Sum-Product-Max Networks for Tractable Decision Making (Extended Abstract)

Mazen Melibari
University of Waterloo
Waterloo, Ontario, Canada
mmelibar@uwaterloo.ca

Pascal Poupart
University of Waterloo
Waterloo, Ontario, Canada
ppoupart@uwaterloo.ca

Prashant Doshi
University of Georgia
Athens, GA, USA
pdoshi@cs.uga.edu

1. INTRODUCTION

Influence diagram (ID) has been the graphical language of choice for probabilistically modeling decision-making problems [5]. IDs extend the probabilistic inference of Bayesian networks with decision and utility nodes to allow the computation of expected utility and decision rules. However, unlike Bayesian networks that have witnessed a rich portfolio of algorithms to automatically learn their structure from data, no algorithms exist to the best of our knowledge for learning the structure and parameters of IDs from data.

Recent investigations into new models for tractable probabilistic inference such as arithmetic circuits [2] and sum-product networks [4] that are suited to learn models from large datasets could help fill this gap. Specifically, approaches to directly learn a network polynomial that is graphically represented as a network of sum and product nodes from data have been devised [3]. Evaluations of the polynomial provide the joint or conditional distributions as desired and are performed in time that is *linear* in the size of the network. Thus, arithmetic circuits and sum-product networks represent a tractable class of inference models compared to the generally intractable inferencing of Bayesian networks.

Motivated by tractable inference, we generalize sum-product networks to a new class of problems that involve probabilistic decision making, in this paper. To enable this, we introduce two new types of nodes: *max* nodes to represent the maximization operation over different possible values of a decision variable, and *utility* nodes to represent the utility values. We refer to the resulting network as a sum-product-max network (SPMN), whose solution provides a decision rule that maximizes the expected utility in linear time. The semantics of the max node is that its output is the decision that leads to the maximal value among all decisions. Analogously to sum-product networks, we introduce a set of properties that guarantee the validity of the SPMN, such that the solution will correspond to the expected utility obtained from a valid embedded probabilistic model and utility function that are encoded by the network.

We present a first method to learn the structure and parameters of valid SPMNs from decision-theoretic data. Such data not only consists of instances of the random state variables but also possible decision(s) and the corresponding valuation(s). This is a significant advance because it brings machine learning to decision making, which has so far relied on handcrafted expert models.

2. SUM-PRODUCT-MAX NETWORKS

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), J. Thangarajah, K. Tuyls, C. Jonker, S. Marsella (eds.), May 9–13, 2016, Singapore.*

2.1 Definition and Solution

SPMNs generalize SPNs [4] by introducing two new types of nodes to an SPN: max and utility nodes. We begin by defining an SPMN.

DEFINITION 1 (SPMN). *An SPMN over decision variables $D_1, \ldots, D_m$ and random variables $X_1, \ldots, X_n$ is a rooted directed acyclic graph. Its leaves are either binary indicators of the random variables or utility nodes. An internal node of an SPMN is either a sum, product or max node. Each max node corresponds to one of the decision variables 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 i is $\max_{j \in Children(i)} v_j$, where $Children(i)$ is the set of children of i, and v_j is the value of the subgraph rooted at child j. The sum and product nodes are defined as in the SPN.*

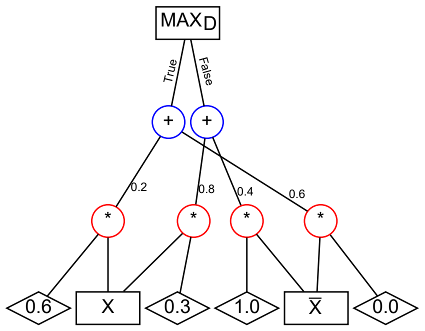

Figure 1: An SPMN for one decision and random variable.

Figure 1 shows a generic example SPMN for a decision-making problem with a single decision D_1 and binary random variable X_1. Indicator nodes $X_1 = T$ and $X_1 = F$ return a 1 and 0 respectively, when the random variable X_1 is true, and vice versa if X_1 is false.

Next, we define a set of properties to ensure that a SPMN encodes a function that computes the maximum expected utility (MEU) given some partial order between the variables and utility function U.

DEFINITION 2 (COMPLETENESS OF SUM NODES). *An SPMN is sum-complete iff all children of the same sum node have the same scope.*

The scope of a node is the set of all random variables associated with indicators and decision variables associated with max nodes that appear in the SPMN rooted at that node.

DEFINITION 3 (DECOMPOSABILITY OF PRODUCT NODES). *An SPMN is decomposable iff no variable appears in more than one child of a product node.*

DEFINITION 4 (COMPLETENESS OF MAX NODES). *An SPMN is max-complete iff all children of the same max node have the same scope, where the scope is as defined previously.*

DEFINITION 5 (UNIQUENESS OF MAX NODES). *An 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.*

Together, these properties allow us to define a valid SPMN.

DEFINITION 6 (VALIDITY). *A SPMN is valid if every sum node in the network is complete, every product node is decomposable, and each max node is complete and unique.*

Evaluation An SPMN is evaluated by setting the indicators that are consistent with the evidence to 1 and the rest to 0. Then, we perform a bottom-up pass of the network during which operators at each node are applied to the output values of its children. The optimal decision rule is found by tracing back (i.e., top-down) through the network and choosing the edges that maximize the decision nodes.

