RC24097 (W0611-004) November 1, 2006
Computer Science

IBM Research Report

Bayesian Learning of Markov Network Structure: Application to Class Probability Estimation

Aleks Jakulin
Department of Statistics
Columbia University
1255 Amsterdam Avenue
New York, NY 10027

Irina Rish
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Bayesian Learning of Markov Network Structure: Application to Class Probability Estimation

Aleks Jakulin JAKULIN@ACM.ORG
Department of Statistics, Columbia University, 1255 Amsterdam Ave, New York, NY 10027, USA

Irina Rish RISH@US.IBM.COM
IBM T.J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532, USA

Abstract

We propose a simple and efficient approach to building undirected probabilistic classification models (Markov networks) that extend naïve Bayes classifiers and outperform existing directed probabilistic classifiers (Bayesian networks) of similar complexity. Our Markov network model is represented as a set of consistent probability distributions on subsets of variables. Inference with such a model can be done efficiently in closed form for problems like class probability estimation. We also propose a highly efficient Bayesian structure learning algorithm for conditional prediction problems, based on integrating along a hill-climb in the structure space. Our prior based on the degrees of freedom effectively prevents overfitting.

1. Introduction

Learning probabilistic models from data has been an area of active and fruitful research in machine learning due to several reasons. First, despite its simplicity, the naïve Bayes (NB) classifier demonstrated surprisingly high accuracy in many domains, and became a popular choice in practice. Its success also led to multiple extensions that attempted to further improve the performance of naïve Bayes by incorporating higher-order dependencies (e.g., tree-augmented naive Bayes and Bayesian networks (Friedman et al., 1997)). Second, in practical applications we are often interested not just in accurate classification, but also in accurate estimation of class probability for solving ranking and cost-based decision problems. Moreover, we may need to learn joint distribution models that allow answering various probabilistic queries besides computing the conditional class probability. A popular choice are graphical probabilistic models such as Markov and Bayesian networks, which also have an advantage of interpretability as they explicitly represent interactions among features.

In this paper, we propose a simple and efficient Bayesian approach that learns undirected probabilistic models (Markov networks). We evaluate our approach on the tasks of class probability estimation and classification. We have chosen undirected models over directed ones since computing the conditional class probability is an easy inference problem that does not require an explicit model of a joint distribution provided by a Bayesian network; it suffices to have an unnormalized representation given by a set of potentials in a Markov network. We also adopt a discriminative structure learning approach (Grossman & Domingos, 2004; Pernkopf & Bilmes, 2005), using a conditional likelihood function to score model structures. Being Bayesian about the structure, we integrate it out, rather than search for a single optimal structure. Our empirical results demonstrate that such Bayesian approach frequently outperforms existing directed probabilistic classifiers of similar complexity (e.g., Bayesian networks with same maximal clique size), while also being extremely fast, sometimes order of magnitudes faster than some competing approaches.

2. Related Work

Most of previous work on probabilistic classifiers focused on directed models, or Bayesian networks. However, we decided to focus on undirected graphical models (Markov networks) since learning explicit (normalized) joint probability distribution $P(\mathbf{X}, Y)$, as in case of Bayesian networks, is unnecessary if our goal is just computing the conditional class probability $P(Y|\mathbf{X})$. This is an easy inference problem even with an unnormalized distribution represented by a Markov net-

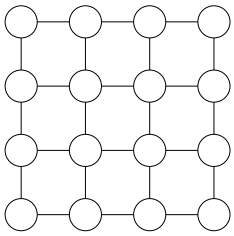

Figure 1. An $n \times n$ lattice Markov network has only pairwise interactions (hyperedge cardinality 2), but the treewidth of n.

work. Undirected models ignore the directionality of associations between variables (e.g., a single Markov net for a joint $P(A, B)$ versus two Bayesian networks $A \rightarrow B$ and $B \rightarrow A$), reducing the complexity of the hypothesis space. On the other hand, undirected models permit the inclusion of a larger number of connections between variables, as we are no longer restricted by the decomposability requirements imposed by the chain rule.

Previous approaches to learning Markov networks often focused on bounded-treewidth models (Chow & Liu, 1968; Meilă & Jordan, 2000; Srebro, 2001; Bach & Jordan, 2002), in order to bound the inference complexity; again, this restriction is unnecessary if we are only concerned with the queries described above. In our approach, we only have to bound the *original* hyperedge cardinality in a Markov network, for the sake of representation efficiency. Note that removing the bounded-treewidth constraint allows to account for important k-way interactions between the variables than the corresponding bounded-treewidth model would have to ignore. For example, consider an $n \times n$ Markov network (e.g., Ising model) in Figure 1. It is well-known that its treewidth equals n, so a bounded-treewidth model with bound $k < n$ would have to ignore many pairwise interactions, while our approach could potentially include all of them.

Note that despite being related, our approach is also different from the conditional random fields (CRFs) (Lafferty et al., 2001) and other approaches such as max-margin Markov networks for sequential classification (Taskar et al., 2003). We focus on 'standard" i.i.d. rather than sequential non-i.i.d. classification problem, and learn a Markov network over the features and class, rather than (conditional) Markov network (random field) over a sequence of dependent class labels. Extending our approach to CRFs would be an interesting direction for future work. Our Bayesian prior which depends on the complexity of the structure can be seen as an approach to penalization of complex structure, just as the maximum-margin criterion penalizes unusually oriented decision boundaries.

