Course project for CSC2515 taught by Sam Roweis, Fall 2006.

Adaptive Tree CPDs in Max-Product Belief Propagation

Daniel Tarlow
Department of Computer Science
University of Toronto
dtarlow@cs.toronto.edu

Abstract

In general, the problem of computing the maximum a posteriori (MAP) assignment in a Bayesian network is computationally intractable. In some cases, such as in tree-structured networks, inference can be done efficiently and exactly. However, there are still practical challenges when trying to do inference in networks containing variables with large cardinalities. In this case, representing and manipulating the local conditional probability densities (CPDs) may be cumbersome with standard techniques. Since one is then typically forced to resort to approximations in the CPD representation, exact inference becomes intractable even in networks with otherwise tractable structure. I present an adaptive CPD representation suitable for max-product inference that is able to adjust its complexity as inference progresses, offering a means of performing exact inference in networks with tractable structure but prohibitively large variable cardinalities. I show results for a series of experiments on Hidden Markov Models, showing that my technique gives a speed improvement over standard Viterbi decoding on networks with large state spaces, then I show that my method is able to successfully perform inference on problems where inference would otherwise be intractable.

1 Introduction

Bayesian networks [6] and other probabilistic graphical models have been applied to a wide variety of real-world problems. A common task for which these models are applied is the probabilistic inference task of computing the posterior distribution of one or more variables. An alternative approach, whose popularity has grown in recent years, is based on the maximum a posteriori (MAP) inference problem — computing the single most likely assignment relative to the distribution.

It is well known that the complexity of either inference task is exponential in the tree width of the network, forcing one to resort to approximate inference techniques for many real-world problems. One of the most promising techniques of recent years is loopy belief propagation to compute posterior marginals, and max-product loopy belief propagation to compute the MAP assignment [5].

Most research related to belief propagation attempts to deal with the problems that result from trying to do inference in networks with loops and in networks with many nodes that interact in a complex manner. A dimension that has received less attention is how to perform inference when the cardinalities of variables are extremely large. In this situation, inference can become intractable due simply to the fact that messages are too large to compute or store.

Common cases where variables might have extremely large cardinalities are in domains where variables in the network represent pixels in an image, words in a sentence, or genes. In networks with large variable cardinalities, the typical solution is to resort to approximations. In the case of images, a super-pixel representation might be used, where only one pixel within any super-pixel is considered [2]. Similarly, one may try to embed the clique potentials in a lower dimensional

subspace. In cases where the type of discretization to use is less clear, one may resort to exactly storing only the K most likely configurations within a clique, assuming that all other configurations have either zero probability or a small fixed probability [7].

2 Tree CPD Representation

Boutilier, Friedman, Goldszmidt, and Koller [1] showed that tree CPDs can be used to represent context-specific independencies in Bayesian networks and simplify some inference procedures. In fact, [7] can be viewed as simply applying the idea of using tree CPDs to Hidden Markov Models. I build upon this idea, additionally showing how to adapt the tree structure over the course of inference in such a way as to guarantee that inference is exact in tree-structured networks.

3 Bounding the MAP Score

My approach is based upon the idea that tree CPDs can be used to check whether a full assignment is the MAP solution, given an upper bound on the best of the other values in each of the local factors. By multiplying tree factors while maintaining one additional bit specifying whether each leaf is a bound or not, we can tell whether each leaf in the resulting tree factor is an exact value or an upper bound. A leaf in a resulting factor is an upper bound if any entry contributing to it was also an upper bound.

Taking this approach, we can see that the value associated with any resulting leaf that is an upper bound is an upper bound on the contribution of that factor to the MAP score. If the resulting factor is a belief over a variable, the maximum-valued entry is fully specified, and the global network structure is a tree, then we have the MAP assignment to one variable.

The outline of my algorithm, then, is to locally refine the CPDs, essentially making informed guesses about what the MAP assignment is, then testing to see if the local guesses do in fact correspond to the MAP assignment for each variable.

