Decision Sum-Product-Max Networks

Mazen Melibari [§], **Pascal Poupart** [§], **Prashant Doshi** [‡]
[§] David R. Cheriton School of Computer Science, University of Waterloo, Canada
[‡] Dept. of Computer Science, University of Georgia, Athens, GA 30602, USA
[§] {mmelibar,ppoupart}@uwaterloo.ca, [‡]pdoshi@cs.uga.edu

Abstract

Sum-Product Networks (SPNs) were recently proposed as a new class of probabilistic graphical models that guarantee tractable inference, even on models with high-treewidth. In this paper, we propose a new extension to SPNs, called Decision Sum-Product-Max Networks (Decision-SPMNs), that makes SPNs suitable for discrete multi-stage decision problems. We present an algorithm that solves Decision-SPMNs in a time that is linear in the size of the network. We also present algorithms to learn the parameters of the network from data.

Introduction

Influence diagrams (IDs) are well-known probabilistic graphical models for multi-stage decision problems. IDs extend Bayesian Networks with decision and utility nodes. As the case with most probabilistic graphical models, solving IDs is NP-hard even in networks with bounded treewidth (Mauá, de Campos, and Zaffalon 2012). Sum-Product Networks (SPNs) (Poon and Domingos 2011) were recently proposed as a new class of probabilistic graphical models that guarantees tractable inference. Several variants of SPNs have been developed in the recent years, including Relational-SPNs (Nath and Domingos 2015) for relational models, and Dynamic-SPNs (**?**) for data with varying lengths. In this paper, we propose a new extension to SPNs, called Decision Sum-Product-Max Networks (Decision-SPMNs), that makes it suitable for discrete multi-stage decision problems. The next section formally defines the proposed model and presents an algorithm to learn the parameters from data.

Decision Sum-Product-Max Networks

Decision-SPMNs are built upon the framework of SPNs. Definition 1 extends the definition of SPNs (Poon and Domingos 2011) by introducing two new types of nodes: decision and utility nodes.

Definition 1 (Decision-SPMN)**.** A Decision-SPMN over decision variables $d_1, ..., d_m$ and random variables $x_1, ..., x_n$ is a rooted directed acyclic graph. Its leaves are either indicators of the random variables $x_1, ..., x_n$ or utility

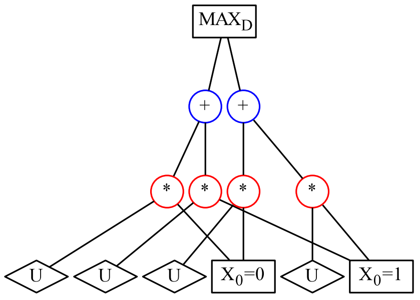

Figure 1: Example Decision-SPMN with one decision and one random variable. Parameters and edge labels are omitted for clearity.

nodes. The internal nodes of an SPMN are Sum, $Product$, or Max nodes. Each Max node corresponds to one of the decision variables $d_1, ..., d_m$ and each outgoing edge from a Max node is labeled with one of the possible values of the corresponding decision variable. The value of a Max node is $\max_{j \in Ch(i)} v_j$, where i is the Max node, $Ch(i)$ is the set of children of i, and v_j is the value of node j. The Sum and $Product$ nodes are defined as in regular SPNs.

We now turn to recall the concepts of information sets and partial ordering. The information sets $I_0, .., I_n$ are disjoint subsets of the random variables such that the random variables in the information set I_{i-1} are observed before the decision associated with variable d_i is made. The partial order $\prec$ is the ordering of the information sets followed by the decision variables, $I_0 \prec d_1 \prec I_1 \prec d_2 \prec ... \prec d_n \prec I_n$.

Next, we define a set of properties to ensure that a Decision-SPMN encodes a function that computes the maximum expected utility (MEU) for some partial order $\prec$ and some utility function U.

Definition 2 (Completeness of Sum Nodes)**.** A Decision-SPMN is complete *iff* all children of the same sum node have the same scope, where the scope is the set of variables that are included in a child.

Definition 3 (Decomposability of Product Nodes)**.** A Decision-SPMN is decomposable iff no variable appears in more than one child of a product node.

Definition 4 (Completeness of Max Nodes)**.** A Decision-

SPMN is max-complete *iff* all children of the same max node have the same scope, where the scope is the set of decision variables that are included in a child.

Definition 5 (Uniqueness of Max Nodes)**.** A Decision-SPMN is max-unique iff each max node that corresponds to a decision variable d appears at most once in every path from root to leaves.

We can obtain the maximum expected utility of a decision problem that has the partial order $\prec$ and utility function U using the Sum-Max-Sum rule, in which we alternate between summing over the variables in an information set and maximizing over a decision variable. The next definition makes a connection between Decision-SPMNs and the Sum-Max-Sum rule. We use the notion $S(e)$ to indicate the value of a Decision-SPMN when evaluated at evidence e.

Definition 6. A Decision-SPMN S is valid *iff* $S(e) = \mathrm{MEU}(\mathrm{e}| \prec, \mathrm{U})$

Figure 1 shows an example of a Decision-SPMN over a decision variable, D, and a random variable, X_0. Solving a Decision-SPMN can be done by setting the indicators that are compatible with the evidence to 1 and the rest to 0, then performing a bottom-up pass on the network. The optimal strategy can be found by tracing back the network and choosing the edges that maximize the decision nodes.

