairwise independent, but have some three-way dependence, then using (9), we can calculate $w(1, 2, 3) = H(X_1) + H(X_2) + H(X_3) - H(X_1, X_2, X_3)$. The weight is again non-negative. It correctly captures the benefit of taking into account the dependency between the variables, as quantified by the reduction in the information divergence. In fact, the weight is equal to the information divergence between the true three-way marginal distribution, and the product distribution of the three singleton marginal distributions.

It is tempting to adopt this clean interpretation of the weights, by which the weight of a candidate clique represents the reduction in information divergence attained by taking into account the additional dependency. Under this interpretation, the weight of a candidate d-clique should be the information divergence between the true d-way marginal and the maximum entropy d-way distribution that agrees with the marginals of all $d - 1$ sub-cliques, similar to the hierarchical decomposition suggested by Amari [1].

The reality is different. Consider a Markov chain over three variables $X_1 \to X_2 \to X_3$. In this case all the information is in the pairwise dependencies, and by the above suggested interpretation, the weight of the candidate three-clique $\{1, 2, 3\}$ should have been zero. Using (9), however, we can check that $w(1, 2, 3) = -I(X_1; X_3) < 0$. On second thought, this should not surprise us. Consider the total weight of a graph containing the three-clique. All the dependencies in the Markov chain are already captured by the two pairwise dependencies (X_1, X_2) and (X_2, X_3). Accordingly, all of the reduction in the information divergence is captured by $w(\{1, 2\}) + w(\{2, 3\})$. However, since we sum all the cliques in the graph, in addition to these two pairwise weights, we will also include $w(\{1, 3\}) = I(X_1; X_3)$. The pairwise weights thus over-count the reduction in the information divergence. The weight of the candidate three-clique, which will always be included if all three pairwise weights are included, accounts for this over-counting.

The weights of candidate cliques thus serve not only to reward for the new dependency made possible by the clique, but also to account for over-counting inside the clique.

Pairwise weights need not perform any such accounting, but the weight of any larger candidate clique can be either positive or negative.

The possibility of negative weights is very problematic from an algorithmic standpoint. Negative weights may cause many combinatorial optimization approaches to fail. In fact, the algorithms presented in [10] for the maximum weight hypertree problem do not work with arbitrary negative weights. Fortunately, the weights we define here do have some positive structure.

Although for any particular candidate clique of more than two vertices, the clique weight can be negative, it is not possible for too many weights to be negative. As argued before, adding edges to a graph *does* reduce the information divergence, but this might be accounted for by more then just one additional clique. The total weight of a triangulated graph G is no less than the total weight of a triangulated sub-graph G' of G. Accordingly, the sum of the weights of cliques in G that are not cliques in G' must be non-negative.

We call a weight function obeying such a constraint a *monotone* weight function. It is enough to require that the weight function be monotone on cliques, i.e., that the total summed weight of a graph containing a single clique $\{v_1, v_2, \ldots, v_d\}$ is no less than the total summed weight of the graph containing the single clique $\{v_1, v_2, \ldots, v_{d-1}\}$. A weight function that is monotone on cliques is also monotone on all triangulated graphs.

Note that these arguments hold only for triangulated graphs. Otherwise the total summed weight of the graph does not represent any meaningful information quantity as the product of the factors may not be a valid distribution function, let alone the projected distribution.

5. The reduction

The problem of projecting a distribution onto Markov networks of bounded tree-width k can thus be reduced to finding a triangulated graph of bounded tree-width (or equivalently, clique size) that maximizes the total summed weight of its (not only maximal) cliques with respect to some monotone weight function. That is, if we knew an algorithm that finds such a graph, we could use it for projecting distributions onto Markov networks of bounded tree-width.

To do so, we would first calculate a weight for each set of at most $k+1$ vertices (including singletons), starting with the small sets and proceeding to the larger ones, using the recurrence (8). We would then find the maximum weight bounded tree-width triangulated graph for these weights, but ignoring the singleton weights (the singleton weights are necessary only for the recursive calculation).

