ID-SPN: A New SPN Learner

Pedram Rooshenas  <pedram@cs.uoregon.edu>  Daniel Lowd  <lowd@cs.uoregon.edu>

HIGHLIGHTS

ID-SPN is a new algorithm for learning tractable joint probability distributions over discrete domains.

ID-SPN combines the strength of tractable graphical models, which easily represent direct interaction among random variables, with mixture models, which easily represent clusters.

ID-SPN learns more accurate models than state-of-the-art learning algorithms for Bayesian networks and sum-product networks.

Key Ideas

A tractable Markov network is a powerful model for learning joint probability distributions and shows promising performance on most experimental datasets, where data does not contain natural clusters [Lowd&Rooshenas, 2013].

Mixture of trees is a powerful model, which learns a mixture over joint probability distribution represented by Chow-Liu trees [Meila&Jordan, 2000].

Our primary results show that mixture of arithmetic circuits is a more accurate model by having a mixture over joint probability distributions represented by tractable Markov networks.

Sum-product network is a powerful graphical model which defines hierarchical mixtures over univariate probability distributions [Poon&Domingos, 2011; Gens&Domingos, 2013].

ID-SPN learns a hierarchical mixture model over multivariate joint probability distributions represented by tractable Markov networks.

The model learned by ID-SPN is a valid SPN model.

Algorithm

ID-SPN learns a sum-product of ACs (SPAC) structure.

ACMN learns Markov Networks (MNs) in which exact inference is tractable. ACMN uses arithmetic circuits (ACs) as the inference representation. ACMN uses the size of the arithmetic circuit as a learning bias, so it only learns an MN if it has a compact representation as an AC. ACMN performs a greedy search through structure space. Structures are scored according to both log-likelihood and AC size:

\[ \text{Score}(s) = \Delta L(s) - \gamma \Delta AC(s) \]

ACMN learns Markov Networks (MNs) in which exact inference is tractable. ACMN uses arithmetic circuits (ACs) as the inference representation. ACMN uses the size of the arithmetic circuit as a learning bias, so it only learns an MN if it has a compact representation as an AC. ACMN performs a greedy search through structure space. Structures are scored according to both log-likelihood and AC size:

\[ \text{Score}(s) = \Delta L(s) - \gamma \Delta AC(s) \]

ID-SPN is a greedy iterative algorithm.

ID-SPN learns a sum-product of ACs (SPAC) structure. An SPAC structure includes sum nodes, product nodes, and AC nodes.

ID-SPN is a greedy iterative algorithm.

The ID-SPN algorithm begins with an SPAC structure that only contains one AC node. In each iteration, ID-SPN attempts to extend the model by replacing one of the AC leaf nodes with a new SPAC subtree over the same variables.

Each extension adds at most one product node followed by at most one sum node for each child.

If the extension increases the log-likelihood of the SPAC model on the training data, then ID-SPN updates the working model and adds any newly created AC leaves to an extension queue.

To learn a product node, we partition variables into some sets that are approximately independent. We find the independent sets by locating the connected components of the graph whose nodes have edges if their pairwise mutual information is greater than a threshold.

To learn a sum node, we partition training data by using expectation-maximization to learn a simple naïve Bayes mixture model over the set of variables.

To learn an AC node, we used the ACMN algorithm.

Any SPAC structure is a valid SPN.

Learning SPNs

Learning SPNs [Gens&Domingos, 2013] performs a top-down clustering, and it creates the SPN directly through recursive partitioning of variables and instances, which results in a hierarchical mixture model structure.

ID-SPN learns a hierarchical mixture model over multivariate joint probability distributions represented by tractable Markov network.

The model learned by ID-SPN is a valid SPN model.

Sum-product networks (SPNs)

SPNs can represent complex mixture models, low-treewidth graphical models, and arithmetic circuits, a very similar representation often used for exact inference in graphical models.

An SPN is either a univariate distribution, a product of SPNs with disjoint scopes, or a weighted sum of SPNs with identical scopes.

Sum-product networks are a powerful graphical model which defines hierarchical mixing models over multivariate joint probability distributions.

Mixture of trees is a powerful model, which learns a mixture over joint probability distribution represented by Chow-Liu trees. Like an SPN, an AC is a rooted, directed, acyclic graph in which interior nodes are sums and products.

For discrete domains, every decomposable and smooth AC can be represented as an equivalent SPN with fewer nodes and edges. Every SPN also can be represented as an AC with at most a linear increase in the number of edges.

Arithmetic circuits (ACs)

ACs (Darwiche,2003) are an inference representation that is closely related to SPNs.

We used the authors’ implementation of LTM, SPN, ACMN and WM, all the hyper parameters are fine-tuned on validation sets.

- LTM [Choi et al., 2011]: learns latent tree models.
- MT [Choi et al., 2011]: learns latent tree models.
- WM (Choi et al., 2011): learns latent tree models.
- ACMN uses arithmetic circuits (ACs) as the inference representation.

- WM (WinMine) [Chickering, 2002]: learns intractable Bayesian networks.
- LTM [Choi et al., 2011]: learns latent tree models.

We computed the log-likelihood of models on test data and report the number of datasets in which ID-SPN has a statistically significantly different log-likelihood.

ID-SPN is significantly more accurate than WinMine on 95 out of 100 configurations and only significantly worse than WinMine on 1 configuration.

We also computed the exact conditional log-likelihood (CLL) of query variables given evidence (\( \log P(X = x | E = e) \)) for both ID-SPN. For WinMine, we computed both CLL and conditional marginal log-likelihood (CMLL), and used whichever is larger.

We used 10, 30, 50, and 90 percentage of variables as a query with the rest of them as evidence (Xevidence) for both ID-SPN. For WinMine, we computed both CL and conditional marginal log-likelihood (CMLL), and used whichever is larger.

We used 10, 30, 50, and 90 percentage of variables as a query with the rest of them as evidence (Xevidence) for both ID-SPN. For WinMine, we computed both CL and conditional marginal log-likelihood (CMLL), and used whichever is larger.

ID-SPN is significantly more accurate than WinMine on 95 out of 100 configurations and only significantly worse than WinMine on 1 configuration.

We also computed the exact conditional log-likelihood (CLL) of query variables given evidence (\( \log P(X = x | E = e) \)) for both ID-SPN. For WinMine, we computed both CL and conditional marginal log-likelihood (CMLL), and used whichever is larger.

In the experiments, we computed both CLL and CMLL, and used whichever is larger.

We used 10, 30, 50, and 90 percentage of variables as a query with the rest of them as evidence (Xevidence) for both ID-SPN. For WinMine, we computed both CL and conditional marginal log-likelihood (CMLL), and used whichever is larger.

ID-SPN is significantly more accurate than WinMine on 95 out of 100 configurations and only significantly worse than WinMine on 1 configuration.

Acknowledgement

This research was partly funded by ARO grant W911NF-08-1-0242 and NSF grant OCI-0960354. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO, NSF, or the U.S. Government.