Fast Inference and Learning with Sparse Belief Propagation

Chris Pal, Charles Sutton and Andrew McCallum
University of Massachusetts
Department of Computer Science
Amherst, MA 01003
{pal,casutton,mccallum}@cs.umass.edu

Abstract

Even in trees, exact probabilistic inference can be expensive when the cardinality of the variables is large. This is especially troublesome for learning, because many standard estimation techniques, such as EM and conditional maximum likelihood, require calling an inference algorithm many times. In max-product inference, a standard heuristic for controlling this complexity in linear chains is beam search, that is, to ignore variable configurations during inference once their estimated probability becomes sufficiently low. Although quite effective for max-product, during sum-product inference beam search discards probability mass in a way that makes learning unstable. In this paper, we introduce a variational perspective on beam search that uses a approximating mixture of Kronecker delta functions. This motivates a novel variational approximation for arbitrary tree-structured models, which maintains an adaptively-sized sparse belief state—thus extending beam search from max-product to sum-product, and from linear chains to arbitrary trees. We report efficiency improvements for max-product inference over other beam search techniques. Also, unlike heuristic methods for discarding probability mass, our method can be used effectively for conditional maximum likelihood training. On both synthetic and real-world problems, we report four-fold increases in learning speed with no loss in accuracy.

1 Introduction

A standard technique for fast max-product inference with large discrete variables is *beam search*, that is, ignoring variable configurations whose estimated max-marginal is sufficiently low. Beam search is often quite effective, and is widely used in applications such as speech recognition and natural-language parsing. In speech recognition, for example, beam search is a critically important component of large-scale hidden Markov model (HMM)-based systems [2, 5, 9].

But large variable spaces are problematic for parameter estimation, because standard estimation techniques, such as expectation maximization and conditional maximum likelihood, require many calls to inference. Furthermore, whereas beam search is quite effective in max-product inference, many learning techniques call for sum-product inference, and discarding probability mass during sum-product inference often makes learning unstable.

Inspired by beam search, in this paper we introduce *sparse belief propagation*, that is, a variational procedure that approximates a distribution with large variable state space by a mixture of Kronecker delta functions that cover a much smaller portion of the state space. This motivates a novel message-passing algorithm in which after each message pass, the belief is compressed while maintaining the approximating global distribution within a fixed KL-divergence of the true distribution. Essentially, this extends beam search from max-product inference to sum-product, and from linear-chain models to arbitrary trees.

For max-product, we show experimentally that this KL-divergence criterion is more robust and efficient than simpler beam search methods. For sum-product inference, we present results using sparse BP to compute marginals for approximate training of conditional random fields. Training using sparse BP can require less than one-fourth of the time of exact training with no loss in accuracy, whereas more naive methods for discarding probability mass yield unstable training methods. On one real-world task, the NetTalk text-to-speech data set [6], we can now train a CRF in about 6 hours for which previously training required over a day, with no loss in accuracy.

The rest of the paper is structured as follows. First, we briefly review HMMs and CRFs and discuss current methods for learning CRFs. We then present our variational view of beam search, and show how it leads to natural extensions to sum-product inference and arbitrary graphs. Finally, we present experimental results for max-product inference, and sum-product embedded within CRF training.

2 Notation and Background

In this section, we present our notation for chain-structured models, on which we focus in this paper. We also briefly review current techniques for CRF parameter estimation.

HMMs are a classical type of generative sequence model. Define an observation sequence of discrete random variables as $\mathbf{x} = x_1, \ldots, x_n$ and a sequence of discrete random variables for the state (label) variables as $\mathbf{y} = y_1, \ldots, y_n$. Then an HMM defines the sequence probability as

$$p(\mathbf{y}, \mathbf{x}) = \prod_{i=1}^{n} p(x_i|y_i)p(y_i|y_{i-1}) \tag{1}$$

where for simplicity we define $p(y_1|y_0) = p(y_1)$. For inference and estimation we are often interested in computing posterior marginal distributions over hidden labels. For many decoding applications, we are interested in quickly computing the most probable $\mathbf{y}$.

A conditional random field (CRF) represents a conditional probability distribution using a set of features. Let Q be a set of cliques in a undirected graph containing the random variables in $\mathbf{x}$ and $\mathbf{y}$. A CRF represents the conditional probability of the label variables $\mathbf{y}$ given observations $\mathbf{x}$ as

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{q \in Q} \Psi_q(\mathrm{y}_q, \mathrm{x}_q), \tag{2}$$

where $Z(\mathbf{x}) = \sum_{y \in \mathbf{y}} \prod_{q \in Q} \Psi_q(\mathrm{y}_q, \mathrm{x}_q)$ is a normalizing factor over all output configurations. A CRF is usually parameterized using feature functions functions $\{f_k\}$ for each q such that

$$\Psi_q(\mathrm{y}_q, \mathrm{x}_q) = \exp\left(\sum_k \lambda_k f_k(\mathrm{y}_q, \mathrm{x}_q)\right), \tag{3}$$

where λ_k are the parameters or feature weights for the model.