3. Markov Network Models

3.1. Notation and Overview

Let $\mathbf{X} = \{X_1, \ldots, X_n\}$ be a set of observed random variables, called *attributes*, and let $\mathbf{x} = (x_1, \ldots, x_n)$ be a vector of values assigned to variables in $\mathbf{X}$. Herein, we assume discrete-valued attributes, i.e. $\mathbf{x} \in \mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$ where each range $\mathcal{X}_i$ is a set of possible values of X_i. Let Y denote an unobserved random variable called the *class*, where $y \in \mathcal{Y}, |\mathcal{Y}| = m$. The set of attributes together with the class (i.e., all variables) is denoted $\mathbf{V} = \mathbf{X} \cup \{Y\}$. An assignment $\mathbf{v}^{(i)} = (\mathbf{x}^{(i)}, y^{(i)})$ of values to the attributes and the class is called an *instance*, or *example* with index (i). We will use a short notation $P(\mathbf{v}) = P(\mathbf{x}, y) = P(x_1, \ldots, x_n, y)$ to describe the joint probability distribution $P(X_1 = x_1, \ldots, X_n = x_n, Y = y)$.

Our models will have the undirected structure of Markov networks. We will define a *Markov network*, or *Markov random field* on random variables $\mathbf{V}$ as $\langle \mathcal{M}, \mathcal{T} \rangle$ where $\mathcal{M}$ is an (undirected) hypergraph $\mathcal{M} = \{\mathcal{S}_1, S_2, \ldots, S_\ell\}$ and $\mathcal{T} = (\Phi_1, \ldots, \Phi_\ell)$ is a set of positive functions, called *potentials* for each of the ℓ hyperedges[1] in $\mathcal{M}$, such that the joint distribution $\hat{P}(\mathbf{v})$ factorizes over them: $\hat{P}(\mathbf{v}) = (1/Z) \prod_{i=1}^{\ell} \Phi(\mathbf{v}_i)$ where Z is a normalization constant. This latter form is referred to as the Gibbs distribution. We use $\hat{P}(\cdot)$ as a shorthand for $P(\cdot|\langle \mathcal{M}, \mathcal{T} \rangle)$. Each hyperedge S_R contains the variables linked to it. These variables form a vector $\mathbf{V}_R$. The potential $\Phi(\mathbf{v}_R)$ corresponding to each hyperedge then maps any combination of values of $\mathbf{v}_R$ into a positive real number.

We now outline our algorithm for class probability estimation. The outline contains many terms that will be defined later, in the section referenced for each step.

1. Given $\mathbf{V} = \mathbf{X} \cup \{Y\}$, and a bound k on hyperedge cardinality, select a set of hyperedges $\mathcal{M} = \{M | M \subseteq \mathbf{Y}\}$ using the approach described in Sect. 4.2.
2. Given $\mathcal{M}$, compute the region graph $\mathcal{R}$ using the cluster variation method where each hyperedge corresponds to an initial region (Sect. 3.2). The region graph captures the overlap between hyperedges.
3. For each region R estimate the submodel $P(\mathbf{V}_R)$ from data (Sect. 4.1). Each submodel is an ordinary probabilistic model, but for a subset of vari-

[1] Usually referred to as 'cliques', but with hypergraphs the notion of a clique could be confusing.

ables.

4. Approximate $P(\mathbf{V})$ by the (non-normalized) product $\Phi(\mathbf{v}) = \prod_{\langle R,c_R\rangle\in\mathcal{R}} P(\mathbf{v}_R)^{c_R}$ where c_R is the counting number for region R in the region graph (Sect. 3.2).
5. Compute $\hat{P}(y|\mathbf{x}) = \Phi(\mathbf{x},y)/\sum_{y'}\Phi(\mathbf{x},y')$ and possibly classify $y^*(\mathbf{x}) = \arg\max_y P(y|\mathbf{x})$.

3.2. Computing the Potentials

The general problem with learning Markov networks from data once the structure is known is how to obtain potentials from the data. Specifically, we tractably express the potentials in terms of *submodels*, where a submodel $P(\mathbf{v}_R)$ is a probability distribution or mass function on the subset of variables corresponding to each hyperedge. Each submodel is estimated from the data. We then make use of the following recursive definition of potentials Φ_R (Srebro, 2001):

$$\Phi_R(\mathbf{v}_R) \triangleq \frac{P(\mathbf{v}_R)}{\prod_{R'\subset R}\Phi_{R'}(\mathbf{v}_{R'})}. \qquad (1)$$

A particular $P(\mathbf{v}_S)$, $S \subset R_1$ is computed by marginalizing $P(\mathbf{v}_{R_1})$, which in turn is modeled directly from data. As S may be a part of another hyperedge $S \subset R_2$, there could be several versions of $P(\mathbf{v}_S)$, depending on what submodel is marginalized (R_1 or R_2). To assure *consistency* we require that there exists some hypothetical $P(\mathbf{V})$ so that each $P(\mathbf{v}_R)$ is its marginalization.

It is of practical convenience to construct an intermediate data structure called a *region graph* $\mathcal{R}$ (Yedidia et al., 2005). Algorithm 1, known as the *cluster variation method*, shows how the region graph is constructed from the set of hyperedges. The region graph is defined as $\mathcal{R} = \{\langle R, c_R\rangle, R \subseteq \mathbf{V}\}$, where for each region R, there is a corresponding *counting number* c_R, that accounts for the overlaps between regions, and helps avoid the double-counting of evidence.

