Mach Learn (2009) 76: 195–209
DOI 10.1007/s10994-009-5130-x

Learning multi-linear representations of distributions for efficient inference

Dan Roth · Rajhans Samdani

Received: 13 June 2009 / Revised: 13 June 2009 / Accepted: 16 June 2009 / Published online: 23 July 2009

Abstract We examine the class of multi-linear representations (MLR) for expressing probability distributions over discrete variables. Recently, MLR have been considered as intermediate representations that facilitate inference in distributions represented as graphical models.

We show that MLR is an expressive representation of discrete distributions and can be used to concisely represent classes of distributions which have exponential size in other commonly used representations, while supporting probabilistic inference in time linear in the size of the representation. Our key contribution is presenting techniques for learning bounded-size distributions represented using MLR, which support efficient probabilistic inference. We demonstrate experimentally that the MLR representations we learn support accurate and very efficient inference.

Keywords Learning probability distributions · Multi-linear polynomials · Probabilistic inference · Graphical models

1 Introduction

A significant fraction of the work in Artificial Intelligence deals with probabilistic inference which necessitates reasoning in terms of representations of probability distributions. In the Machine Learning community, graphical models like Bayes Nets (BN) (Pearl 1988) have received much attention in representing probability distributions. BN provide an intuitive and comprehensible way of representing probability distributions as it provides an easy way to express probabilistic dependencies. However, inference with these representations is known to be computationally hard—inference in BN is #P-complete

Editors: Aleksander Kołcz, Dunja Mladenić, Wray Buntine, Marko Grobelnik, and John Shawe-Taylor.

D. Roth · R. Samdani (✉)
Department of Computer Science, University of Illinois at Urbana-Champaign, Illinois, USA
e-mail: rsamdan2@illinois.edu

D. Roth
e-mail: danr@illinois.edu

(Roth 1996). Although there has been much work on inference techniques in BN like conditioning (Darwiche 2001), variable elimination (Shachter et al. 1980; Dechter 1996; Zhang and Poole 1996), jointree based approaches (Pearl 1988; Jensen et al. 1990), and arithmetic circuits based techniques (Darwiche 2003), inference in BN still remains a hard problem in practice. To alleviate this problem, several approximate inference techniques have been developed like Gibbs sampling (Gilks et al. 1995), variational inference (Wainwright and Jordan 2008), and loopy belief propagation (Yedidia et al. 2005). While there are several cases in which these methods give accurate and fast results, providing performance and accuracy guarantees in general is difficult.

In this paper, we present a representation of distributions over categorical variables which facilitates fast inference and learning. We represent the distribution explicitly in the form of a multi-linear polynomial which provides exact inference in time linear in its size, and develop algorithms that directly learn the distribution in the multi-linear form.

1.1 Related work

Most of the recent work using probability distributions to support inference focuses on *learning a model* from the data rather than explicitly constructing the distributional representation. Since most of the works that makes use of representations of probability distributions for inference is concerned with graphical models, this has been the focus also of the learning work. In most cases, however, the resulting representation is still hard to make inferences with. While the use of graphical models was initially motivated by comprehensibility, this issue seems less important when the representation is learned from data, unless a restricted distributional representation is learned.

Initial works in learning BN (Heckerman et al. 1995) penalize the number of edges and parameters to simplify the resulting representation and avoid overfitting, but this doesn't directly affect the complexity of inference. A large class of approaches for generating BN which provide efficient inference rely on either mixture models (Meila and Jordan 2000) (limited representation capacity) or thin tree width networks (Srebro 2003) (computationally hard for graphs with large tree widths). Recently, there has been work by Lowd and Domingos (2008) which learns a Bayes net while directly penalizing the complexity of the associated arithmetic circuit. Another approach for representing probability distributions which particularly takes benefit of context specific independence is based on Probabilistic Decision Graphs (PDG) (Jaeger et al. 2006). However, similar to BN, PDG are not guaranteed to be efficient in the most general cases.

Given that probabilistic inference with MLR is efficient, in this paper, instead of using graph-based representations, we study the problem of directly learning multi-linear polynomial distributions. The class MLR of multilinear probability distributions has already been explored by Castillo et al. (1996, 1997) who show that a probability distribution represented over BN can be represented as a multi-linear polynomial over network parameters and use MLR for symbolic probabilistic inference in BN. Darwiche studies (e.g., Darwiche 2003) arithmetic circuits as an alternative representation for BN, and computes a multi-linear polynomial called the *Network Polynomial* for the distribution in order to facilitate inference.

All these works consider MLR as an intermediate representation within the BN framework, whereas we consider it independently as a powerful and expressive class. Since inference in MLR has a direct linear relation with the size of the polynomial, we present techniques to learn bounded-size MLR distributions. Moreover, to the best of our knowledge, this is the first attempt at directly learning distributions as MLR.

2 Preliminaries

Consider the following setting: we have a set of n random variables $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$ such that the variable X_i takes labels from a finite set $S_i = [\kappa_i]$ where $[\kappa_i]$ is the set of all natural numbers from 1 to κ_i. Now, any distribution $P(\mathcal{X})$ over $\mathcal{X}$ can be trivially written as

$$P(\mathcal{X}) = \sum_{s_1 \in S_1, s_2 \in S_2, \ldots, s_n \in S_n} p_{s_1,s_2,\ldots,s_n} \prod_{j=1}^{n} \mathcal{I}_{X_j = s_j} \tag{1}$$

where $p_{s_1,s_2,\ldots,s_n} = Pr(X_1 = s_1, X_2 = s_2, \ldots, X_n = s_n)$ and $\mathcal{I}_{X_j=s_j}$ is an indicator variable such that

$$\mathcal{I}_{X_j=s_j} = \begin{cases} 1 & \text{if } X_j = s_j, \\ 0 & \text{otherwise.} \end{cases}$$

In the above form of representing $P(\mathcal{X})$, each term contains exactly one indicator variable corresponding to each $X_i \in \mathcal{X}$. Also, the size of each term in this representation is n and the number of terms is exponential in n. This is an explicit way of writing the probability distribution over these variables and any probability distribution over discrete variables (including BN) can be expressed in this form.