Parameter estimation in a CRF is typically accomplished by maximizing the log-likelihood of fully-observed training data $\mathcal{D} = \{\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i\}_{i=1}^{N}$. If we define $\boldsymbol{\lambda}$ as a parameter vector, F

as a feature function and $\mathbf{F}(\mathbf{y}, \mathbf{x}) = \sum_q F(\mathrm{y}_q, \mathrm{x}_q)$ as the *global feature function*, then the gradient of the log-likelihood $\mathcal{L}$ with respect to the model parameters is given by

$$\nabla_{\boldsymbol{\lambda}} \mathcal{L} = \nabla_{\boldsymbol{\lambda}} \Big(\sum_i \log p(\tilde{\mathbf{y}}_i | \tilde{\mathbf{x}}_i, \boldsymbol{\lambda}) \Big) = \sum_i \Big(\mathbf{F}(\tilde{\mathbf{y}}_i, \tilde{\mathbf{x}}_i) - E_p \langle \mathbf{F}(\mathbf{y}_i, \tilde{\mathbf{x}}_i) \rangle \Big), \tag{4}$$

where $E_p\langle\cdot\rangle$ denotes the expectation under the distribution $p = p(\mathbf{y}_i | \tilde{\mathbf{x}}_i, \tilde{\boldsymbol{\lambda}})$. It is important to observe that this requires performing inference once for each sequence, per iteration of the optimizer. For data sets with large state spaces, this procedure can require days of computation.

Following previous work [7], we optimize the parameters using limited-memory BFGS [4], a quasi-Newton gradient based optimizer. This has implications for inference algorithms that compute $E_p\langle\cdot\rangle$ approximately. Inaccurate marginals lead to inaccurate gradients, which will severely hurt the convergence of any gradient-based optimizer, as observed in negative results with early stopping of loopy BP [8].

3 Sparse Belief Propagation

Beam search can be viewed as maintaining a sparse distribution that is as close as possible to a large distribution. In this section, we formalize this intuition using a variational argument, which motivates several new beam-search-like approximation algorithms.

3.1 Unstructured Discrete Distributions

Consider a discrete distribution $p(x)$, where x is assumed to have very many possible configurations. We approximate p by a sparse distribution q, which we write as a mixture of Kronecker delta functions:

$$q(x) = \sum_{i \in I} q_i \delta_i(x), \tag{5}$$

where $I = \{i_1, \ldots, i_k\}$ is the set of indices i such that $q(x = i)$ is non-zero, and $\delta_i(x) = 1$ if $x = i$. We refer to the set I as *the beam*.

In this section, we consider the problem of finding the distribution $q(x)$ of smallest weight such that $\mathrm{KL}(q\|p) \leq \epsilon$. First, suppose the set $I = \{i_1, \ldots, i_k\}$ is fixed in advance, and we wish to choose the probabilities q_i to minimize $\mathrm{KL}(q\|p)$. Then the optimal choice is simply $q_i = p_i / \sum_{i \in I} p_i$, a result which can be verified using Lagrange multipliers on the normalization constraint of q.

Second, suppose we wish to determine the set of indices I of a fixed size k which minimize $\mathrm{KL}(q\|p)$. Then the optimal choice is when $I = \{i_1, \ldots, i_k\}$ consists of the indices of the largest k values of the discrete distribution p. First, define

$$Z(I) = \sum_{i \in I} p_i \tag{6}$$

Then the optimal approximating distribution is:

$$\arg\min_q \mathrm{KL}(q\|p) = \arg\min_I \left\{ \arg\min_{\{q_i\}} \sum_{i \in I} q_i \log \frac{q_i}{p_i} \right\} \tag{7}$$

$$= \arg\min_I \left\{ \sum_{i \in I} \frac{p_i}{Z(I)} \log \frac{p_i / Z(I)}{p_i} \right\} \tag{8}$$

$$= \arg\max_I \big\{ \log Z(I) \big\} \tag{9}$$

That is, the optimal choice of indices is the one that retains most probability mass.

This means that it is straightforward to find the discrete distribution q of minimal weight such that $KL(q||p) \leq \epsilon$. We can sort the elements of the probability vector p, truncate after the total mass exceeds ϵ, and renormalize to obtain q.