Given the region graph, we can compute the joint probability distributions as:

$$\hat{P}(\mathbf{v}) = \frac{1}{Z}\prod_{\langle R,c_R\rangle\in\mathcal{R}} P(\mathbf{v}_R)^{c_R}. \qquad (2)$$

It is well-known (Pearl, 1988) that when the Markov network is triangulated and thus yields a clique tree, the Gibbs distribution can be represented exactly through (2) and no normalization is needed, as $P(\mathbf{v}) = \prod_{R\in\mathcal{R}}\Phi_R(\mathbf{v}_R)$, where the potentials $\Phi_R(\mathbf{v}_R)$ are defined by (1). In general, when the counting numbers are greater than zero only for the initial regions, the recursive definition of potentials is exact (Yedidia et al., 2005).

$\mathcal{R}_0 \leftarrow \{\emptyset\}$ {Redundancy-free set of hyperedges.}
for all $S \in \mathcal{M}$ **do** {for each hyperedge}
  if $\forall S' \in \mathcal{R}_0 : S \not\subseteq S'$ **then**
    $\mathcal{R}_0 \leftarrow \mathcal{R}_0 \cup \{S\}$ {S is not redundant}
  end if
end for
$\mathcal{R}_0 \leftarrow \{\langle S, 1\rangle;\ S \in \mathcal{R}_0\}$
$k \leftarrow 1$
while $|\mathcal{R}_{k-1}| > 2$ **do** {there are feasible subsets}
  $\mathcal{R}_k \leftarrow \{\emptyset\}$
  for all $\mathcal{I} = S^\dagger \cap S^\ddagger : S^\dagger, S^\ddagger \in \mathcal{R}_{k-1}, \mathcal{I} \notin \mathcal{R}_k$ **do** {feasible intersections}
    $c \leftarrow 1$ {the counting number}
    for all $\langle S', c'\rangle \in \mathcal{R}, \mathcal{I} \subseteq S'$ **do**
      $c \leftarrow c - c'$ {consider the counting numbers of all regions containing the intersection}
    end for
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{\langle \mathcal{I}, c\rangle\}$
    $\mathcal{R}_k \leftarrow \mathcal{R}_k \cup \{\mathcal{I}\}$
  end for
end while
return $\{\langle R, c\rangle \in \mathcal{R}; c \neq 0\}$ {Region graph.}

Algorithm 1: Cluster variation method for constructing the region graph given a set of hyperedges $\mathcal{M} = \{S_1, S_2, \ldots, S_\ell\}$.

3.3. Performing Inference

While in general it is NP-hard to compute $P(\mathbf{Q}|\mathbf{E})$, where $\mathbf{Q} \subseteq \mathbf{V}$, $\mathbf{E} \subseteq \mathbf{V}$, in a Markov network representing a joint distribution $P(\mathbf{V})$, the problem becomes easy when the number of unobserved variables $\mathbf{V} \setminus \mathbf{E}$ is *small*, or when the *treewidth* of the network is small. Treewidth, also known as induced width, is a graph parameter that controls the complexity of some commonly used probabilistic inference algorithms (the complexity is exponential in the treewidth). The treewidth of a networks, given a particular variable ordering, equals to largest clique size of the *triangulated* network, where the triangulation is performed along the given ordering and reflects the process of creating new probabilistic functions by the inference algorithm.

Given a set of random variables $\mathbf{V} = \mathbf{X} \cup \{Y\}$, a set $\mathcal{R} = \{R|R \subseteq \mathbf{V}\}$ of subsets (regions) of $\mathbf{V}$, where Y belongs to at least one region, and a product $\Phi(\mathbf{v}) = \Phi(\mathbf{x},y) = \prod_{R\in\mathcal{R}}\Phi_R(\mathbf{v}_R)$ of non-negative functions (potentials) defined on these regions, let $\hat{P}(\mathbf{v}) = (1/Z)\Phi(\mathbf{v})$ be the corresponding joint probability distribution over $\mathbf{V}$, where Z is a normalization constant. It is very easy to see that:

1. Computing $\hat{P}(Y|\mathbf{x})$ does not require global normalization, i.e. $\hat{P}(Y|\mathbf{x}) = \Phi(\mathbf{x},Y)/\sum_{y'}\Phi(\mathbf{x},y')$;[2]

[2]Indeed, the first claim follows from $\hat{P}(y|\mathbf{x}) = \hat{P}(\mathbf{x},y)/\hat{P}(\mathbf{x}) = (1/Z)\Phi(\mathbf{x},y)/\sum_{y'}(1/Z)\Phi(\mathbf{x},y')$, since by definition $\Phi(\mathbf{v}) = \Phi(\mathbf{x},y)$.

2. The classifier can be computed using a product of only those potentials that contain Y, i.e. $h^*(\mathbf{x}) = \arg\max_y \prod_{\{R\in\mathcal{R}|Y\in R\}} \Phi_R(\mathbf{v}_R)$.[3]

Of course, this holds also when we have several query variables $\mathbf{Y}$, but only the vector is short. More complex queries (e.g. with missing data) might require several iterations, where each individual iteration can take the simple form as for inferring the class probability.