Parameters Learning

Let $\mathcal{D}$ be a dataset that consists of $|\mathcal{D}|$ instances, where each instance, $\mathcal{D}_i$, is a tuple of the values of observed random variables, $\mathbf{X}$, the values of decision variables, $\mathbf{D}$, and a single utility value, u, that represents the utility of the joint assignment of values for $\mathbf{X}$ and $\mathbf{D}$; i.e. $\mathcal{D}_i = \langle \mathbf{X}, \mathbf{D}, U(\mathbf{X}, \mathbf{D}) \rangle$. Algorithm 1 gives an overview of the parameter learning process. The process is split into two subtasks: 1) Learning the values of the utility nodes, 2) Learning the embedded probability distribution. The following sections describe these two sub-tasks in detail.

```
Algorithm 1: Decision-SPMN Parameters Learning
  input  : S: Decision-SPMN, $\mathcal{D}$: Dataset
  output: Decision-SPMN with learned parameters
  $S \leftarrow \mathrm{learnUtilityValues}(\mathrm{S}, \mathcal{D})$;
  $S \leftarrow \mathrm{decisionSpmnEM}(\mathrm{S}, \mathcal{D})$;
```

Learning the Values of the Utility Nodes The first subtask is to learn the values of the utility nodes. We start by introducing the notion of *specific-scope*. The *specific-scope* for an indicator node is the value of the random variable that the indicator represents; for all other nodes the *specific-scope* is the union of their childrens' *specific-scopes*. For example, an indicator node, $\mathbb{I}_x$, for $X = x$ has the *specific-scope* $\{x\}$, while an indicator node, $\mathbb{I}_{\bar{x}}$, for $X = \bar{x}$ has the *specific-scope* $\{\bar{x}\}$. A sum node over $\mathbb{I}_x$ and $\mathbb{I}_{\bar{x}}$ has the *specific-scope* $\{x, \bar{x}\}$. A product node that has two children, one with *specific-scope* $\{x, \bar{x}\}$ and another one with *specific-scope* $\{y\}$, will have the *specific-scope* $\{x, \bar{x}, y\}$. A simple algorithm that performs a bottom-up pass and propagates the *specific-scope* of each node to its parents can be used to define the *specific-scope* of all the nodes in a SPMN.

For each unique instance $\mathcal{D}_i$ in $\mathcal{D}$ we perform a top-down pass, where we follow all the nodes that have values consistent with $\mathcal{D}_i$ in their *specific-scope*. If we reach a utility node, then we set its value to the utility value in $\mathcal{D}_i$.

Learning the Embedded Probability Distribution The second sub-task is to learn the parameters of the embedded probability distribution. In particular, we seek to learn the weights of the sum nodes. This can be done using a special derivation of the Expectation-Maximization (EM) algorithm that is suitable for SPMNs. For each instance, $\mathcal{D}_i$, in the dataset, we set the indicators to their values in $\mathbf{X}_i$ (the observed values of the random variables in instance $\mathcal{D}_i$). We then perform inference by evaluating the SPMN using a bottom-up pass. In order to integrate the decisions, $\mathbf{D}$, for instance $\mathcal{D}_i$, each max node will multiply the value of its children with either 0 or 1 depending on the value of the corresponding decision in the instance. This multiplication is equivalent to augmenting the SPMN with indicators for the max nodes. And since we are only concerned with the weights of the sum nodes in this sub-task, all the utility nodes can be treated as hidden variables with fixed probability distributions, where summing them out will always result in the value 1. We also perform a top-down pass to compute the gradient of the nodes. The expected counts of each child of a sum node is maintained using a counter for each child. We normalize and assign those values to the sum nodes at the end of each iteration. This process is repeated until convergence.

Conclusion and Future Work

We proposed a new extension to SPNs, called Decision Sum-Product-Max Networks (Decision-SPMNs), that makes SPNs suitable for discrete multi-stage decision problems. Solving Decision-SPMNs can be done in a time that is linear in the size of the network. We also presented an algorithm to learn the parameters of the network from data. An important direction for future work is to develop an efficient structure learning algorithm for Decision-SPMNs. Another future work is to experimentally evaluate how Decision-SPMNs and alternative discrete multi-stage decision models can perform on real-life datasets.

References

Mauá, D. D.; de Campos, C. P.; and Zaffalon, M. 2012. The complexity of approximately solving influence diagrams. In *Twenty-Eighth Conference on Uncertainty in Artificial Intelligence (UAI-12)*, 604–613. AUAI Press.

Melibari, M.; Poupart, P.; and Doshi, P. 2015. Dynamic Sum Product Networks for Tractable Inference on Sequence Data. *ArXiv e-prints*.

Nath, A., and Domingos, P. 2015. Learning relational sum-product networks. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Poon, H., and Domingos, P. 2011. Sum-product networks: A new deep architecture. In *Proc. 12th Conf. on Uncertainty in Artificial Intelligence*, 2551–2558.