We define the *Multi-Linear Representation*, MLR, of a probability distribution as a multinomial over the indicator variables $\mathcal{I}_{X_i=s_i}$ such that each term in the multinomial has at most one indicator variable corresponding to any variable $X_i \in \mathcal{X}$. This form of representation is called multi-linear representation because it is linear in terms of indicator variables of any variable $X_i \in \mathcal{X}$ if the remaining variables are kept fixed. The distribution defined in (1) is clearly represented as an MLR thus demonstrating that MLR is a universal representation (Darwiche 2003).

A distribution D over $\mathcal{X}$ can be specified as an MLR as follows. Let $D = (R, C)$, where $R = \{r_1, r_2, \ldots, r_t\}$ is the collection of the terms (i.e. monomials) in the polynomial and $C = \{c_1, c_2, \ldots, c_t\} \in \mathcal{R}^t$ is the set of the coefficients of the terms (later on we'll see that not any arbitrary set of coefficients from $\mathcal{R}^t$ works for a multi-linear polynomial to be a valid distribution). We can specify each term r_i as $r_i = \prod_{j=1}^{n} \mathcal{I}_{X_j=s_{ij}}$, where $s_{ij} \in S_i \cup \{0\}$ (0 being a dummy state, with $s_{ij} = 0$ implying that r_i doesn't depend on X_j) and $\mathcal{I}$ is an indicator function such that

$$\mathcal{I}_{X_j=s_{ij}} = \begin{cases} 1 & \text{if } s_{ij} = 0 \text{ or } X_j = s_{ij}, \\ 0 & \text{otherwise.} \end{cases}$$

Thus the probability distribution $P_D(\mathcal{X})$ can be specified as

$$P_D(\mathcal{X}) = \sum_{i=1}^{t} c_i r_i = \sum_{i=1}^{t} c_i \prod_{j=1}^{n} \mathcal{I}_{X_j=s_{ij}}. \tag{2}$$

In case each variable in $\mathcal{X}$ is Boolean, that is, the domain of each $X_i \in \mathcal{X}$ is $\{0, 1\}$, we can write any distribution as an MLR in an even simpler way as

$$P_D(\mathcal{X}) = \sum_{i=1}^{t} c_i \prod_{j=1}^{n} X_j^{s_{ij}} \left(1 - X_j\right)^{s'_{ij}}$$

where $s_{ij}, s'_{ij} \in \{0, 1\}$ and $s_{ij} + s'_{ij} \leq 1$. Thus P_D can in fact be written as a multi-linear polynomial over the variables themselves. For example $P(X) = \frac{1}{14}(1 + x_1 + x_2 - x_1x_3)$ is an MLR distribution over Boolean random variables X_1, X_2, X_3. Under this distribution, the probabilities of the events $\{X_1 = 1, X_2 = 0, X_3 = 1\}$ and $\{X_1 = 0, X_2 = 1, X_3 = 1\}$ are $\frac{1}{14}$ and $\frac{1}{7}$, respectively.

3 Representational issues: validity and compactness

In this section, we consider two important questions regarding MLR. First, pertaining to the usefulness of the representation: how compactly can a distribution be represented in MLR? Second, pertaining to the learnability: what constraints must a multi-linear polynomial satisfy in order to be a probability distribution?

3.1 Compactness

As mentioned in the last section, the MLR representation of some distribution may require exponentially many terms. However, several interesting families of distributions can be compactly represented as MLR.

Consider, for example, the class of all distributions with terms of size equal to k. Clearly the number of such terms is bounded by $\binom{n}{k}m^k$ where m is the maximum cardinality over the domain sets of variables. In case of Boolean variables, it is bounded by $\binom{n}{k}2^k$. Thus if k is sub-log n then any distribution in this class has size polynomial in n. Note that this representation does not have any independence or conditional independence assumptions. In fact, there exist polynomial-size distributions in this class in which no conditional independence assumption between any subsets of $\mathcal{X}$ hold.

To illustrate the above point we consider a distribution defined over a set of Boolean random variables $\mathcal{X}$ as

$$G(\mathcal{X}) = \frac{\sum_{i=1}^{n} X_i}{n2^{n-1}}.$$

It is easy to see that G is a valid probability distribution as it is non-negative for all instances and the sum of probabilities over all instances is 1. Now consider the conditional probability distribution of X_n given $X_1 = x_1, X_2 = x_2, \ldots, X_k = x_k$. We get that

$$\begin{aligned}
&\Pr(X_n = x_n | X_1 = x_1, X_2 = x_2, \ldots, X_k = x_k) \\
&= \frac{\Pr(X_n = x_n, X_1 = x_1, X_2 = x_2, \ldots, X_k = x_k)}{\Pr(X_1 = x_1, X_2 = x_2, \ldots, X_k = x_k)} \\
&= \frac{x_n + \left(\sum_{i=1}^{k} x_i\right) + (n - 1 - k)\, 2^{n-1}}{\left(\sum_{i=1}^{k} x_i\right) + (n - k)\, 2^{n-1}} = 1 - \frac{2^{n-1} - x_n}{\left(\sum_{i=1}^{k} x_i\right) + (n - k)\, 2^{n-1}}.
\end{aligned}$$

