Learning Tractable Graphical Models Using Mixture of Arithmetic Circuits

Amirmohammad Rooshenas and **Daniel Lowd**
Department of Computer and Information Science
University of Oregon
Eugene, OR 97403
{pedram,lowd}@cs.uoregon.edu

Abstract

In recent years, there has been a growing interest in learning tractable graphical models in which exact inference is efficient. Two main approaches are to restrict the inference complexity directly, as done by low-treewidth graphical models and arithmetic circuits (ACs), or introduce latent variables, as done by mixtures of trees, latent tree models, and sum-product networks (SPNs). In this paper, we combine these approaches to learn a mixtures of ACs (MAC). A mixture can represent many distributions exponentially more compactly than a single AC. By using ACs as mixture components, MAC can represent complex distributions using many fewer components than required by other mixture models. MAC generalizes ACs, mixtures of trees, latent class models, and thin junction trees, and can be seen as a special case of an SPN. Compared to state-of-the-art algorithms for learning SPNs and other tractable models, MAC is consistently more accurate while maintaining tractable inference.

Introduction

Probabilistic graphical models such as Bayesian and Markov networks have been widely used in many areas of AI, including computer vision, natural language processing, robotics, and more. Their biggest limitation is that inference is intractable: in general, computing the exact probability of a marginal or conditional query is #P-complete. However, there are many special cases of graphical models where inference is tractable. These include graphical models with low treewidth, graphical models with other kinds of structure such as determinism or context-specific independence, and any mixture of tractable models. For problems where the quality and efficiency of inference are important, performing exact inference in a tractable model may be better than performing approximate inference in an intractable model, even if the intractable model fits the data slightly better.

In this paper, we focus on the problem of learning a tractable graphical model from data, with the goal of answering probabilistic queries accurately and efficiently. There are two key challenges to doing this. First, some relationships are difficult to represent while keeping the model tractable. Thus, we need tractable models that are as expressive as possible, so that we do not compromise too much accuracy. Second, finding a tractable model that fits the data well is often a highly non-convex problem, so even if the data could be fit by a tractable model, we might not be able to find it efficiently. Thus, we need efficient search procedures that identify good structures within this space.

The best-known algorithm for learning tractable graphical models is the Chow-Liu algorithm for learning tree-structured graphical models (Chow and Liu 1968). It finds the maximum likelihood tree-structured model in polynomial time, but it is limited to a tree-width of one. A number of attempts have been made to generalize the algorithm to learn thin junction trees, where the tree-width is larger than one but still constrained (e.g., (Bach and Jordan 2001; Chechetka and Guestrin 2008)); however, many algorithms only find a local optimum, and the running time is often very slow. Arithmetic circuits (Darwiche 2003) are an inference representation that can exploit context-specific independence and determinism to efficiently represent many high-treewidth models as well. They can be learned by adapting greedy Bayesian or Markov network learning algorithms so that they maintain an equivalent arithmetic circuit and penalize structure modifications based on how much they increase the circuit size (Lowd and Domingos 2008; Lowd and Rooshenas 2013).

A second direction is to introduce latent variables or a mixture. The mixture of trees (Meila and Jordan 2000) algorithm uses expectation maximization (EM) to learn a mixture model where each component is a tree-structured graphical model. Alternately, a latent tree model (e.g., (Choi et al. 2011)) consists of a tree of latent variables with observable variables at the leaves. Recently, sum-product networks (Poon and Domingos 2011) have been shown to be both more accurate and more efficient than deep belief networks at modeling image data. A sum-product network is a network of sum nodes, which represent mixtures, and product nodes, which represent independent subgroups of variables. Weights are learned using hard EM, and the structure is either prespecified for the problem (Poon and Domingos 2011) or learned using a hierarchical clustering algorithm (Dennis and Ventura 2012).

Each of these directions has its own strengths. Mixture models are good at representing domains that have natural

Table 1: Log-likelihood comparison

Dataset	Var#	MAC	ACMN	MT	CL	SPN	LTM
NLTCS	16	-6.01	**-6.00**	-6.01	-6.76	-6.12	-6.49
Plants	69	**-12.59**	-12.80	-13.18	-16.52	-13.68	-16.39
Audio	100	**-39.78**	-40.32	-40.07	-44.37	-41.11	-41.90
Jester	100	**-52.71**	-53.31	-53.06	-58.23	-54.04	-55.17
Netflix	100	**-56.40**	-57.22	-56.77	-60.25	-58.73	-58.53
MSWeb	294	-9.82	**-9.76**	-9.88	-10.19	-10.31	-10.21
Book	500	**-34.19**	-35.56	-34.70	-37.82	-34.98	-34.22
WebKB	839	**-153.84**	-159.13	-156.05	-163.43	-165.60	-156.84
Reuters-52	889	**-83.75**	-90.23	-86.36	-94.39	-95.32	-91.23
20 Newsgroup	910	**-153.50**	-163.26	-155.11	-164.13	-159.83	-156.77

clusters, since each cluster can be represented by a relatively simple mixture component. Thin junction trees and arithmetic circuits are better when the domain has conditional or context-specific independencies, since they can represent direct relationships among the observed variables. Neither is more expressive than the other: converting a mixture model to a model with no latent variables typically results in a clique over all variables, and converting a thin junction tree to a mixture of trees or other latent variable model could require an exponential number of clusters. Therefore, we propose to get the best of both worlds by learning mixtures of arithmetic circuits (MAC) models.