For tree-width one, i.e., triangulated graphs with no three-cliques, this is the problem of finding a maximum weight tree, and as we saw before, the weights are in agreement with Chow and Liu. For higher tree-width, this is the problem of finding a maximum weight *hypertree* [10].

Although the recursive definition provides for a relatively quick method of calculating the weights for small k, it is still necessary to calculate $\sum_{d=2}^{k+1}\binom{n}{d} = \mathrm{O}(n^{k+1})$ weights, and calculating each weight involves summing $\mathrm{O}(2^{k+1})$ subclique weights, for a total time complexity of $\mathrm{O}((2n)^{k+1})$.

As we have not considered the representation of the target distribution, we cannot discuss the complexity of the reduction in terms of the problem 'size', as this of course depends on the representation. We do not want to go into the issues of input representations of the distribution, except for one special case which originally motivated us: the case in which the distribution is an empirical distribution of some sample.

The "input representation" in this case is the sample itself, of size $\Theta(Tn\log m)$, where T is the sample size and m is the number of possible outcomes for each random variable. So, if k is part of the input, the reduction is *not* polynomial in the sample, as it is exponential in k while the sample is independent of it. If k is constant, then the reduction is polynomial.

As the number of parameters in the resulting model, and therefore the complexity of calculations on the resulting distribution, is also exponential in k, it is tempting to hope that the reduction is comparable to, or at least polynomial in, the resulting number of parameters. This is essentially the output size of the learning problem, and practically also a bound on the input size, as one would generally not have less data then there are parameters to learn. However, this is *not* the case. The number of parameters is only $\mathrm{O}(nm^{k+1})$. Therefore if $n \gg m$, the reduction is super-polynomial even in the resulting number of parameters.

For a small, fixed, target width k, the reduction is polynomial, and the number of weights to be calculated might not be prohibitive. For example, even for a target width of $k = 2$, this is already a useful generalization of Chow and Liu.

6. The reverse reduction

In order to use the formulation (6) to analyze the computational hardness, we must show how to perform the reverse reduction, i.e. transform an input of the maximum hypertree problem (a weight function) to an input of the projection problem (a distribution) so that the projected distribution implies the maximum hypertree. In this section, we show that for every non-negative weight function on vertex sets of fixed size, there exists a distribution that yields weights proportional to this set of weights. We thus demonstrate that the problem of finding a maximum hypertree, at least for a non-negative weight function on vertex sets of a fixed size, can be reduced to projecting a distribution onto Markov networks of bounded tree-width.

Furthermore, a "small" sample can be constructed, with an empirical distribution yielding weights that are close enough to these weights, conserving the exact structure of the projected graph. This establishes the following theorem (proved by Sections 6.1 and 6.2):

Theorem 1 (The Reverse Reduction). *The problem "given n vertices, and a non-negative integer weight $w(h)$ for each vertex set h of size exactly $k+1$, find the maximum weight k-hypertree" can be reduced, in time polynomial in the output, to the problem "given m samples of n binary random variables, find a maximum likelihood Markov network of tree-width at most k", with sample size $m = poly(n^{k+1}\max_h w(h))$.*

This reduction is weak, in the sense that the sample size needed to produce specific weights is polynomial in the *value* of the weights (and so exponential in the size of their representation). Still, this pseudo-polynomial reduction is enough to show NP-hardness of finding a maximum likelihood Markov network of bounded tree-width, even for tree-width two.

6.1. A distribution yielding desired weights

We start by constructing a distribution that yields weights proportional to a desired target weight function w (we assume without loss of generality that $\forall_h w(h) < 1$). We consider only (non-negative) weight functions on vertex sets of size *exactly* $k+1$, and construct distributions P_w yielding weights w' that are proportional to the desired weights w on vertex sets of size $k+1$, and zero on smaller sets. In order to be achieve this, we consider each vertex as a binary variable, and construct a distribution that is *almost* uniform over them, but introduce biases over sets of $k+1$ variables. Marginals over smaller sets remain uniform, yielding the required zero weights. It is not, in general, possible to achieve arbitrary biases on all sets of $k+1$ variables concurrently, as large biases can "interfere" with each other. Therefore, the biases must be very small, resulting in very small weights, but still proportional to the desired set of weights.