4. Bayesian Structure Learning

The above formulation of the Markov network model allows efficient inference. The task for learning is to determine the parameters of the model: the *structure* and the *submodels*. We will adopt the Bayesian framework, based on an explicit description of the model in terms of its parameters $\phi = \langle \mathcal{M}, \boldsymbol{\Theta}, \vartheta \rangle$, where $\mathcal{M}$ is the model structure (hypergraph), while ϑ and $\boldsymbol{\Theta}$ are the submodel prior and the submodel parameters, respectively. Each submodel $\mathbf{V}_R$ is specified in terms of a parameter vector $\boldsymbol{\theta}_R$, so that $P(\mathbf{V}_R|\boldsymbol{\theta}_R)$.

We will assume a prior distribution over structures $P(\mathcal{M})$, and a prior distribution over the submodel parameters $P(\boldsymbol{\Theta}|\vartheta)$. The prior for the whole model is then $P(\phi) = P(\mathcal{M})P(\vartheta)P(\boldsymbol{\Theta}|\vartheta) = P(\mathcal{M})P(\vartheta)\prod_R P(\boldsymbol{\theta}_R|\vartheta)$. Because we assume independence of $\boldsymbol{\Theta}$ and $\mathcal{M}$, the submodels remain the same irrespectively of the structure: this results in a major speed-up.

The Bayesian paradigm (to be distinguished from the Bayes rule) is that one should be uncertain about what the exact model is. Instead of finding the 'best' model parameters, we assign probabilities to each setting of ϕ, 'averaging' together a weighted ensemble of models (both structures and parameters). For prediction we make use of all plausible structures instead of arbitrarily picking just the best one (Friedman & Koller, 2003). This has also been shown to improve results in practice (Cerquides & López de Màntaras, 2003). In a class probability estimation setting, the final result of our inference based on data $\mathcal{D}$ will be the following class predictive distribution:

$$P(y|\mathbf{x}) \propto \int P(\phi|\mathcal{D})P(y|\mathbf{x},\phi)d\phi \qquad (3)$$

[3]The second claim is easily obtained from the definition of Bayesian classifier, $h^*(\mathbf{x}) = \arg\max_y \hat{P}(y|\mathbf{x})$, and the following observation: $\hat{P}(y|\mathbf{x}) = \frac{\Phi(\mathbf{x},y)}{\sum_{y'}\Phi(\mathbf{x},y')} = \frac{\prod_{\{Q\in\mathcal{R}|Y\notin Q\}}\Phi(\mathbf{v}_Q)}{\sum_{y'}\Phi(\mathbf{x},y)}\prod_{\{R\in\mathcal{R}|Y\in R\}}\Phi_R(\mathbf{v}_R)$, where $(\prod_{\{Q\in\mathcal{R}|Y\notin Q\}}\Phi(\mathbf{v}_Q))/\sum_{y'}\Phi(\mathbf{x},y)$ is independent of Y.

Here, $P(y|\mathbf{x},\phi)$ is based on (2). For efficiency purposes, we employ the formulation of Bayesian model averaging (Hoeting et al., 1999), where only those parameter values with a sufficiently high posterior probability are remembered and used.

4.1. Parameters for Consistent Submodels

Our Markov network model is based on partially overlapping submodels. Although technically not necessary, it is desirable for the submodels to be consistent in the sense that all of them are marginalizations of some joint model. We model the submodels on discrete variables as multinomials with a symmetric Dirichlet prior:

$$P(\boldsymbol{\theta}_R|\vartheta) = \text{Dirichlet}(\alpha_R,\ldots,\alpha_R),\quad \alpha_R = \frac{\vartheta}{\prod_{V\in R}|\mathcal{V}|}$$

It is easy to prove that this prior assures that all the posterior mean submodels are consistent if the same value of ϑ was used for each of them. This prior is best understood as the expected number of outliers: to any data set, we add ϑ instances uniformly distributed across the space of variables. We have set the parameter $\vartheta = 1$, which means that one outlier per dataset was assumed: we see this to be a reasonable prior assumption that speeds up the learning. Due to conjugacy of the Dirichlet prior, the desired posterior mean probability given data $\mathcal{D}$ within region R is simply

$$P(\mathbf{v}_R|\mathcal{D},\vartheta) = \frac{\vartheta/|\mathcal{V}_R| + \sum_i^{|\mathcal{D}|}\mathbb{I}\{\mathbf{v}_R^{(i)} = \mathbf{v}_R\}}{|\mathcal{D}| + \vartheta}.$$

4.2. Structure Learning

4.2.1. Parsimonious Structures

The structure in the context of our Markov network model is simply a selection of the submodels. $P(\mathcal{M})$ models our prior expectations about the structure of the model. We will now introduce a parsimonious prior that asserts a higher prior probability to simpler selections of submodels, and a lower prior probability to complex selections of submodels as to prevent overfitting. A quantification of complexity based on degrees of freedom is given by (Krippendorff, 1986). In many practical applications we are not interested in the joint model. Instead, we want to predict labels Y from attributes $\mathbf{X}$. In such cases, a considerable part of uncertainty about the value of $\mathbf{X}$ gets canceled out, and the effective degrees of freedom are fewer ("Conditional density estimation is easier than joint density estimation.").

