Discriminative Structure Learning of Arithmetic Circuits

Amirmohammad Rooshenas and **Daniel Lowd**
Department of Computer and Information Science
University of Oregon
Eugene, OR 97401, USA
{rooshena,lowd}@uoregon.edu

Abstract

The biggest limitation of probabilistic graphical models is the complexity of inference, which is often intractable. An appealing alternative is to use tractable probabilistic models, such as arithmetic circuits (ACs) and sum-product networks (SPNs), in which marginal and conditional queries can be answered efficiently. In this paper, we present the first discriminative structure learning algorithm for ACs, DACLearn (Discriminative AC Learner), which optimizes conditional log-likelihood. Based on our experiments, DACLearn learns models that are more accurate and compact than other tractable generative and discriminative baselines.

Introduction

Probabilistic graphical models such as Bayesian and Markov networks have been widely used for knowledge representation and reasoning in diverse domains such as natural language processing, coding theory, data mining, computational biology, computer vision, information extraction, and social networks. However, the problem of inference limits their effectiveness and broader applicability: in many real-world problems, exact inference is intractable and approximate inference can be unreliable and inaccurate. This poses difficulties for parameter and structure learning as well, since most learning methods rely on inference.

A compelling alternative is to work with model classes where inference is efficient, such as bounded treewidth models (Chechetka and Guestrin 2008), mixtures of tractable models (Meila and Jordan 2000), sum-product networks (SPNs) (Poon and Domingos 2011; Gens and Domingos 2013), and arithmetic circuits (ACs) (Lowd and Domingos 2008; Lowd and Rooshenas 2013).

ACs and SPNs are closely related representations that represent probability distributions as computation graphs, consisting of the sums and products necessary to answer any probability query. Marginal and conditional probability queries can be computed in linear time in the number of edges in their graphs.

In discrete domains, ACs can be translated into SPNs in linear time and space, and vice versa (Rooshenas and Lowd 2014). We focus on ACs as our representation since they are a slightly better fit to our methods. Compared to Bayesian and Markov networks, SPN and AC structures are often very difficult to specify by hand. In response, several structure learning algorithms have been developed.

However, most work on learning the structure of SPNs, ACs, and other tractable models has focused exclusively on generative models, which represent a full joint distribution over all random variables. In many problem domains, some variables are always known at query time and can be assumed to be evidence. For example, in most language modeling tasks, the words in the document are observed, and in most computer vision problems, the pixels in the image are known. Discriminative models represent the conditional distribution $P(\mathbf{Y}|\mathbf{X})$ of the query variables $\mathbf{Y}$ given evidence $\mathbf{X}$, which allows them to avoid the difficult and expensive task of modeling a full joint distribution over the evidence, $P(\mathbf{X})$.

In this paper, we present DACLearn (Discriminative AC Learner), a flexible and powerful method for learning tractable discriminative models.

DACLearn

DACLearn builds on methods for learning tractable Markov networks (ACMN algorithm) (Lowd and Rooshenas 2013). Like ACMN, DACLearn searches through a combinatorial space of log-linear models with conjunctive features, such as $x_4 = \text{true} \wedge y_3 = \text{false}$. DACLearn begins with an initial AC structure and set of candidate features. In each iteration, DACLearn scores candidate features based on how much they increase the model accuracy, penalized by how much they increase the size of the AC. The highest-scoring features are added to the AC, and new features are added to the set of candidate features.

The key difference between ACMN and DACLearn is that ACMN optimizes log-likelihood while DACLearn optimizes conditional log-likelihood (CLL):

$$CLL(D) = \sum_{i \in D} \sum_{j} (\theta_j f_j(\mathbf{x}_i, \mathbf{y}_i) - \log Z(\mathbf{x}_i)), \qquad (1)$$

where f_j is a conjunctive feature over the state of a subset of query and evidence variables, and θ_j is the feature weight. Z is the partition function which depends on evidence $\mathbf{x}_i$.

Since DACLearn is learning a conditional distribution, $P(\mathbf{Y}|\mathbf{X})$, it does not need to represent a valid probability

Copyright © 2016, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Table 1: Number of DACLearn wins/ties/losses compared to other baselines with 50% and 80% of variables are evidence.

	IDSPN	EACMN	CACMN	MCTBN
50%	10/5/5	18/0/2	15/2/3	12/2/3
80%	15/3/2	17/3/0	14/4/0	14/6/0

distribution over $\mathbf{X}$. This makes the structural modifications required for conditioning on an evidence variable very minor, so that conditioning on evidence is cheap. The AC must still support efficient marginalization of query variables, so complex interactions among query variables may still lead to large ACs.