The distribution P_w will be a uniform mixture of $\binom{n}{k+1}$ distributions P_w^h, one for each $h \in \binom{V}{k+1}$. Each such P_w^h will deviate from uniformity only by a bias of $r(h)$ in the parity of the variables in h. We show below how to select $r(h)$ according to $w(h)$. Explicitly:

$$P_w^h(x) = \begin{cases} \frac{1+r(h)}{2^{|V|}} & \text{if } \sum_{v\in h} x_v \text{ is odd,} \\ \frac{1-r(h)}{2^{|V|}} & \text{if } \sum_{v\in h} x_v \text{ is even.} \end{cases} \tag{10}$$

This results in a mixed distribution P_w in which all marginals over at most k variables are uniform (and therefore have zero corresponding weight), while the marginal over a set h of exactly $k+1$ variables has a bias of $b = r(h)/\binom{n}{k+1}$. The corresponding weight is therefore

$$\begin{aligned} w'(h) &= -H(X_h) - \sum_{h'\subset h} w(h) = \sum_{v\in h} H(X_v) - H(X_h) \\ &= |h| \times 1 - H(X_h) = (k+1) + \sum_{x_h} P_w(x_h)\log P_w(x_h) \\ &= (k+1) + 2^k \frac{1+b(h)}{2^{k+1}} \log\frac{1+b(h)}{2^{k+1}} + 2^k \frac{1-b(h)}{2^{k+1}} \log\frac{1-b(h)}{2^{k+1}} \\ &= \frac{1}{2}\big(\big(1+b(h)\big)\log\big(1+b(h)\big) + \big(1-b(h)\big)\log\big(1-b(h)\big)\big). \end{aligned} \tag{11}$$

Using the natural logarithm and taking the Taylor expansion:

$$= \sum_{i=2 \text{ even}}^{\infty} \frac{b(h)^i}{i(i-1)} = \frac{b(h)^2}{2} + \mathrm{O}\big(b(h)^4\big) = \frac{r(h)^2}{2\binom{n}{k+1}^2} + \mathrm{O}\big(r(h)^4\big). \tag{12}$$

Choosing $r(h)$ to be approximately $\sqrt{w(h)}$ (or more precisely, the inverse function of (11)) yields weights proportional to w.

6.2. A sample yielding desired weights

We have shown a distribution that produces weights proportional to any desired non-negative weight function. But since the biases in this distribution might be irrational (being the inverse of (11)), there is no finite sample that has such a distribution as its empirical distribution.

We will show a finite sample that results in weights that are close enough to the desired weights, such that the optimal structure is conserved. Given a rational weight function w, we will show a sample with empirical distribution $\widehat{P}_w$ that produces weights $w''(h) = w'(h) + e(h)$ such that w' are proportional to w, and $\sum_h |e(h)| < 1/Q_{w'}$ where $Q_{w'}$ is the common denominator of w'. This is enough, since the total summed w' and w'' weights of cliques in the optimal graph will be within $1/Q_{w'}$, less than the possible difference due to taking cliques with differing weights.

We first show how to construct a sample that yields an empirical distribution similar in structure to P_w, with rational biases on $k+1$ candidate edges. For any mapping[4] $h \mapsto p_h/Q < 1$ we construct a sample $\mathcal{S}_{p/Q}$ with empirical distribution $\widehat{P}_{p/Q}$ such that all k-marginals are uniform, and for $|h| = k+1$:

$$\widehat{P}_{p/Q}(x_h) = \begin{cases} \left(1 + \frac{p_h}{Q\binom{n}{k+1}}\right) 2^{-|V|} & \text{if } \sum_{v \in h} x_v \text{ is odd,} \\ \left(1 - \frac{p_h}{Q\binom{n}{k+1}}\right) 2^{-|V|} & \text{if } \sum_{v \in h} x_v \text{ is even.} \end{cases}$$

Unlike the exact P_w, parities of larger sets might be very biased. However, these do not effect the resulting weights when searching for width-k Markov networks.