Let us assume a set of overlapping submodels of the vector $\mathbf{V}$, and the resulting region graph $\mathcal{R}$ obtained using the CVM. The number of *degrees of freedom* of the model $\mathcal{M}$ with a corresponding region graph $\mathcal{R}$ intended for predicting Y from $\mathbf{X}$ is:

$$df_{\mathcal{M}_Y} \triangleq \sum_{\langle S,c\rangle\in\mathcal{R}} c\left(\prod_{V\in S}|\mathcal{V}| - \prod_{\substack{V\in\mathcal{S}\\V\neq Y}}|\mathcal{V}|\right) \qquad (4)$$

V is either Y or a part of $\mathbf{X}$, and $\mathcal{V}$ is the number of values V can take. This quantification accounts for overlap between submodels in the same fashion as cluster variation method does for probabilities. Of course, conditional modeling corresponds to joint modeling when $Y = \emptyset$.

The following prior corresponds to the assumption of exponentially decreasing prior probability of a structure with an increasing number of degrees of freedom (or effective parameters):

$$P(\mathcal{M}_Y) \overset{\triangle}{\propto} \exp\left\{-\frac{m\ df_{\mathcal{M}_Y}}{m - df_{\mathcal{M}_Y} - 1}\right\} \qquad (5)$$

We discourage the degrees of freedom from exceeding the number of training instances $m = |\mathcal{D}|$. This prior also corresponds to the Akaike information criterion (AIC) with small-sample correction (Burnham & Anderson, 2002). However, due to dependence on the number of instances m, some would not consider this to be a Bayesian prior. An orthodox Bayesian choice would then be $P(\mathcal{M}) \overset{\triangle}{\propto} e^{df_{\mathcal{M}}}$.

The likelihood function for conditional modeling can also be adjusted to account for the fact that we will be using the model for predicting Y from $\mathbf{X}$. The non-Bayesian approach searches for the structure that yields the maximum conditional likelihood (Grossman & Domingos, 2004). A Bayesian approach instead scores structures by the means of a conditional likelihood function, as is customary in Bayesian regression (Gelman et al., 2004). We hereby use the following conditional likelihood function that assumes i.i.d.:

$$P(\mathbf{v}^{(1)\ldots(m)}|\phi) \triangleq \prod_{i=1}^{m} P(\mathbf{y}^{(i)}|\mathbf{x}^{(i)},\phi) \qquad (6)$$

Because $\mathcal{M}$ was assumed to be independent of ϑ and $\mathbf{\Theta}$, we prepare $\mathbf{\Theta}$ in advance, before assessing $\mathcal{M}$. The $P(\mathbf{y}^{(i)}|\mathbf{x}^{(i)},\mathcal{M})$ is obtained using (2).

4.2.2. Sampling the Structure Space

In the process of structure learning, we perform a walk in the space of structures. For all practical purposes, we are not interested in the 'best' structure, but the walk should nevertheless attempt to visit more structures with high posterior probability than structures with low posterior probability, as the latter do not affect the predictive distribution (3) much. While MCMC approaches have been proposed in the past (Friedman & Koller, 2003), we apply a simple hill-climbing approach that yields good results for a lower computational cost.

During the hill climb, we seek to greedily maximize the posterior probability of a structure. Let us assume that we are performing conditional modeling, with the intention of predicting Y. Our initial structure will have a single initial hyperedge of cardinality 1, $\{Y\}$. In the successive step, we will consider all possible attributes X_i creating hyperedges $\{X_i\}\cup\{Y\}$, and pick the one that yields the highest posterior probability: this corresponds to step-wise forward selection algorithm with one-step look-ahead. This approach is very efficient, as observed also by (Caruana et al., 2004): including a new hyperedge corresponds to just multiplying the predictions for an individual instance with another term and renormalizing. With the considerable increase in performance that ensues, we can afford to find the best hyperedge at every step of the forward selection.

When no hyperedge of cardinality k results in an increase of posterior probability, we start searching through hyperedges of cardinality $k+1$, and so on. An example of a consequence of this stage-based search is that we prevent immediately adding the hyperedge ABY if adding AY and BY is just as good. We always add hyperedges, and never delete them. Of course, a larger hyperedge may cover smaller hyperedges, effectively eliminating them. To limit the search time, we terminate when k reaches a particular value (e.g. 4), and we heuristically select a number (e.g., 1000) of most promising hyperedges for each k. The promise of a hyperedge is calculated from the results of its subsets.

All the hyperedges we introduce include Y. As described in Sect. 3.3, hyperedges that do not include any of the variables in Y do not affect the predictions for Y if the values for $\mathbf{X}$ are given. For that reason, conditional modeling allows working with larger hyperedges with the combinatorial explosion occurring later than with joint modeling. In a more complex situation of structured labels, we could assume a particular pattern of hyperedges.

At some point, we will reach the local maximum posterior probability peak, and no improvement in posterior probability will be possible. However, we do continue to search further for a few more iterations, as

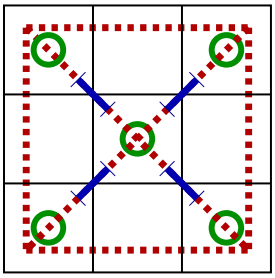

Figure 2. Hyperedges of cardinality 4 are not merely a theoretical curiosity. In this illustration we show the tic-tac-toe game board, which comprises 9 squares, each corresponding to a 3-valued variable with the range $\{\times, \circ, \_\}$. The goal is to develop a predictive model that will indicate if a board position is winning for $\times$ or not: this is the 2-valued class variable. The illustration shows the hyperedges in the MAP model identified by our algorithm: 2-way hyperedges (5 green circles), 3-way hyperedges (4 blue serif lines), and 4-way hyperedges (6 red dashed lines). Each hyperedge includes the class (not shown).