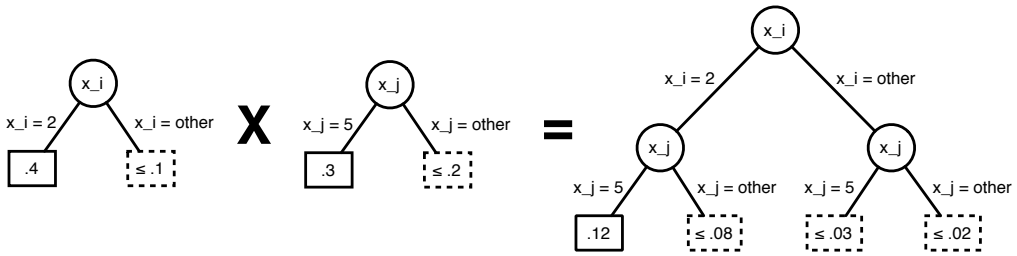


Figure 1: Multiplication of two tree CPDs. Note that we know that the MAP joint assignment is $x_i = 2$ and $x_j = 5$, because the value of the factor at the leaf is greater than any other bound or non-bound leaf. If the largest value was associated with a bound node, then we would need to further refine the local trees in order to find the MAP assignment.

4 Guessing and Checking the MAP Solution

The next step that must be specified is how to choose the guesses to be made at each point.

I build a cluster graph that has one cluster for each factor, which allows each cluster to calculate the joint belief over its scope. Since the belief is the product of incoming messages, which are trees, the resulting belief will also be a tree. The goal is to have the leaf with the largest value be a single assignment.

A simple method for choosing how to refine the local CPD, then, is to split the leaf that has the largest value associated with it in the tree representing that cluster's belief. Formally, if $\mathbf{X_{C_i}}$ is the set of "bound variables" for a given leaf, which do not have a value specified, and $\mathbf{Y_{C_i}} = Scope(C_i) - \mathbf{X_{C_i}}$ is the set of variables that do have values specified at the given leaf, then we would like to find:

$$X^*_{C_i} = \arg\max_{\mathbf{X_{C_i}}} b_{C_i}(Y_{C_i}, X_{C_i})$$

Combining $X^*_{C_i}$ with Y_{C_i} gives a fully specified assignment to all variables within the scope of C_i, and this is the leaf that is created and added to C_i's tree.

5 Refining Local CPD Trees

A naive way to find the needed values is compute a standard table CPD, then sort assignments along the needed dimensions. However, a major savings can be achieved here if special purpose data structures or algorithms can be used to find the top assignments without having to explicitly calculate and sort potentials for all possible assignments. For example, if a clique represents string distance between English words, then there are many techniques for building indices, which can make finding the top matches very efficient for any given word. In domains with images, any number of pruning techniques for template matching algorithms or convolutions may be used. The problem becomes slightly more involved in cliques that range over multiple variables and require that some variable values be fixed, but a similar approach may be followed. In my experiments, I use Aspell, a special-purpose spell checker to generate a (relatively) small number of candidate words, then I only calculate potentials for and sort these candidate assignments.

6 Experiments: A spell checker that uses contextual information

Graphical models with words as variables were one of the primary motivations of this work, so I decided to build a network that has words as the hidden state in the network. By viewing typed words as noisy observations of the true word that a user meant to type, we can formulate a Hidden Markov Model as a spelling correcter.

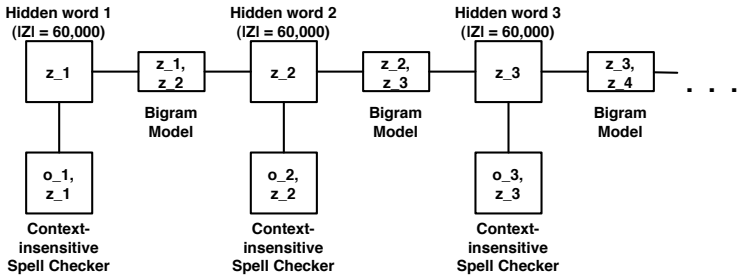


Figure 2: (a) A Hidden Markov Model that ties together a number of context insensitive spell checkers. Note that the state space for the English language might be on the order of 60,000 words. The motivation for this work is to develop exact inference methods that may scale to such a problem.