We will build the sample as a pooling of $\binom{n}{k+1}$ equisized samples $\mathcal{S}^h_{p/Q}$, one for each candidate edge of size $k+1$. Each such $\mathcal{S}^h_{p/Q}$ will be constructed from Q equisized blocks of $(k+1)$-wise uniformly independent sample vectors. But for p of these blocks, we will invert the elements of h appropriately so as to set the parity of x^t_h to be odd for all sample vectors in the block. Note that this can be done without disrupting the uniformity of any other set of vertices of size at most $k+1$. The resulting $\mathcal{S}^h_{p/Q}$ will be uniform on all subsets of size up to $k+1$, except for a bias of $p(h)/Q$ on h. Pooling these together yields the desired empirical distribution.

Using [2], $(k+1)$-wise independent blocks can be created of size $2n^{k+1}$, yielding a total sample size of $\binom{n}{k+1} Q 2n^{k+1} = \mathrm{O}(Qn^{2k+2})$, where Q is the common denominator of the rational weights.

We now know how to construct a sample with specified *rational* biases. However, the biases corresponding to rational weights are not rational. We first show how to achieve approximate weights with biases that are square roots of rationals, and then describe how these can be approximated with actual rationals.

We saw in (12) that the biases of the mixture components should be approximately the square roots of the desired weights. Using biases $r'(h) = \sqrt{w(h)}$ yields the following weights (where $b(h)' = r'(h)/\binom{n}{k+1} < 1$):

[4] The common denominator Q of the biases may be different than the common denominator $Q_{w'}$.

$$
\begin{aligned}
w''(h) &= \sum_{i=2 \text{ even}}^{\infty} \frac{b(h)'^i}{i(i-1)} < \frac{b(h)'^2}{2} + \sum_{i=4 \text{ even}}^{\infty} \frac{b(h)'^4}{i(i-1)} \\
&= \frac{b(h)'^2}{2} + \frac{\ln 4 - 1}{2} b(h)'^4 \\
&= \frac{1}{2\binom{n}{k+1}^2} w(h) + e(h),
\end{aligned}
$$

where

$$\sum_h |e(h)| < \binom{n}{k+1} \frac{\ln 4 - 1}{2} \frac{(\max w)^2}{\binom{n}{k+1}^4} < \frac{0.19}{\binom{n}{k+1}^3} \max w^2.$$

Recall that we would like $\sum_h |e(h)| < 1/Q_{w'}$. Since the common denominator $Q_{w'}$ scales linearly with the weights, we can achieve this goal by scaling the weights down. But since the weights may not be square rationals, taking their square root might produce irrational weights. This can be overcome in a similar fashion, by using a rational approximation to the square root.

This establishes Theorem 1, where the integer weights are scaled to rationals $w(h)/\max_{h'} w(h') \leqslant 1$.

6.3. The reduction and hardness

We saw how to reduce the maximum hypertree problem to the maximum likelihood Markov network problem, with the same k, and even if the variables are all binary. Note that our reduction is only pseudo-polynomial, as the sample size needed is polynomial in the value of the weights. However, in [15] we show that the maximum hypertree problem is NP-hard, even with zero/one weights:

Theorem 2 (Proved in [15]). *The maximum hypertree problem is* NP-*hard, even for width two, zero/one weights, and weights only on pairs of vertices. Furthermore, it is* NP-*hard to approximate to within any constant additive offset.*

This is enough to show NP-hardness of the maximum likelihood Markov network problem, even for bound tree-width two:

Corollary 3. *Finding the maximum likelihood Markov network of tree-width at most k, for an explicitly given sample, is* NP-*hard, even for $k = 2$ and if all the variables are binary. Furthermore, it is* NP-*hard to approximate the log-likelihood, or the information divergence from the empirical distribution, to within any constant additive offset.*

Note that approximating the log-likelihood to within an additive offset is equivalent to approximating the likelihood to within a multiplicative factor. Hence, it is NP-hard to approximate the likelihood to within a multiplicative factor.