3.2 Structured Distributions: Sparse Belief Propagation

In this section, we extend the ideas of the previous section to graphically-structured distributions, yielding the sparse BP algorithm.

For a structured distribution of many random variables, such as a linear chain, the naive procedure of enumerating all configurations and sorting is clearly impractical. Therefore, rather than compressing the entire distribution, we instead compress the marginal beliefs after every message pass

Thus, we define *sparse belief propagation* as follows. Perform BP using a standard schedule for message-passing in a tree. For each message from node i to node j, do:

1. Pass the message in the standard way:
$$m_{ij}(x_j) \leftarrow \sum_{x_i} \psi(x_i, x_j) \prod_{k \in N(j) \setminus i} m_{kj}(x_i) \tag{10}$$
2. Compute the new dense belief b as
$$b(x_j) \propto \psi(x_j) \prod_{i \in N(j)} m_{ij}(x_j) \tag{11}$$
3. Compress into a sparse belief $b'(x_j)$, maintaining $\mathrm{KL}(b'\|b) \leq \epsilon$. Call the resulting beam I_j.
4. Compress $m_{ij}(x_j)$ to respect the new beam I_j.

Note that in every compression operation, the beam I_s is recomputed from scratch; therefore, variable configurations can both leave and enter the beam during a message pass.

Also, because the graph has no loops, it is easy to see by substitution that each message $m_{ij}(x_j)$ is the sum of the mass of all configurations of upstream nodes such that $X_j = x_j$. In particular, after all messages have been sent, for any node i we have
$$Z_i(I) = \sum_{x_i} \psi(x_i) \prod_{i \in N(j)} m_{ij}(x_j) \tag{12}$$
is the total mass of the beam I in the global distribution (analogously, in fact, to standard BP).

But this means, by the argument about unstructured distributions in the last section, that $Z(I)$ is the KL divergence $\mathrm{KL}(p\|q)$ between the global distributions $p(x)$ and $q(x)$. A consequence of this is that each message-pass/compression operation maintains the global invariant that the global divergence $\mathrm{KL}(p\|q)$ is no more than ϵ, or equivalently, that the total mass of all the paths in the beam is at least ϵ.

Finally, we discuss a few practical considerations. We have found improved results by adding a minimum belief size constraint K, which prevents a belief state $q(x_s)$ from being compressed below K non-zero entries. Also, in linear chains, we have found that sparse BP usually finds a good beam after a single forward pass. This is desirable because would like to minimize the number of iterations, especially in a tree: If finding a good beam requires many forward and backward iterations, then after a while one may as well do exact forward-backward.

Sparse beliefs can be applied in the same manner to loopy sum-product and max-product updates, but we leave theoretical and practical analysis of that approach for future work.

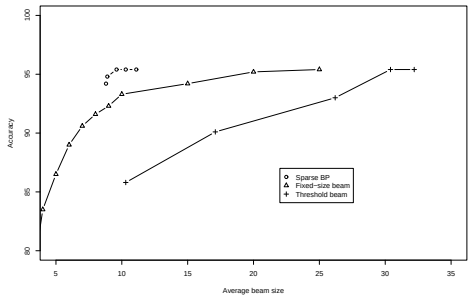

Figure 1: *Comparison of Beam Search Methods over Different Threshold:* Sparse BP achieves top accuracy with a much smaller average beam size than the beam search methods.

	Beam / Constraint Size					
	2	3	4	5	6	12
Accuracy	82.7	88.3	90.4	91.2	91.4	91.6

Table 1: Comparing recognition accuracy on the NetTalk data set using a CRF with a Viterbi accuracy of 91.6%.

4 Results and Analysis

In this section we evaluate sparse BP for both max-product and sum-product inference in HMMs and CRFs and the real-world NetTalk dataset [6].

4.1 Experiments with Max-Product Decoding

First, we evaluate whether in the max-product setting, the speed improvement of sparse BP versus exact max-product, and how sparse BP compares to traditional beam search methods. In our first set of experiments, we generate synthetic data from an HMM of length 75. Transition matrix entries are sampled from a Dirichlet with $\alpha = .1$ and emission matrices are generated from a mixture of a low entropy, sparse conditional distribution with 10 non-zero elements and a high entropy Dirichlet with $\alpha = 10^4$, with priors of $.75$ and $.25$ respectively. The goal is to simulate a regime where most states are highly informative about their destination, but a few are less informative.