Clearly the above function is not independent of the value of k - it varies as k changes. Since the distribution is symmetric we can claim that the distribution of any variable in $\mathcal{X}$ conditioned on a subset of $\mathcal{X} - \{X_n\}$ varies with the size of that subset. In other words, conditioned on any $X_1 \subset \mathcal{X} - \{X_n\}$, X_n is not independent of any $X_2 \subseteq \mathcal{X} - X_1 - \{X_n\}$, $X_2 \neq \phi$. This, in particular, implies that the corresponding BN representation in its moralized form is a complete graph and inference is thus exponential in complexity. Hence we notice that

there are distributions which can be represented compactly in MLR but have no efficient representation in BN-based approaches.

Theorem 1 *An MLR representation of a probability distribution can be exponentially more compact than the corresponding Bayesian Network representation.*

3.2 Validity

This subsection is important for learning in MLR. Consider a distribution $P_D(\mathcal{X})$ represented as MLR:

$$P_D(\mathcal{X}) = \sum_{i=1}^{t} c_i r_i = \sum_{i=1}^{t} c_i \prod_{j=1}^{n} \mathcal{I}_{X_j = s_{ij}}.$$

In this form, for P_D to represent a valid probability distribution, the coefficients cannot take arbitrary values. In particular, given a multi-linear polynomial P, it is easy to see that P is a valid distribution *iff* it satisfies

$$P(x) \geq 0 \quad \forall x \in S_1 \times S_2 \times \cdots \times S_n \tag{3}$$

$$\text{and} \quad \sum_{x \in S_1 \times S_2 \times \cdots \times S_n} P(x) = 1. \tag{4}$$

For the purpose of learning, we would want the above two properties to be easily verifiable or "imposable". Given any multi-linear polynomial, constraint (4) is easy to impose via normalization. To see this, consider w.l.o.g. the case when $\mathcal{X}$ consists of Boolean random variables. We are given a multi-linear polynomial:

$$P(x) = \sum_{i=1}^{t} c_i \prod_{j=1}^{n} x_j^{b_{ij}} \left(1 - x_j\right)^{b'_{ij}}$$

with $b_{ij}, b'_{ij} \in \{0, 1\}$ and $b_{ij} + b'_{ij} \leq 1$. Condition (4) yields the constraint

$$\sum_{x \in \{0,1\}^n} P(x) = 1 \Rightarrow \sum_{i=1}^{t} c_i 2^{n - \sum_{j=1}^{n} (b_{ij} + b'_{ij})} = 1.$$

To impose this condition, we can simply divide P by a normalization constant $Z = \sum_{i=1}^{t} c_i 2^{n - \sum_{j=1}^{n} (b_{ij} + b'_{ij})}$ which is easy to compute. Thus, the normalization condition can be verified or imposed in time linear in the size of $P(x)$. However, verifying the positivity constraint (3) is not trivial:

Theorem 2 *Verifying positivity for an arbitrary multi-linear polynomial is NP-hard.*

Proof We prove this theorem by reducing 3-SAT to positivity verification. The 3-SAT problem can be stated as: Given a set C of clauses defined over a set of boolean variables $\mathcal{X}$ such that the size of each clause in C is no greater than 3, does there exist an assignment for variables in $\mathcal{X}$ such that it satisfies all the clauses in C? W.l.o.g., we can assume that each clause contains at most one literal corresponding to each variable in $\mathcal{X}$. Consider $C' = \{\overline{c} | c \in C\}$, the set of terms obtained by negating all the clauses in C. So 3-SAT can be equivalently

posed as: Does there exist an instantiation of variables in $\mathcal{X}$ that does not satisfy any of the terms in C'? Now consider a polynomial $P_{C'}$ defined as

$$P_{C'} = \left(\sum_{c' \in C'} c' \right) - \frac{1}{2}$$

where we treat the *and* operation in the terms in C' as a *product*—clearly this leaves the value of each term unchanged. $P_{C'}$ is a multi-linear polynomial over $\mathcal{X}$ as each term contains at most one literal corresponding to each variable in $\mathcal{X}$.

We claim that $P_{C'}$ does not satisfy positivity *iff* C is satisfiable. To see this, observe that if $P_{C'}$ satisfies positivity then it must be that for all possible instantiations of X, at least one of the terms in C' must be satisfied implying that the corresponding clause in C is not satisfied and thus C is not satisfiable. For the other way, if $P_{C'}$ doesn't satisfy positivity that is there exists an instance $x \in \mathcal{X}$ such that $P_{C'}(x) < 0$ then it must be that x doesn't satisfy any of the terms in C' and thus satisfies all the clauses in C.

Since $P_{C'}$ can be constructed in time polynomial in the size of C' (and hence C), the above implies that we have obtained a polynomial time reduction of 3-SAT to positivity verification. This proves that the latter is also NP-Hard. □

The above implies that learning a multi-linear distribution with arbitrary coefficients is a hard problem. One can alleviate this problem by restricting C to the set $(\mathcal{R}^+ \cup \{0\})^t$ thus trivially satisfying constraint (3). However, by imposing this restriction we lose out on compactness as it may also expand the structure. For example with this restriction, we must express the distribution $\frac{1}{3}(1 - x_1 x_2)$ over Boolean variables x_1 and x_2 as $\frac{1}{3}((1 - x_1)(1 - x_2) + x_1 + x_2)$. In this paper we assume the coefficients to be positive during learning. The resulting blow-up of the MLR is discussed further in a forthcoming paper. Since in the proof of Theorem 1, the example has positive coefficients, the compactness result still holds for MLR distributions with positive coefficients.