7. Approximation algorithms

Although the maximum hypertree problem is NP-hard, an integer programming based approximation algorithm can be devised [10]. For any constant k, a polynomial-time algorithm is given that finds a triangulated graph of tree-width at most k, which has a total summed weight within a constant factor of the maximum possible total summed weight. Unfortunately, this constant approximation factor depends heavily on k—for width k, we find a graph with total summed weight at least $1/(8^k k!(k+1)!)$ of the optimal. Algorithms with better approximation ratios may be possible, perhaps even with approximation ratios that do not depend on k. We discuss how this type of approximation for the combinatorial problem translates into a sub-optimal solution for the maximum likelihood learning problem.

Recall the decomposition of the information divergence that was presented in Fig. 1. When the target distribution is the empirical distribution, the maximum log likelihood of a triangulated graph G is given by $-H(\widehat{P_G})$. Fig. 1 can be viewed as representing the maximum likelihood of Markov networks over $\emptyset$ (fully independent models), Markov networks over G, and the maximum attainable likelihood (the likelihood of the empirical distribution itself), where the vertical axis is the *negative* log-likelihood (higher log likelihood is lower in the figure). The weight of the graph is then the gain in maximum log likelihood versus the fully independent model. A constant factor approximation on the weight of the graph translates to a constant factor approximation on the gain in log likelihood, and hence also provides for a constant factor approximation on the log likelihood itself (approximating the *gain* in log likelihood is a stricter criterion than approximating the log likelihood itself).

Such a constant factor approximation translates to a constant *exponential* factor approximation on the likelihood itself. However, we cannot hope for much more. Since the maximum hypertree problem is NP-hard to approximate to within any additive constant, we can conclude that it is NP-hard to approximate the likelihood (or relative gain in likelihood) to within any multiplicative constant. That is, for any constants $k > 1$ and c, it is NP-hard to find a Markov network of tree-width at most k, with likelihood at least c times the optimal likelihood among Markov networks with tree-width at most k.

The approximation result seems less appealing when viewing the problem as a projection problem of minimizing the information divergence. As can be seen in Fig. 1, a constant factor approximation on the weight of the graph translates to a constant factor approximation on the *reduction* in information divergence versus a fully independent model, and does not provide for a constant factor approximation on the information divergence itself. If the information divergence to the true optimal graph $D(P^{\mathbf{T}} \| G^*)$ is much smaller than the divergence to the completely independent model (and hence much smaller than the reduction in information divergence), then approximating the information divergence itself is a much stricter requirement than approximating the reduction in its value. This happens, for example, when the target distribution is very close to being a tree-width k Markov network.

8. Discussion

We demonstrated how the problem of projecting a distribution onto Markov networks of bounded tree-width can be cast as a combinatorial optimization problem of finding a maximum weight hypertree. By studying the maximum hypertree problem, we were able to prove that the projection problem is NP-hard and to open the door to approximation algorithm with a provable performance guarantee. In particular, we show how an approximation algorithm devised in [10] provides for a (large) constant factor approximation to the log likelihood given an empirical distribution. A large gap remains between this positive result and the computational hardness result. However, now that the maximum weight hypertree problem has been presented to the algorithms community, further progress on it will directly correspond to improved algorithms for projecting a distribution onto Markov networks of bounded tree-width.

Although the hardness results discussed here do not directly imply hardness of efficient PAC learning, the same reverse reduction, with a slightly different analysis, can be used to show that it is hard to probably approximately project a distribution onto Markov networks of bounded tree-width based on samples from the distribution. As with all hardness results about PAC learning, the hardness is with respect to randomized reductions, and establishes that learning cannot be done in polynomial time unless $\mathrm{NP} = \mathrm{RP}$. Such hardness was suspected, but not proved, by Höffgen [9].

An interesting open question concerns efficient PAC learnability of Markov networks of bounded tree-width, when the target distribution is guaranteed to be such a Markov network (this is sometimes referred to as *proper* PAC learnability). The general hardness result discussed above should not deter us from seeking efficient algorithms when the target distribution is guaranteed to be inside the concept class. For example, even though probably approximately projecting a distribution onto Markov networks of bounded path-width is hard in general (as can be shown by a similar reverse reduction from the Hamiltonian path problem), it can be done in polynomial time when the target distribution is guaranteed to be in the concept class [9]. Such proper PAC learnability of Markov networks of bounded tree-width is at least as hard as triangulating a graph (since it effectively finds a triangulation of the target distribution). But even though triangulation is very difficult in practice, it can be done in linear time for fixed width k [4] (other, more practical algorithms, without such theoretical guarantees, are also known [14]), leaving open the possibility for efficient learning algorithms for small widths. Note that if a constant-factor approximation algorithm on the information divergence itself is found, it can also be used for this PAC learning task.