Empirically, MAC is more accurate than 5 baseline methods on 8 out of 10 datasets, and it comes in second on the remaining 2.

MAC: Mixture of ACs

To learn a mixture of arithmetic circuits (MAC) model, we first partition the data into a small number of clusters (2-6) using the k-means algorithm. We chose k-means because it is simple and efficient, but an alternate EM-based clustering method would probably work as well. Next, we use the ACMN algorithm (Lowd and Rooshenas 2013) to learn an arithmetic circuit for each segment of the data. The ACMN algorithm is based on greedy algorithms for learning Markov networks and explores a similar space of models, preferring those that have compact arithmetic circuits. The ACMN algorithm is relatively slow to run on the complete dataset, but is faster to run when the input is smaller or the resulting circuit is constrainted to be smaller. Thus, MAC actually runs faster than the original ACMN algorithm. The component circuits are put together into a mixture with equal weights, defining an initial model: $P(X=x) = \sum_{i=1}^{k} \pi_i P_i(X=x)$, where π_i are the mixture weights (initially $1/k$) and P_i is the probability distribution from the ith arithmetic circuit. This model is refined by running hard EM: each training instance is reassigned to the mixture component P_i that gives it the highest probability; then, the parameters of each mixture component are updated to fit their assigned instances. This process is repeated until convergence. When updating the mixture components, we do not change the structure, since rerunning the ACMN algorithm in each iteration would be very expensive.

Experiments

We evaluated MAC on 10 datasets with 16 to 910 binary-valued variables, also used by (Davis and Domingos 2010) and others. In order to show the accuracy of MAC, we compared it with the state-of-the-art learning methods for SPNs (Dennis and Ventura 2012), mixtures of trees (MT) (Meila and Jordan 2000), ACMN (Lowd and Rooshenas 2013), and latent tree models (LTM) (Choi et al. 2011). We also included the Chow-Liu algorithm (CL) as an additional baseline. All hyperparameters were tuned on validation data. We used the authors' original code for SPNs, LTM[1], and ACMN, and our own implementation of CL, MT, and MAC. We also extended ACMN to use an L_1 prior, which we found led to slightly better results. Additional details about our implementations and tuning procedures are omitted due to space, but are available upon request.

Table 1 shows the log-likelihood of each method on each dataset. All methods produced models that allow for efficient exact inference, but MAC produced the most accurate models on 8 out of 10 datasets and came in second on the remaining two. On NLTCS and MSWeb, ACMN does slightly better than MAC, suggesting that those datasets do not have well-defined clusters. MAC does almost as well, since each of its components is a (slightly simpler) ACMN model. On other datasets, such as 20 Newsgroups, the mixture models LTM and MT do much better than ACMN. Here, MAC does even better, since its mixture components are richer.

Conclusion

Over the last few years, many methods have been developed to learn different types of graphical models that guarantee efficient inference. Most of them either learn a restricted Bayesian or Markov network or some type of mixture model. MAC combines these ideas by learning a mixture of arithmetic circuits. In our experiments, MAC displays high accuracy across a range of benchmark datasets, suggesting that the combination of arithmetic circuits and mixture models is good for learning models that are both accurate and efficient. In future work, we plan to apply MAC to image datasets, such as those previously used to evaluate SPNs.

[1] http://people.csail.mit.edu/myungjin/latentTree.html

References

Bach, F., and Jordan, M. 2001. Thin junction trees. *Advances in Neural Information Processing Systems* 14:569–576.

Chechetka, A., and Guestrin, C. 2008. Efficient principled learning of thin junction trees. In Platt, J.; Koller, D.; Singer, Y.; and Roweis, S., eds., *Advances in Neural Information Processing Systems 20*. Cambridge, MA: MIT Press.

Choi, M. J.; Tan, V.; Anandkumar, A.; and Willsky, A. 2011. Learning latent tree graphical models. *Journal of Machine Learning Research* 12:1771–1812.

Chow, C. K., and Liu, C. N. 1968. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory* 14:462–467.

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

Davis, J., and Domingos, P. 2010. Bottom-up learning of Markov network structure. In *Proceedings of the Twenty-Seventh International Conference on Machine Learning*. Haifa, Israel: ACM Press.

Dennis, A., and Ventura, D. 2012. Learning the architecture of sum-product networks using clustering on variables. In *Advances in Neural Information Processing Systems 25*, 2042–2050.

Lowd, D., and Domingos, P. 2008. Learning arithmetic circuits. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*. Helsinki, Finland: AUAI Press.

Lowd, D., and Rooshenas, A. 2013. Learning Markov networks with arithmetic circuits. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2013)*.

Meila, M., and Jordan, M. 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 *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI-11)*. Barcelona, Spain: AUAI Press.