4 Inference

In the next two sections, for the sake of simplicity and w.l.o.g., we assume that all the random variables in $\mathcal{X}$ are Boolean unless otherwise mentioned. We briefly review the marginal and conditional inference techniques in MLR and show that marginal and conditional inference queries over a multi-linear distribution can be answered in time linear in the size of the distribution. For details, the reader can refer to Castillo et al. (1996), Darwiche (2003).

Consider the task of computing a marginal distribution over a set of variables $\mathcal{X}' = \{X_1, X_2 \ldots, X_k\} \subseteq \mathcal{X}$. The marginal distribution $P_D(\mathcal{X}')$ over $\mathcal{X}'$ can be obtained by summing over the remaining variables as:

$$\begin{aligned} P_D(\mathcal{X}') &= \sum_{\mathcal{X}/\mathcal{X}'} \sum_{i=1}^{t} c_i \prod_{j=1}^{n} X_j^{s_{ij}} \left(1 - X_j\right)^{s'_{ij}} \\ &= \sum_{i=1}^{t} c_i 2^{n - |\mathcal{X}'| - \sum_{j=k+1}^{n} (s_{ij} + s'_{ij})} \prod_{j=1}^{k} X_{ij}^{s_{ij}} \left(1 - X_{ij}\right)^{s_{ij}}. \end{aligned}$$

Clearly, the marginal distribution obtained has degree smaller than the original distribution and hence is also a multi-linear polynomial. Moreover, it can be obtained from the original distribution in time $O(t * n)$ where t is the number of terms in P_D and n is the number of variables in $\mathcal{X}$ (i.e. linearly in the size of $P_D(\mathcal{X})$).

Given evidence e about $\mathcal{X}' \subseteq \mathcal{X}$, denote by $P_D(\mathcal{X}|\mathcal{X}' = e)$ the distribution obtained by substituting $\mathcal{X}' = e$ in $P_D(\mathcal{X})$. Procedure for obtaining the conditional distribution follows easily from that of marginal distribution. Given evidence e about $\mathcal{X}_1 \subseteq \mathcal{X}$ and a set $\mathcal{X}_2 \subseteq \mathcal{X}/\mathcal{X}_1$, let $P_D(\mathcal{X}_2|\mathcal{X}_1 = e)$ be the conditional distribution of $\mathcal{X}_2$ given $\mathcal{X}_1 = e$. We can write

$$P_D(\mathcal{X}_2|\mathcal{X}_1 = e) = \frac{P(\mathcal{X}_2 \cup \mathcal{X}_1|\mathcal{X}_1 = e)}{P(\mathcal{X}_1|\mathcal{X}_1 = e)}.$$

$P(\mathcal{X}_2 \cup \mathcal{X}_1)$ and $P(\mathcal{X}_1)$ can be found in time $O(t * n)$ as described above and hence the conditional distribution $P_D(\mathcal{X}_2|\mathcal{X}_1 = e)$ can be computed in $O(t * n)$ time. Also, since $P(\mathcal{X}_1|\mathcal{X}_1 = e)$ is a function of only e, $P_D(\mathcal{X}_2|\mathcal{X}_1 = e)$ is an MLR for a given e.

5 Learning

We now describe the key contribution of the paper. Learning includes two basic components: learning the structure of the distribution that is learning which terms are present in the MLR, and learning the coefficients of each term.

5.1 Learning the terms

One trivial way to write the terms of a distribution is to consider all possible terms of size n; however, this clearly results in an exponentially large distribution. Our task is to perform density-estimation and thus we approximate the structure of the distribution with bounded number of terms. One way to restrict the number of terms is to restrict the structure to consist only of terms of size k. This assumption results in polynomial number of terms with respect to n for a small enough k and the question is—how good a representation this can be and how to learn it.

Another way of restricting the number of terms is to use some prior expertise to provide relevant terms. This expertise could be human provided or can be generated using heuristic methods. In this paper, we describe one particularly useful heuristic approach for estimating the terms in the distribution based on the idea of *frequent patterns* from data-mining. First, we consider a few definitions from the data-mining literature.

Given a set of items It and a database $DB = \{Record|Record \subseteq It\}$, the support of any subset $It' \subseteq It$ can be defined as

$$support(It') = |\{Record|Record \in DB, It' \subseteq Record\}|.$$

Given a support threshold, $supp$, we define the set of frequent pattern $Freq(DB, supp)$ over DB as

$$Freq(DB, supp) = \{It' \subseteq It|support(It') \geq supp\}.$$

However, to avoid redundancy between terms it makes more sense in our case to pick the largest possible frequent patterns satisfying the support constraint rather than all possible frequent patterns. We define:

$$Max\text{-}Freq(DB, supp) = \{It'|support(It') \geq supp, \nexists It'' \supset It', support(It'') \geq supp\}.$$

However the problem of finding max-frequent patterns is NP-hard (Gunopulos et al. 2003) and thus we instead use *closed patterns* defined as:

$$Closed(DB, supp) = \{It' | support(It') \geq supp, \nexists It'' \supset It', support(It'') = support(It')\}.$$

For more details on frequent itemset mining, the interested reader can refer to Agrawal et al. (1993), Pasquier et al. (1999), Burdick et al. (2005). Essentially, we consider the frequently occurring patterns in the data as a good predictor of the underlying distribution and use them to estimate the distribution in the MLR form. To summarize this approach, we convert the given training data into a database of items (ensuring each feature value becomes a distinct item) and from that mine closed patterns with high enough support. We then use the resulting closed patterns to obtain the terms of the MLR distribution. The selected terms are likely to be sufficient to represent the probability mass of the observed examples and support generalization (as we only mine the most frequent patterns). The crucial part here is to pick the right threshold such that the obtained set of terms is not too large and is yet expressive enough.