Algorithm 1 shows the high-level pseudo-code of the DACLearn algorithm. The biggest challenge is that scoring candidate models is expensive because each example has a different partition function. To overcome this, DACLearn uses novel heuristics for initializing the structure and selecting candidate features. The initial structure is based on a Chow-Liu tree over the query variables, with additional features connecting query variables to evidence.

Algorithm 1 DACLearn

```
fs ← initial candidate features.
C ← AC representing initial structure.
fh ← feature max heap based on data support.
do
    for f in fs do
        s ← $\Delta_{cll}(f) - \gamma\Delta_{edges}(f)$ //$\gamma$ is the complexity penalty.
        if s > feature penalty then //to avoid overfitting.
            Update C to represent the previous features plus f.
            Joint parameter optimization.
            fh.push(f)
        end if
    end for
    ftop ← fh.pop()
    if ftop = ∅ then Stop.
    fs ← generate candidate features using ftop.
until Stop
return C
```

Experiments

We run our experiments using 20 datasets with 16 to 1556 binary-valued variables, which also used by Gens and Domingos (2013) and Rooshenas and Lowd (2014).

To observe the performance of discriminative structure learning in the presence of variable number of query variables, we create two versions of these 20 datasets. In one, we label, randomly chosen 50% of variables as evidence variables and the other half as query variables. We create the other version of the datasets by randomly selecting 80% of variables as evidence variables while the remaining 20% are representing query variables.

For baselines, we compare to a state-of-the-art generative SPN learner (Rooshenas and Lowd 2014), IDSPN, and a generative AC learner, EACMN, which is the original ACMN algorithm (Lowd and Rooshenas 2013) improved with DACLearn's initialization and search heuristics. To show the effect of discriminative parameter learning, we also take the best models learned by EACMN, based on the average log-likelihood on validation data, and optimize their parameters in order to maximize the conditional log-likelihood (CLL) of $P(\mathbf{Y}|\mathbf{X})$. We call this method conditional ACMN (CAMCN). Finally, we choose MCTBN (Hong, Batal, and Hauskrecht 2014) as our last baseline, which learns a mixture of conditional tree Bayesian networks.

For all of the above methods, we learn the model using the training data and tune the hyper-parameters using the validation data, and we report the average CLL over the test data. To tune the hyper-parameters, we used a grid search over the hyper-parameter space. We bounded the learning time of all methods to 24 hours.

Table 1 summarizes our experiments by showing the number of time that DACLearn is significantly better (win), not different (tie), or worse (loss) than the baselines based on the average CLL in the presence of 50% and 80% evidence variables. Wins and losses are determined by two-tailed paired t-tests ($p < 0.05$) on the test set CLL. MCTBN and CACMN reach the 24 hour limit on some datasets, so we exclude those experiments. DACLearn is significantly more accurate than our baselines on at least half the experiments.

Table 1 shows that discriminative parameter learning (CACMN) is less accurate than discriminative structure learning (DACLearn).

As we mention earlier, discriminative ACs are more compact than generative ACs. The average size of the circuits learned by IDSPN and EACMN are 2.2M and 1.1M edges, respectively, while the average size of the circuits learned by DACLearn is 55K edges when we have 50% evidence variables and 22K edges when we have 80% evidence variables. This means that inference in discriminative ACs is 100 times faster than IDSPN when we have 80% evidence variables.

Acknowledgments

This research was supported by NSF grant IIS-1451453 and a Google Faculty Research Award.

References

Chechetka, A., and Guestrin, C. 2008. Efficient principled learning of thin junction trees. In *NIPS'08*.

Gens, R., and Domingos, P. 2013. Learning the structure of sum-product networks. In *ICML'13*.

Hong, C.; Batal, I.; and Hauskrecht, M. 2014. A mixtures-of-trees framework for multi-label classification. In *CIKM'14*.

Lowd, D., and Domingos, P. 2008. Learning arithmetic circuits. In *UAI'08*.

Lowd, D., and Rooshenas, A. 2013. Learning Markov networks with arithmetic circuits. In *AISTATS'13*.

Meila, M., and Jordan, M. I. 2000. Learning with mixtures of trees. *Journal of Machine Learning Research* 1:1–48.

Poon, H., and Domingos, P. 2011. Sum-product networks: A new deep architecture. In *UAI'11*.

Rooshenas, A., and Lowd, D. 2014. Learning sum-product networks with direct and indirect variable interactions. In *ICML'14*.