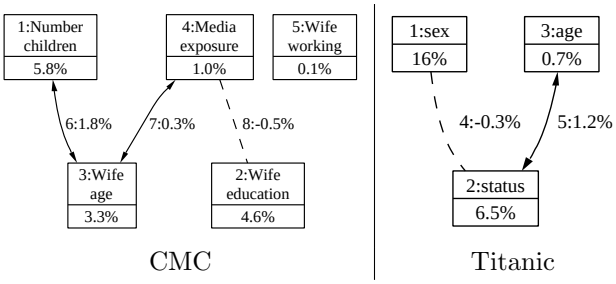

Figure 3. This figure shows the Bayesian model average for two real-life datasets: CMC (contraception use in Indonesia) and Titanic (survival of Titanic passengers). All the posterior mass for Titanic is concentrated in the model that assumes two 3-variable hyperedges, [status of the passenger, age, survival] and [status, sex, survival]. Each node and each connection is numbered with the step of the hill-climb when it was selected. If the posterior probability is high enough, a number between 0 and 1 indicates the relative weight of that model in the class-predictive distribution. For CMC three structures are used to make predictions. The percentages indicate the interaction information expressed as a proportion of class entropy: which helps understand the nature of the hyperedge.

those structures may still have a high enough posterior probability to affect the Bayesian model averaging. We stop the search when the posterior probability is less than a percent of the maximum posterior structure probability. Furthermore, all the models that were evaluated are included in the model average: even if they were not selected, they might still have a relatively high posterior probability.

With the above algorithm we can discover very interesting structures in a very short amount of time. An example of a maximum posterior probability structure for the tic-tac-toe dataset is shown in Fig. 2: the structure was obtained in 0.03 seconds on an ordinary laptop computer. The hyperedges correspond to meaningful notions of corner and center points, to connections between them, and finally to the diagonals and edges: indeed these structures are what humans examine when playing the game. Another example of structures obtained with our algorithm appears in Fig. 3.

5. Empirical Evaluation

To validate our modeling approach from Sections 3 and 4, we have applied the methodology to the problem of class-probability estimation. Numerous techniques exist for this purpose, and they can be roughly divided into those that pursue a discriminative structure, yet employ the generative chain rule (such as the naïve Bayes, tree-augmented naïve Bayes (Friedman et al., 1997) and general Bayesian network classifiers (Grossman & Domingos, 2004)) and those that employ both discriminative structure and discriminative parameter values (Pernkopf & Bilmes, 2005; Greiner et al., 2005; Jing et al., 2005). It is widely recognized that it is generally too hard to perform both general structure search and optimization of discriminative parameter values. Still, a limited amount of structure selection is performed even with discriminative parameter values, such as step-wise model selection (Roos et al., 2005; Madigan et al., 2005) or TAN-like structures (Pernkopf & Bilmes, 2005; Jing et al., 2005), but rarely one can afford an exhaustive search for interactions.

We will evaluate the benefit gained by a) allowing hyperedges that result in cyclic dependencies, b) the benefits of Bayesian model averaging, and c) verifying if our prior protects against overfitting. Furthermore, we compare our approach to other related approaches.

To evaluate a class-probability estimate, we will use the expected negative log-likelihood (log-loss) of class assignment $-E[\log P(y|\mathbf{x})]$. For each of the 39 UCI data sets, we performed 5 replications of 5-fold cross-validation. The data sets were all discretized with the Fayyad-Irani method beforehand, and the missing values were interpreted as special values. The structure learning time with our procedure for *all* the 39 datasets using our method was less *54 seconds* on a laptop computer using our implementation in Python and C++. Of course, one has to account for the fact that each dataset is trained 25 times (5 folds, 5 replications).

Judging from the rankings in Table 1, we can conclude that the single best-performing feature is Bayesian