5.2 Exact learning of the coefficients

Having estimated the set of terms R of the distribution D, the next step is to learn the "most likely" set of coefficients C^*, given the constraints. We are given a set of m training examples $Y = \{Y^1, Y^2, \ldots, Y^m\}$ which are instances over Boolean variables. The distribution as MLR looks like

$$P(X; C) = \sum_{i=1}^{t} c_i \prod_{j=1}^{n} X_j^{s_{ij}} \left(1 - X_j\right)^{s'_{ij}}.$$

Our task is to learn the maximum likelihood estimate for C. Thus assuming that the samples are drawn independently, we maximize

$$P(Y|C) = \prod_{l=1}^{m} P\left(X = Y^l | C\right) = \prod_{l=1}^{m} \left(\sum_{i=1}^{t} c_i \prod_{j=1}^{n} \left(Y_j^l\right)^{s_{ij}} \left(1 - Y_j^l\right)^{s'_{ij}} \right),$$

subject to the following constraints on the coefficients:

$$c_i \geq 0, \quad \forall i$$

and the normalization constraint:

$$\sum_{x \in \{0,1\}^n} \sum_{i=1}^{t} c_i \prod_{j=1}^{n} x_j^{s_{ij}} (1 - x_j)^{s'_{ij}} = 1.$$

Assuming that the term r_i has k_i literals, the normalization constraint implies

$$\sum_{i=1}^{t} 2^{n-k_i} c_i = 1.$$

To simplify the notation, lets define constants $q_{i,l} = (Y_j^l)^{s_{ij}}(1 - Y_j^l)^{s'_{ij}}$. So overall, the problem of learning MLE coefficients, after taking log of the objective function, is:

$$\begin{aligned} \max \quad & \sum_{l=1}^{m} \log\left(\sum_{i=1}^{t} q_{i,l} c_i\right), \\ \text{subject to} \quad & c_i \geq 0, \quad \forall i \\ \text{and} \quad & \sum_{i=1}^{t} 2^{n-k_i} c_i = 1. \end{aligned}$$

This is a constrained convex programming problem, since the objective function (which needs to be maximized) is concave and the constraints are linear. Consequently, any local maximum for this problem is guaranteed to be a global maximum. Now, we can use standard convex optimization software to compute the MLE set of coefficients C^*. We denote the set of coefficients thus obtained as an Exact-MLR model.

5.3 An approximation algorithm for learning the coefficients

The exact learning procedure mentioned above can be slow in converging. Moreover, most convex optimization techniques require expensive Hessian computations at each stage. We therefore present a technique for efficiently computing an approximate solution by maximizing a lower bound on the objective function. We call this solution an Approx-MLR model. We evaluate this technique as a standalone model and also use it to initialize the exact optimization procedure described above (for faster convergence).

For now, we assume that $c_i > 0,\ \forall i$ (since if $c_i = 0$ for some term r_i then we can ignore that term anyway). Later on we'll see that this assumption is actually satisfied for the solution obtained in this part. Consider the log-likelihood objective function from the previous part

$$LL = \sum_{l=1}^{m} \log\left(\sum_{i=1}^{t} q_{i,l} c_i\right).$$

We define a quantity $Q(l) = \sum_i^t q_{i,l}$ which is the number of terms satisfied by the lth data instance Y_l. Rewrite LL as

$$LL = \sum_{l=1}^{m} \log\left(\sum_{i=1}^{t} \frac{q_{i,l}}{Q(l)} c_i\right) + \sum_{l=1}^{m} \log Q(l). \tag{5}$$

Using the fact that log is a concave function, we rewrite (5) and obtain

$$LL \geq \sum_{l=1}^{m} \sum_{i=1}^{t} \frac{q_{i,l}}{Q(l)} \log c_i + \sum_{l=1}^{m} \log Q(l) = LL'.$$

The plan is as follows: instead of maximizing LL we maximize a lower bound on LL, namely, LL'. In fact, observe that LL is the same as LL' if all the terms are distinct and of size n. The constraints on the coefficients remain the same. We ignore the $\log Q(l)$ term in LL' as it is a constant and thus we have a different optimization problem:

$$\max \quad \sum_{l=1}^{m} \sum_{i=1}^{t} \frac{q_{i,l}}{Q(l)} \log c_i,$$

$$\text{subject to} \quad c_i \geq 0, \quad \forall i,$$

$$\text{and} \qquad \sum_{i=1}^{t} 2^{n-k_i} c_i = 1.$$

This is again a convex optimization problem but the solution to this can be obtained in closed form. To solve this we introduce Lagrange multipliers $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_t\}$ and μ and write down the Lagrangian of the above problem as

$$L(C, \Lambda, \mu) = -\sum_{l=1}^{m} \sum_{i=1}^{t} \frac{q_{i,l}}{Q(l)} \log c_i + \mu \left(\sum_{i=1}^{t} 2^{n-k_i} c_i - 1 \right) + \sum_{i=1}^{t} \lambda_i c_i.$$

Now we impose the KKT conditions which imply that at one of the optimum values c_i', λ_i', and μ' the following is satisfied

$$\lambda_i' c_i' = 0, \quad \forall i,$$

$$2^{n-k_i} c_i' = 1,$$

$$\text{and} \quad \frac{\partial L(C, \Lambda, \mu)}{\partial c_i} = 0, \quad \forall i.$$