2.2 Structure Learning

Our method for learning SPMNs labeled as LearnSPMN generalizes LearnSPN [3], which is a recursive top-down learning method for SPNs. This allows automated learning of computational models of decision-making problems from appropriate data. LearnSPMN extends LearnSPN to generate the two new types of nodes introduced in SPMNs: max and utility nodes. Equally important, the generalization also requires modifying a core part of LearnSPN so that the learned structure respects the constraints that are imposed by the partial order $\mathcal{P}^{\prec}$ on variables involved in the decision problem. Algorithm 1 describes the structure-learning method.

```
Algorithm 1: LearnSPMN
input  : $\mathcal{D}$:data, $\mathbf{V}$: set of variables, $i$:infoset index, $\mathcal{P}^{\prec}$: partial order
output :
if $|\mathbf{V}| = 1$ then
    if the variable $V$ in $\mathbf{V}$ is a utility then
        $u \leftarrow estimate \Pr(V = True)$ from $\mathcal{D}$;
        return a utility node with the value $u$
    else
        return smoothed univariate distribution over $V$
else
    $rest \leftarrow \mathcal{P}^{\prec}[i+1...]$;
    if $\mathcal{P}^{\prec}[i]$ is a decision variable then
        for $v \in$ decision values of $\mathcal{P}^{\prec}[i]$ do
            $\mathcal{D}^v \leftarrow$ subset of $\mathcal{D}$ where $\mathcal{P}^{\prec}[i] = v$
        return $\mathrm{MAX}_v$ LearnSPMN($\mathcal{D}^v$, rest, i + 1, $\mathcal{P}^{\prec}$)
    else
        Try to partition $\mathbf{V}$ into independent subsets $\mathbf{V}_j$ while keeping
          $rest$ in one partition;
        if a partition is found then
            return $\prod_j$ LearnSPMN($\mathcal{D}$, $\mathbf{V}_j$, i, $\mathcal{P}^{\prec}$)
        else
            partition $\mathcal{D}$ into clusters $\mathcal{D}^j$ of similar instances;
            return $\sum_j \frac{|\mathcal{D}^j|}{|\mathcal{D}|} \times$ LearnSPMN($\mathcal{D}^j$, $\mathbf{V}$, i, $\mathcal{P}^{\prec}$)
```

LearnSPMN takes as input a dataset $\mathcal{D}$ and a partial order $\mathcal{P}^{\prec}$. Each utility variable in the data is first converted into a binary random variable, say U, independent from other utility variables by using the well-known Cooper transformation [1]. Specifically, $\Pr(U = true|Parents(U)) = \frac{u-u_{min}}{u_{max}-u_{min}}$ where u_{min} and u_{max} are the minimum and maximum values for that utility variable and $Parents(U)$ is a joint assignment of the variables that U depends on. This step has an added benefit in that it allows for stochastic utility functions as well such as in bandit problems. Next, we duplicate each instance a fixed number of times and replace the utility value of each instance by an i.i.d. sample of *true* or *false* from the corresponding distribution over U. Consequently, utility variables may be treated as traditional random variables in the learning method. Algorithm 1 iterates through the partial order $\mathcal{P}^{\prec}$. For each decision variable D, a corresponding max node is created. For each set $\mathcal{V}$ of random variables in an information set of the partial order, the algorithm constructs an SPN of sum and product nodes by recursively partitioning the random variables in non-correlated subsets and by partitioning the data into clusters of similar instances. As in LearnSPN, LearnSPMN can be implemented using any suitable method such as a pairwise χ^2 or G-test to partition the variables and instances. Clustering algorithms such as EM and K-means can be used to partition the data into clusters of similar instances.

3. EXPERIMENTS

We evaluate LearnSPMN by applying it to a small test bed of 3 data sets whose attributes consist of state and decision variables and corresponding utility values. These datasets were created by simulating a randomly generated directed acyclic graph of nodes whose conditional probability tables and utility tables were populated by values from symmetric Dirichlet distributions. Table 1 gives some descriptive statistics for these data sets.

Dataset	#Dec_var	|ID|	|Dataset|	|SPMN|
Random-ID 1	3	116	100K	730
Random-ID-2	5	283	100K	922
Random-ID 3	8	580	100K	2940

Table 1: Synthetic datasets and learned model statistics. #Dec_var is the number of decisions variables in the problem, |ID| is the total representational size of the ID (total clique size + sepsets), |Dataset| is the size of the dataset, and |SPMN| is the size of the learned SPMN.

The last column of Table 1 reports on the size of the SPMN that was learned for each dataset. While the size is usually larger than the total representational complexity of the corresponding ID, we emphasize that the run time complexity of SPMN is *linear* in the size of the network. For the top two datasets, SPMN yields an identical MEU as that from the IDs and the third dataset exhibits a 10.3% difference in the MEU from the SPMN.

Acknowledgments

This research is supported in part by grants from NSERC and ONR #N000141310870.

REFERENCES

[1] G. F. Cooper. A method for using belief networks as influence diagrams. 1998.

[2] A. Darwiche. A differential approach to inference in bayesian networks. In *UAI*, pages 123–132, 2000.

[3] R. Gens and P. Domingos. Learning the structure of sum-product networks. In *Proceedings of The 30th International Conference on Machine Learning*, pages 873–880, 2013.

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

[5] R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34(6):871–882, 1986.