	log-loss / instance									
domain	NB	TN	BC	M2	M3	M4	M2	BT	B3	B4
adult	*·0.42*	0.33	0.39	0.31	0.30	0.30	0.31	0.30	0.30	**0.30**
glass	1.25	*·1.76*	1.21	1.12	1.12	1.12	0.99	**0.99**	**0.99**	**0.99**
horse-colic	1.67	*·5.97*	3.36	0.83	0.83	0.83	0.82	**0.82**	**0.82**	**0.82**
iris	0.27	0.32	0.20	0.27	0.27	0.27	0.18	**0.18**	**0.18**	**0.18**
lymph	1.10	1.25	1.23	0.98	0.98	0.98	0.79	**0.79**	**0.79**	**0.79**
monk2	0.65	0.63	0.61	*·0.65*	0.54	0.45	0.65	0.60	0.53	**0.45**
p-tumor*	3.17	*·4.76*	2.84	2.65	2.65	2.65	**2.55**	**2.55**	**2.55**	**2.55**
tic-tac-toe	*·0.55*	0.49	0.52	0.53	0.42	0.08	0.53	0.52	0.42	**0.07**
titanic	0.52	0.48	0.48	*·0.52*	0.48	0.48	0.52	0.48	0.48	**0.48**
vehicle	*·1.78*	1.14	1.29	0.82	0.69	0.69	0.80	0.66	**0.66**	**0.66**
voting	*·0.60*	0.53	0.48	0.16	0.21	0.21	0.14	0.14	**0.14**	**0.14**
zoo*	0.38	0.46	0.51	0.40	0.40	0.40	**0.38**	**0.38**	**0.38**	**0.38**
breast-wisc	0.21	0.23	*·0.25*	0.17	0.21	0.21	0.16	**0.16**	0.16	0.16
cmc	1.00	*·1.03*	1.00	0.93	0.93	0.93	0.93	**0.92**	0.92	0.92
hepatitis	0.78	*·1.31*	1.11	0.48	0.48	0.48	0.40	**0.39**	0.39	0.39
ionosphere	0.64	0.74	*·1.70*	0.38	0.39	0.39	0.31	**0.30**	0.30	0.30
wdbc	0.26	0.29	0.39	0.14	0.15	0.15	0.13	**0.13**	0.13	0.13
australian	0.46	*·0.94*	0.78	0.37	0.39	0.41	**0.35**	0.37	0.38	0.37
balance	0.51	*·1.13*	0.74	0.51	0.51	0.51	**0.51**	0.51	0.51	0.51
breast-LJ	0.62	0.89	0.80	0.57	0.67	0.67	**0.56**	0.58	0.58	0.58
crx	0.49	*·0.93*	0.91	0.36	0.37	0.37	**0.35**	0.35	0.35	0.35
german	0.54	*·1.04*	1.00	0.53	0.64	0.65	**0.52**	0.58	0.59	0.59
heart	1.25	*·1.53*	1.38	1.10	1.11	1.11	**1.09**	1.09	1.09	1.09
lung*	5.41	*·6.92*	3.05	2.37	2.37	2.37	**1.18**	1.18	1.18	1.18
pima	0.50	0.49	0.50	0.48	0.49	0.51	**0.48**	0.48	0.48	0.48
post-op	0.93	1.78	1.25	0.79	0.79	0.79	**0.67**	0.67	0.67	0.67
segment	0.38	1.06	*·1.29*	0.17	0.17	0.17	**0.17**	0.17	0.17	0.17
hayes-roth	0.46	*·1.18*	0.76	**0.45**	**0.45**	**0.45**	0.45	0.45	0.45	0.45
lenses	2.44	*·2.99*	1.15	**0.34**	**0.34**	**0.34**	0.40	0.40	0.40	0.40
monk1	*·0.50*	0.09	0.09	0.49	0.08	**0.01**	0.49	0.08	0.08	0.02
ecoli	0.89	0.94	**0.67**	0.85	0.85	0.85	0.81	0.81	0.81	0.81
monk3	0.20	0.11	**0.08**	0.20	0.11	0.11	*·0.20*	0.11	0.11	0.11
o-ring	0.83	0.76	**0.59**	1.41	1.41	1.41	0.67	0.67	0.67	0.67
bupa	0.62	**0.60**	0.61	0.62	0.62	0.62	0.63	0.61	0.61	0.61
car	*·0.32*	**0.18**	0.18	0.32	0.19	0.19	0.32	0.19	0.19	0.19
mushroom	*·0.01*	**0.00**	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
shuttle	0.16	**0.06**	0.06	0.17	0.07	0.07	*·0.17*	0.07	0.07	0.07
soy-small*	0.00	**0.00**	*·0.39*	0.00	0.00	0.00	0.00	0.00	0.00	0.00
wine	**0.06**	0.29	*·0.46*	0.19	0.19	0.19	0.11	0.11	0.11	0.11
rank/LL	7.8	7.8	7.4	6.0	5.8	5.7	4.7	3.4	3.4	**3.1**
rank/ER	5.8	7.0	5.8	6.5	5.6	5.5	6.1	4.7	**4.5**	**4.5**

Table 1. A comparison of different undirected and directed probability models on 39 datasets. NB is naïve Bayes, TN is tree-augmented naïve Bayes, BC is the discriminative search for Bayesian network classifiers (Grossman & Domingos, 2004), M2-M4 is the maximum a posteriori Markov network with structure search with maximum hyperedge cardinality of 2 through 4, B2-B4 are corresponding Bayesian model averaged Markov networks, and BT is the Bayesian model averaging (BMA) on cycle-free Markov hypertrees with hyperedges of cardinality less than 4. The best result is typeset in bold, the results of those methods that outperformed the best method in at least 2 of the 25 experiments are underlined, and the worst result is marked with $(\cdot)$. At the bottom we list the average rank of a method across all the datasets, both for log-loss (LL) and error rate (ER).

method	**time**	**LL**	**ER**
Bayesian multinomial regression	4.84m	**2.7**	**3.7**
LIBSVM dot product kernel	9.11m	3.5	4.2
LIBSVM RBF kernel	12.72m	3.2	4.0
Markov nets $k = 4$ + BMA	6.2m	4.0	5.0
C4.5	0.03m	4.9	5.7
naïve Bayes	0.01m	6.3	5.7
TAN	0.05m	6.6	6.2

Table 2. A rank comparison achieved by several types of models on an extended set of 46 UCI datasets.

model averaging: it has consistently outperformed the maximum a posteriori structures. The second important conclusion is that our Bayesian prior successfully prevents overfitting in a systematic way: as we increase the depth of structure search, the results improve (although B2 does win by performing essentially just feature selection in a number of cases when there seem to be no higher-order hyperedges). The third conclusion is that Markov networks perform well regardless of whether the task is classification (error rate, ER), or class probability estimation (log-loss, LL). The fourth conclusion is that allowing cycles does help, but not in a radical way (of course this may be simply due to our simplified way of computing potentials).