Using the fact that $c_i > 0$, we manipulate the above equations to obtain:

$$c_i' = \frac{2^{k_i} \sum_{l=1}^{m} \frac{q_{i,l}}{Q(l)}}{m 2^n}. \tag{6}$$

One can observe that c_i' is roughly linearly dependent on the number of training examples satisfied by the term and a training example makes greater contribution to c_i' if it isn't satisfied by too many terms (i.e. $Q(l)$ is low). This is intuitive in that since in the MLE process we assume that each training example has equal weightage, if an example satisfies less number of terms then the terms satisfied by that example must carry greater weight. Also this result is clearly in accord with the assumption that $c_i > 0$, because c_i' is zero only when r_i is satisfied by none of training examples—in that case we are better off omitting r_i. We show experimentally that Approx-MLR solution works reasonably well in practice.

5.4 Summary of learning techniques

To summarize, following are steps involved in learning MLR distributions, given the training data Y and set of variables $\mathcal{X}$:

- First step is generating the terms, R, of the MLR. The terms can be generated either by mining closed patterns, with appropriate threshold t, from an itemized version of Y and converting them into terms, or by producing all monomials over $\mathcal{X}$ of size k. In the latter approach, prune away infrequently satisfied terms if necessary and perform smoothing by adding a constant term (the former approach vacuously includes this term as the empty set has full support in a database).
- Generate the Approx-MLR solution for the coefficients (which can be computed in one pass over Y).
- Initialize the exact optimization process with the Approx-MLR solution. Carry out optimization for a certain no. of iterations to obtain the Exact-MLR solution. Output (R, Exact-MLR) and (R, Approx-MLR) as different learned MLR distributions.

6 Experimental results

We compare the class of multi-linear representation of distributions with Bayesian networks on real-world datasets. We show results both for Exact-MLR and Approx-MLR. As a baseline, we use the WinMine toolkit (Chickering et al. 2002) which is commonly used in the ML community as one of the best BN learning software available.

6.1 Datasets

We evaluate our learning techniques on 3 real-world datasets taken from the UCI repository (Asuncion and Newman 2007): Primary Tumor, House Votes, and Letter Recognition. Although, these datasets are meant for the purpose of classification, learning joint distribution over these variables is still a good problem and they have been used in previous works on learning distributions (Lowd and Domingos 2005). Moreover, these datasets are indeed generated from randomly occurring distributions in that the training examples are not contrived to train a classifier. Table 1 contains a summary of the datasets used. We randomly split the data into training and test set in 9:1 ratio. For the datasets with small number of examples i.e. House Votes and Primary Tumor, we perform 10-fold cross validation (after randomizing the order of the instances in the data). For tuning the parameters, we divide the training data into tuning and validation sets in the ratio 8:2. Since WinMine treats Missing values as distinct values, we do the same for MLR.

6.2 Learning

To generate the terms of the distribution we use the two approaches we have described in Sect. 5: picking terms of fixed size $k = 3$ (call it K-3), and picking terms based on closed-patterns with certain support threshold. We use the ILLIMINE package (http://illimine.cs.uiuc.edu/) for mining closed patterns. For fast inference and learning, we restrict the number of terms to 7,000. We tune the support threshold parameter for closed-patterns based approach on the validation set. To generate varying number of terms, we pick 3 different thresholds based on the validation process. We call the model with lowest threshold (and hence the largest number of terms) F-1, the one with medium threshold, F-2, and the one with highest threshold, F-3. F-1, F-2, and F-3 correspond to thresholds: 80, 100, and 120 for Primary Tumor; 300, 400, and 500 for Letter Recognition; and 90, 100, and 110 for House Votes. Since the number of terms for $k = 3$ is extremely large in case of the Letter Recognition dataset (because of high cardinality for most variables), we do not try K-3 approach on it. For the rest of the datasets, we reduce the number of terms in the K-3 case by ordering the terms according to the number of training examples they satisfy and pruning away a certain fraction of the lower order terms. We set the pruning fraction to the value which results in high accuracy in the validation step, while keeping as less number of

Table 1 Description of the datasets used. Max. Cardinality is the maximum over the number of values any variable can take. Avg. Cardinality is the average over the number of values for features

Dataset	Dimension	No. of Instances	Max. Cardinality	Avg. Cardinality
Letter Recognition	17	20,000	26	16.6
House Votes	17	435	3	2.94
Primary Tumor	18	339	21	3.5

terms as possible. For Primary Tumor and House Votes, we prune 80% and 70% of size-3 terms, respectively. Also, we smooth the K-3 case by introducing a constant term.

For each of the term structures learned, we learn the set of coefficients as described in Sect. 5. We learn the APPROX-MLR model directly using the entire training data. For the purpose of learning Exact-MLR solution we use the software CVX-OPT (http://abel.ee.ucla.edu/cvxopt/). For the Exact-MLR model, allowing the optimization process to carry on till convergence may result in overfitting. We tune the number of iterations in the optimization process to the value which results in highest joint likelihood during the validation step and then learn the model over the entire training data. We initialize the optimization process by using the APPROX-MLR solution for faster learning. However in the case of Letter Recognition, the likelihood declines from the very first iteration and so we restart the process with a uniform distribution.

For learning the corresponding WinMine model, we perform similar steps. We use the default parameter settings for WinMine except for the per-parameter penalty κ. We use the tuning and validation data to tune κ, picking the best performing value from the set $\{0.001, 0.01, 0.1, 0.5, 1.0\}$. We then use that value to learn the final model from the entire training data.