I use the GNU Aspell spell checker to generate a list of candidate words for each observed word. This generates a large number of closely related words, but it does not compute a meaningful distance function. To compute the distance to be used in the clique potential, I use dynamic programming to compute an edit distance between each observed word and the candidate words. The operations allowed are Insertion, Deletion, Substitution, and Swap, each at a fixed cost of one. The clique potentials are then calculated as $\phi(s') = e^{-\frac{dist(s,s')}{\sigma}}$, where s is the word that was actually typed, s' is each word suggested by Aspell, and σ is a bandwidth that was chosen by hand (since the focus of this work is the inference, not the learning).

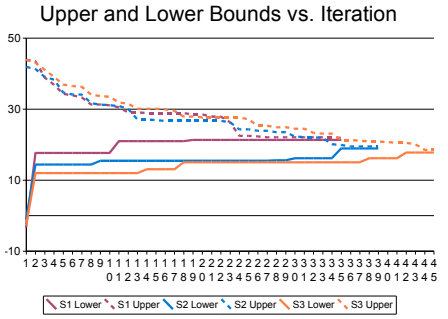
Simply to make the experiments easier to run and analyze, I limited the vocabulary to 1000 words. To model transition probabilities, I used the bigrams data set from assignment 3 and took the first 1000 words.

I pass messages forwards and backwards while holding the CPD structures constant, then I calculate the beliefs at each observation node and check to see whether the maximum entry is a bound. If it is a bound, then I further split that CPD, choosing the next most likely local assignment according to the scheme described above. I continue to iterate until either no beliefs are bounds, or until 100 iterations. Note that stopping early and taking the best non-bound assignment is essentially the approximate inference algorithm of [7], except with adaptive rather than fixed tree CPDs. Ideally, we would also split the transition cliques by constraining one variable to a given value (depending on which bound assignment was the largest) and maximizing over the other. However, due to the fact

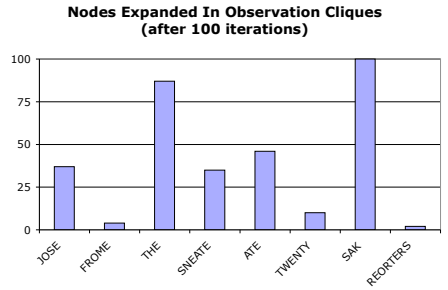
that this project became a much larger task than I had intended, I instead chose to have the transition cliques simply to mirror the splits of the adjacent singleton cliques.

To generate data, I asked a friend who didn't know what I was doing to make up ground truth sentences from the limited vocabulary that I was using. I then asked him to type each sentence a number of times while making a large number of typos. I show results over a subset of this data.

Software was implemented in C++. All experiments were conducted on a 1.5 GHz Macintosh PowerBook G4 with 768MB RAM.



(a)



(b)

Figure 3: (a) Successively expanding leaves in the tree CPDs can be viewed as squeezing together an upper and lower bound on the log likelihood of the model. The bounds can be calculated by taking the product of all potentials consistent with the best assignment while allowing or not allowing some assignments to be upper bounds. I show the path that this bound takes over the course of inference for three similar sentences. When the bounds become equal, the Squeeze Theorem guarantees that we have the exact solution. (b) If a node computes its belief for a variable after the HMM has been calibrated and the most likely assignment to that variable is not a bound, then we know the value of a single variable and we can stop expanding that tree. Here, the number of nodes that were expanded per observation clique are shown for one sentence.