It is also interesting to study the weights that carry the decomposition of the reduction in information divergence. In particular, we might ask if the monotonicity is the true property defining the structure of these weights. That is, is any monotone weight function realizable by some distribution? This question could potentially be answered by extending the reverse reduction of Section 6 from a positive weight function on sets of a fixed number of vertices, to any monotone weight function.

In this work we concentrated on finding maximum likelihood models. It would be interesting to extend this work also to scoring functions that are appropriate for model selection. In fact, minimum description length (MDL) scores can be decomposed to clique weights over triangulated graphs. However, the weights are no longer monotone and the approximation results do not hold. Moreover, although the optimal MDL score might be achieved on a non-triangulated graph, the weights sum up correctly only on triangulated graphs. The hardness results do carry over to MDL scores, i.e. finding the *triangulated* graph of bounded tree-width that minimizes its MDL is NP-hard.

Acknowledgements

I would like to thank Tommi Jaakkola for helpful discussions and advice, and David Karger for working with me on the approximation algorithm presented in the companion paper [10] and for continued advice. I would also like to thank an anonymous reviewer for pointing out Höffgen's work [9].

References

[1] S. Amari, Information geometry on hierarchy of probability distributions, IEEE Trans. Inform. Theory 47 (5) (2001) 1701–1711.
[2] N. Alon, J.H. Spencer, The Probabilistic Method, Wiley, New York, 1991.
[3] J. Besag, Spatial interaction and the statistical analysis of lattice systems, Proc. Roy. Statist. Soc. Ser. B (1974) 192–236.
[4] H.L. Bodlaender, A linear time algorithm for finding tree-decompositions of small treewidth, SIAM J. Comput. 25 (1996) 1305–1317.
[5] D.M. Chickering, Learning Bayesian networks is NP-complete, in: D. Fisher, H.-J. Lenz (Eds.), Learning from Data: AI and Statistics, Vol. V, Springer-Verlag, New York, 1996, pp. 121–130.
[6] C.K. Chow, C.N. Liu, Approximating discrete probability distributions with dependence trees, IEEE Trans. Inform. Theory IT-14 (3) (1968) 462–467.
[7] T.M. Cover, J.A. Thomas, Elements of Information Theory, Wiley–Intersciences, New York, 1991.
[8] S. Dasgupta, Learning polytrees, in: Proc. 15th Conf. on Uncertainty in Artificial Intelligence (UAI-99), Stockholm, Sweden, 1999, pp. 134–141.
[9] K.U. Höffgen, Learning and robust learning of product distributions, in: Proceedings of the Sixth Annual Workshop on Computational Learning Theory, Santa Cruz, CA, 1993, pp. 77–83.
[10] D. Karger, N. Srebro, Learning Markov networks: Maximum bounded tree-width graphs, in: Proceedings of the 12th ACM–SIAM Symposium on Discrete Algorithms, 2001.
[11] F.M. Malvestuto, Approximating discrete probability distributions with decomposable models, IEEE Trans. Syst. Man Cybernetics 21 (5) (1991) 1287–1294.
[12] M. Meila-Predoviciu, Learning with mixtures of trees, Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, 1999.
[13] J. Pearl, Probabilistic Reasoning in Intelligent Systems, Revised 2nd Printing Edition, Morgan Kaufmann, San Mateo, CA, 1997.
[14] K. Shoikhet, D. Geiger, A practical algorithm for finding optimal triangulations, in: Proceedings AAAI-97, Providence, RI, 1997, pp. 185–190.
[15] N. Srebro, Maximum likelihood Markov networks: An algorithmic approach, Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA, 2000.
[16] N. Wermuth, S. Lauritzen, Graphical and recursive models of contingency tables, Biometrika 72 (1983) 537–552.