Learning in WinMine was only slightly faster than APPROX-MLR. However learning in EXACT-MLR was considerably slower than WinMine. It took Winmine less than two minutes to learn each model whereas EXACT-MLR took from a few tens of minutes (in case of approx. 1500 terms) to a couple of hours (in case of approx. 6000 terms) to converge. However, since learning is performed offline, it is much more important to have faster and accurate inference even if it comes at the cost of a slower learning process.

6.3 Accuracy of learned models

We measure the accuracy of models as the average joint log-likelihood of the test data. For the House Votes and Primary Tumor dataset, we report the average of log-likelihood obtained over the 10-folds. The results are shown in Table 2. MLR-based approaches outperform WinMine on Primary Tumor and Letter Recognition dataset. On the latter, MLR's performance is significantly better than WinMine as per a two-tailed paired t-test with $p = 0.05$. The difference between the two is negligible in case of House Votes.

6.4 Performance of learned model on inference

To compare the performance of learned models for each dataset, we generate inference queries using the test data in a way similar to (Lowd and Domingos 2008). Specifically, for

Table 2 Joint log-likelihood for test data per example. A-MLR represents Approx-MLR, E-MLR represents Exact-MLR, and W-M represents WinMine. F-1, F-2, and F-3 represent closed frequent pattern based approaches with thresholds increasing in that order. K-3 represents learning with distributions having terms of size 3

Dataset	F-1		F-2		F-3		K-3		W-M
	E-MLR	A-MLR	E-MLR	A-MLR	E-MLR	A-MLR	E-MLR	A-MLR	
Pri. Tum.	**−13.86**	−15.05	−14.15	−15.22	−14.34	−15.40	−15.23	−16.37	−19.55
Let. Rec.	−44.14	**−43.22**	−45.03	−44.26	−44.90	−44.37	–	–	−52.27
Hou. Vot.	−12.99	−14.57	−13.55	−14.98	−13.88	−15.35	−16.22	−16.82	−12.97

an instance from the test data, we generate query and evidence variables and then calculate the log probability of the configuration of the query variables given the evidence variables, as per the learned model. To generate such queries from a test instance, we pick each variable independently as a query variable with probability p_q or as an evidence variable with probability p_e. We finally compute average joint log-likelihood over the instances. This approximates, within a constant factor, the Kullback-Leibler divergence between the learned and the true distribution, as per the test data. We generate queries from the entire test data for each fold in case of Primary Tumor and House Votes. For Letter Recognition, we randomly sample 500 test instances to generate inference queries. We measure the inference time for all the methods on a machine with CentOS and 16 GB RAM, running at 2.33 GHz.

For MLR-based approaches, we use exact inference. For WinMine, we use Gibbs sampling to perform approximate inference. Since Gibbs sampling takes a long time to converge, we try different settings: fast sampling (using 10 chains, 100 burn-in iterations, and 100 sampling iterations), slow sampling (using 10 chains, 100 burn-in iterations, and 1000 sampling iterations), and very slow sampling (using 10 chains, 1000 burn-in iterations, and 1000 sampling iterations).

Table 3 shows the time taken per query for different settings and datasets. Since the inference time doesn't depend on coefficients, we report the average time taken for different approaches for generating terms.

As far as inference accuracy is concerned, we didn't observe much variation in the performance of the models within both the classes: MLR and BN. In case of Primary Tumor, Freq-1 + Exact-MLR gives best performance. For Letter Recognition, Freq-1 + Approx-MLR gives best performance. For House Votes, Gibbs-very-slow is the best performer with Gibbs-slow coming very close. Table 4 reports log-likelihoods for the best performing models for MLR and BN, averaged over all queries and over all values of p_q which varies over the set $\{0.3, 0.4, 0.5, 0.6\}$, while p_e is fixed at 0.3. Similarly in Table 5, p_q is fixed at 0.3 and p_e takes values from $\{0.3, 0.4, 0.5, 0.6\}$.

Table 3 This table reports the average time taken (in ms) for answering queries corresponding to each dataset. F-1, F-2, and F-3 represent closed frequent pattern based approaches with thresholds increasing in that order. K-3 represents learning with distributions having terms of size 3. G-v-s, G-S, and G-f represent very slow, slow, and fast Gibbs sampling

Dataset	F-1	F-2	F-3	K-3	G-v-s	G-s	G-f
Pri. Tum.	135	67	35	103	803	446	80
Let. Rec.	72	37	22	–	11,118	6,013	1,165
Hou. Vot.	85	46	25	91	644	356	64

Table 4 This table reports the best performing model for both MLR and BN and its performance. The values reported are average joint log-likelihoods with fixed $p_e = 0.3$

Dataset	MLR		BN	
	Best Model	log-like. +/− std.	Best Model	log-like. +/− std.
Pri. Tum.	F-1+Exact-MLR	$\mathbf{-6.23 +/- 0.44}$	Gibbs-very-slow	$-7.06 +/- 0.74$
Let. Rec.	F-1+Approx-MLR	$\mathbf{-19.93 +/- 0.24}$	Gibbs-slow	$-28.08 +/- 0.28$
House Votes	F-1 +Exact-MLR	$-5.77 +/- 0.36$	Gibbs-very-slow	$\mathbf{-4.82 +/- 0.56}$

Table 5 This table reports the best performing model for both MLR and BN and its performance. The values reported are average joint log-likelihoods with fixed $p_e = 0.3$

Dataset	MLR		BN	
	Best Model	log-like. +/− std.	Best Model	log-like. +/− std.
Pri. Tum.	F-1+Exact-MLR	$\mathbf{-4.22}$	Gibbs-very-slow	-4.48
Let. Rec.	F-1+Exact-MLR	$\mathbf{-13.60 +/- 0.23}$	Gibbs-slow	$-20.43 +/- 0.27$
House Votes	F-1 +Exact-MLR	$-3.84 +/- 0.3$	Gibbs-very-slow	$\mathbf{-3.12 +/- 0.39}$

Inference for best performing model in MLR is an order of magnitude faster than the best performing model in WinMine in case of Primary Tumor and House Votes, and is more than 100 times faster in case of Letter Recognition. Moreover, MLR models outperform WinMine in Primary Tumor and Letter Recognition, with the performance being significantly better ($p = 0.05$) on the latter. BN-based approaches outperform MLR-based ones on House Votes.