Algorithm	Sentence	Errors
Input	JOSE ~~FROME~~ THE ~~SNEATE~~ ~~ACE~~ TWENTY ~~SAK~~ ~~REORTERS~~.	5
HMM	JOSE FROM THE SENATE ~~AGE~~ TWENTY SIX REPORTERS.	1
Edit Distance	JOSE FROM THE SENATE ~~ACE~~ TWENTY ~~SAY~~ REPORTERS.	2
Aspell	JOSE ~~FRO~~ ~~ME~~ THE SENATE ~~ACE~~ ~~SKA~~ ~~REORDERS~~.	4
MS Word	JOSE ~~FRAME~~ THE SENATE ~~ACE~~ TWENTY ~~SKI~~ REPORTERS.	3
Ground Truth	JOSE FROM THE SENATE ATE TWENTY SIX REPORTERS	0

Algorithm	Sentence	Errors
Input	A ~~MLAN~~ ~~DLARS~~	2
HMM	A MILLION DOLLARS	0
Edit Distance	A ~~MEANS~~ ~~DAILY~~	2
Aspell	A ~~MILAN~~ ~~DALES~~	2
MS Word	A ~~MANN~~ ~~DEARS~~	2
Ground Truth	A MILLION DOLLARS	0

Figure 4: Results from inference over three different sentences. Comparisons are provided between my method and pure edit distance (which is essentially half of the input given to the HMM), Aspell, Microsoft Word, and the ground truth. Note that comparisons against Microsoft Word and Aspell are not fair comparisons because the vocabulary that they had to choose from was much larger. However, it does help emphasize that these are difficult problems.

7 Discussion

The results presented that evaluate this model as a spell checker should be taken with a grain of salt. The vocabulary was very limited, and the transition model did not contain enough data to truly justify a full bigram model. Because of this, the real world performance should be heavily dependent on peculiarities within the data. Further, the bandwidth parameter in the edit distance controls how heavily to weight local versus contextual information, and no effort was made to set that to anything beyond a semi-reasonable value.

However, this work might be viewed as a very early proof-of-concept that there may be principled ways of adapting CPD structures over the course of inference. Recent work [3] has shown that adapting message schedules in loopy belief propagation, based upon message contents, can significantly improve the performance of inference. I suggest that a similar thing may be possible for CPDs that follows in the spirit of this work.

8 Summary

In this work, I present a means of adapting CPD structure over the course of max-product inference in such a way as to guarantee that inference is exact. The underlying assumption that this technique makes is that information about a variable coming from different sources is somewhat consistent. Though one may be able to construct examples where this technique does not provide any savings over inference with standard CPD representations, it would be difficult to imagine what this would be saying about the domain. For example, in the spell checking domain, two sources of information would have to not shed light on what the best true word was: first, the word must not be spelled correctly and the edit distance between the typed word and the true word must be large; and second, contextual information coming from either the previous or the next word must not be particularly revealing about what the true word is. In this case, my technique would struggle.

However, as presented, this is not a technique for solving tricky and complex inference problems; this is meant to be a method to solve inference problems that would be easy if it were not for the size of the variable cardinalities in the network. With this approach, problems where computing even a single message may be prohibitively expensive might be solved quickly and exactly.

9 Acknowledgments

Thanks to Andrew Peterman and Rohan Seth for being data-generating guinea pigs. Also, thanks to Mike Jurka for some C++ snippets to load large sparse bigram matrices.

References

[1] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. pages 115–123.

[2] Gal Elidan, Geremy Heitz, and Daphne Koller. Learning object shape: From drawings to images. CVPR, 2:2064–2071, 2006.

[3] Gal Elidan, Ian McGraw, and Daphne Koller. Residual belief propagation. In Uncertainty in Artificial Intelligence (UAI), 2006.

[4] Finn Jensen and Frank Jensen. Optimal junction trees. In Proceedings of the 10th Annual Conference on Uncertainty in Artificial Intelligence (UAI-94), pages 360–36, San Francisco, CA, 1994. Morgan Kaufmann.

[5] K. Murphy and Y. Weiss. Loopy belief propagation for approximate inference: An empirical study. pages 467–475.

[6] J. Pearl. Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann, 1988.

[7] Sajid Siddiqi and Andrew Moore. Fast inference and learning in large-state-space hmms. In Proceedings of the 22nd International Conference on Machine Learning, August 2005.