Discriminative Bayes network classifiers, TAN and NBC are quite consistently outperformed by Markov networks. B4 was significantly outperformed by TAN on datasets car, mushroom, and soy-small. It seems that the prior is a bit too conservative on artificial datasets such as car and mushroom, and on small datasets with a large number of attributes.

Comparisons beyond graphical models. We also performed a 'reality check', performing comparison with top methods outside of the graphical model family. A discriminative structure with generative submodels seems to be on average outweighed by discriminative parameters without structure search. In Table 2 we show the results of several popular types of learning algorithms that require no tuning. SVM does outperform our Markov network model. We have used LIBSVM (Chang & Lin, 2005), a very well-performing implementation of support vector machines which also supports class probability estimation and multiclass problems. But the recently introduced implementation of Bayesian multinomial regression (Madigan et al., 2005) outperforms SVM both in log-loss, in error rate and in performance. BMR is a fully linear model which includes no structure search.

6. Conclusion

In summary, we feel that undirected models have many advantages over directed models, especially as it is not possible or at least controversial to establish causal direction from observational data. Undirected models should deserve more attention. Our priors and Bayesian model averaging work surprisingly well and effectively prevent overfitting. Our heuristic structure search is also much faster than most alternatives; we could dare to say that it is one of the best, but could definitely be made less ad hoc. As for other further work, it would be highly desirable to combine the handling of higher-order interactions in Markov networks with effective discriminative parameter learning in regression models. This could perhaps be achieved by finding discriminative parameters for well-performing discriminative structures, or by finding an equally efficient way of performing inference on Markov networks but for the specific purpose of conditional prediction.

References

Bach, F., & Jordan, M. (2002). Thin junction trees. *Advances in Neural Information Processing Systems 14* (pp. 569–576).

Burnham, K. P., & Anderson, D. R. (2002). *Model selection and multimodel inference*. Springer. 2nd edition.

Caruana, R., Niculescu, A., Crew, G., & Ksikes, A. (2004). Ensemble selection from libraries of models. *Proc. 21st ICML*. Banff, Alberta, Canada.

Cerquides, J., & López de Màntaras, R. (2003). Tractable Bayesian learning of tree augmented naive Bayes classifiers. *Proc. 20th ICML* (pp. 75–82).

Chang, C.-C., & Lin, C.-J. (2005). *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chow, C. K., & Liu, C. N. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Information Theory*, *14*, 462–467.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, *29*, 131–163.

Friedman, N., & Koller, D. (2003). Being Bayesian about network structure: A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, *50*, 95–126.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2004). *Bayesian data analysis*. Boca Raton, FL: Chapman & Hall/CRC. 2nd edition.

Greiner, R., Su, X., Shen, B., & Zhou, W. (2005). Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. *Machine Learning*, *59*, 297–322.

Grossman, D., & Domingos, P. (2004). Learning Bayesian network classifiers by maximizing conditional likelihood. *Proc. 21st ICML* (pp. 361–368). Banff, Canada: ACM Press.

Hoeting, J. A., Madigan, D., Raftery, A. E., & Volinsky, C. T. (1999). Bayesian model averaging: A tutorial. *Statistical Science*, *14*, 382–417.

Jing, Y., Pavlovic, V., & Rehg, J. M. (2005). Efficient discriminative learning of Bayesian network classifiers via boosted augmented naive Bayes. *Proc. 22nd ICML* (pp. 369–376). Bonn, Germany: ACM Press.

Krippendorff, K. (1986). *Information theory: Structural models for qualitative data*, vol. 07–062. Beverly Hills, CA: Sage Publications, Inc.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proc. of the International Conference on Machine Learning (ICML)* (pp. 282–289).

Madigan, D., Genkin, A., Lewis, D. D., & Fradkin, D. (2005). Bayesian multinomial logistic regression for author identification. *25th MaxEnt Workshop*. San Jose.

Meilă, M., & Jordan, M. I. (2000). Learning with mixtures of trees. *Journal of Machine Learning Research*, *1*, 1–48.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems*. San Francisco, CA, USA: Morgan Kaufmann.

Pernkopf, F., & Bilmes, J. (2005). Discriminative versus generative parameter and structure learning of Bayesian network classifiers. *Proc. 22nd ICML* (pp. 657–664). Bonn, Germany: ACM Press.

Roos, T., Wettig, H., Grünwald, P., Myllymäki, P., & Tirri, H. (2005). On discriminative Bayesian network classifiers and logistic regression. *Machine Learning*, *59*, 267–296.

Srebro, N. (2001). Maximum likelihood bounded tree-width Markov networks. *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI)* (pp. 504–511).

Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin Markov networks. *Neural Information Processing Systems Conference 16*. Vancouver, Canada.

Yedidia, J. S., Freeman, W. T., & Weiss, Y. (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, *51*, 2282–2312.