The results obtained in the inference experiments can be attributed to the fact that BN fits the training data more accurately when the number of states for each variables is less (which is the case in House Votes) but performs much worse as the cardinality of each variable becomes high (like in Letter Recognition). Intuitively this happens because larger number of possible states for each variable results in fewer counts for each state leading to inaccurate estimates in BN. The MLR models we learned, on the other hand, typically have smaller terms which means they exhibit better generalization. This is because smaller terms are satisfied by a larger number of possible instantiations thus avoiding overfitting. This generalization property proves beneficial when the possible number of instantiations is large but results in slower fitting to the training data.

7 Conclusion

In this paper, we presented techniques for directly learning distributions in the multi-linear polynomial form to support faster inference. Experiments on real-world datasets suggest that our techniques for learning can generate MLR models which are equally or more accurate than the corresponding ones in BN, with the former providing orders of magnitude faster inference. Interesting directions for future work include trying or developing approaches for generating terms with high descriptive power and low redundancy and developing learning techniques for MLR to better fit the training data without losing the good generalization properties we currently show.

Acknowledgements The authors wish to thank Daniel Lowd for his help with the Gibbs sampling program. This work is partly supported by an ONR Award on "Guiding Learning and Decision Making in the Presence of Multiple Forms of Information" and by the Siebel Scholars Foundation.

References

Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *SIGMOD '93, 1993* (pp. 207–216).

Asuncion, A., & Newman, D. (2007). UCI Machine learning repository.

Burdick, D., Calimlim, M., Flannick, J., Yiu, T., & Gehrke, J. (2005). MAFIA: A maximal frequent itemset algorithm. *IEEE Transactions of Knowledge Data Engineering*, *17*, 1490–1504.

Castillo, E., Gutiérrez, J. M., & Hadi, A. S. (1996). Goal oriented symbolic propagation in Bayesian networks. In *AAAI/IAAI 1996* (Vol. 2, pp. 1263–1268).

Castillo, E., Gutiérrez, J. M., Hadi, A. S., & Solares, C. (1997). Symbolic propagation and sensitivity analysis in Gaussian Bayesian networks with application to damage assessment. *AI in Engineering*, *11*(2), 173–181.

Chickering, D. M. (2002). *The WinMine Toolkit* (Tech. Rep. MSR-TR-2002-103). Microsoft, Redmond, WA.

Darwiche, A. (2001). Recursive conditioning. *Artificial Intelligence*, *126*(1–2), 5–41.

Darwiche, A. (2003). A differential approach to inference in Bayesian networks. *Journal of the ACM*, *50*(3), 280–305.

Dechter, R. (1996). Bucket elimination: A unifying framework for probabilistic inference. In *UAI 1996* (pp. 211–219).

Gilks, W. R., Richardson, S., & Speigelhalter, D. J. (1995). *Markov chain Monte Carlo in practice*. Boca Raton: Chapman & Hall/CRC.

Gunopulos, D., Khardon, R., Mannila, H., Saluja, S., Toivonen, H., & Sharm, R. S. (2003). Discovering all most specific sentences. *ACM Transactions on Database Systems*, *28*(2), 140–174.

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, *20*(3), 197–243.

Jaeger, M., Nielsen, J. D., & Silander, T. (2006). Learning probabilistic decision graphs. *International Journal of Approximate Reasoning*, *42*(1–2), 84–100.

Jensen, F. V., Lauritzen, S., & Olesen, K. (1990). Bayesian updating in recursive graphical models by local computation. *Computational Statistics Quarterly*, *4*, 269–282.

Lowd, D., & Domingos, P. (2005). Naive Bayes models for probability estimation. In *Proc. ICML-05* (pp. 529–536).

Lowd, D., & Domingos, P. (2008). Learning arithmetic circuits. In *UAI 2008* (pp. 383–392).

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

Pasquier, N., Bastide, Y., Taouil, R., & Lakhal, L. (1999). Discovering frequent closed itemsets for association rules. In: *ICDT'99, 1999* (pp. 398–416).

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems*. San Mateo: Morgan Kaufman.

Roth, D. (1996). On the hardness of approximate reasoning. *Artificial Intelligence*, *82*(1–2), 273–302.

Shachter, R. D., D'Ambrosio, B., & Favero, B. D. (1980). Symbolic probabilistic inference in belief networks. In *AAAI 1990* (pp. 126–131).

Srebro, N. (2003). Maximum likelihood bounded tree-width Markov networks. *Artificial Intelligence*, *143*(1), 123–138.

Wainwright, M. J., & Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, *1*(1–2), 1–305.

Yedidia, J. S., Freeman, W.T., Weiss, Y., (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, *51*(7), 2282–2312.

Zhang, N. L., & Poole, D. (1996). Exploiting causal independence in Bayesian network inference. *Journal Artificial Intelligence Research (JAIR)*, *5*, 301–328.