	Beam / Constraint Size				
	3	5	10	15	20
Std. N-best	57.8	72.4	82.3	85.4	86.2
$KL \leq .5$	79.7	80.5	83.2	85.8	86.3
$KL \leq .15$	**87.2**	87.2	86.9	87.1	87.1
	Average Beam Size				
$KL \leq .5$	4.9	6.2	10.2	15.0	20.0
$KL \leq .15$	**22.3**	22.6	24.0	25.9	28.3

Table 2: Comparing CRF recognition accuracy on synthetic HMM data for Viterbi Beam Search, our constrained max field algorithm in a CRF. Exact Viterbi decoding had an accuracy of 87.3%.

In Figure 1, we compare sparse BP to traditional beam search algorithms. We compare to two common approaches: a fixed beam size, and an adaptive beam where message entries are retained if their log score is within a fixed threshold of the best so far. For each algorithm, we report the average beam size, using the minimum threshold needed to find the exact Viterbi solution. Sparse BP, using the $KL \leq 0.001$ compression and an additional $|I_i| \geq 4$ constraint, achieves the exact solution with an average only 9.6 states per variable. On the other hand, the fixed beam requires between 20 and 25 states, and the threshold adaptive beam requires 30.4 states per variable to achieve the same accuracy.

Second, we also present max-product results for linear-chain conditional random fields on both synthetic and real-world data. We generate synthetic data from an HMM with 100 hidden states and 100 possible observations. We constrain the HMM to be sparse, having at most 5 transitions per state and 5 emission values per states. We used 50 sequences of length 75 for optimizing the model and 50 examples for testing the model. We train a linear-chain CRF using standard exact training, and compare sparse BP with other beam search methods on the testing data.

Using exact Viterbi decoding, the CRF had a recognition accuracy of 87.3% on the synthetic test data. Table 2 compares exact decoding with the beam results based on thresholds.

Finally, we report decoding results for a real-world CRF trained on the NetTalk data set [6]. The CRF we constructed had 52 states, and was trained using 19075 examples (pronunciations of single words) and tested using 934 examples. Standard Viterbi decoding using the CRF produced an accuracy of 91.6%. Table 1 summarizes accuracy results for fixed beam sizes. Our other experiments found that both the $KL \leq \epsilon$ method and the threshold methods produced the exact Viterbi accuracy when the average number of states explored was 14.

4.2 Learning Experiments

In this section, we present results showing that sum-product sparse BP can be embedded within CRF training, yielding significant speedups in training time with no loss in testing performance.

First, we train CRFs using synthetic data generated from a 100 state HMM generated in the same manner as in the previous section. Again, we use 50 sequences for training and 50 sequences for testing. In all cases we use exact Viterbi decoding to compute testing accuracy. Figure 2 illustrates learning curves comparing sparse BP with $KL \leq 0.5$, $|I_i| \geq 30$ to exact forward-backward optimization. Sparse BP uses one-fourth of the time of exact training with no loss in accuracy.

Also, we examine how important is it for the beam to be adaptive, but also comparing to with an fixed beam with an average size the same as the average number of states used by sparse BP. Sparse BP and the fixed beam converge to same solution, We find that sparse BP and the fixed beam converge to the same solution, but sparse BP finishes faster, indicating that the adaptive beam does help training time. Most of the benefit occurs later in training, as the model becomes farther from uniform.

The right-hand graph in Figure 2 shows the same learning curves on a different scale, adding several other methods for discarding probability mass: a fixed beam of the minimum beam size for the KL beam and a threshold based beam which explores on average roughly the same number of states as the KL beam. In the case of the smaller, fixed beam of size N, our L-BFGS optimizer terminated with an error as a result of the noisy gradient computation. In the case of the threshold beam, the gradients of the optimization were erratic, but L-BFGS did terminate normally. However the recognition accuracy of the final model was low, at 67.1%.

Finally, we present results training on the real-world NetTalk data set. In Figure 3 we

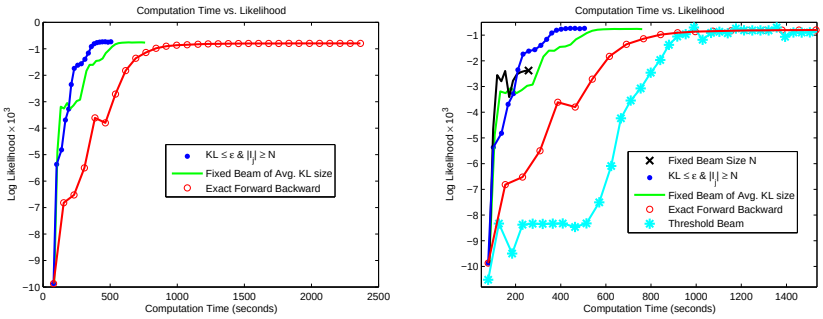

Figure 2: Comparison of beam search methods for CRF training on synthetic data. At left, comparison to exact training. Sparse BP has the same accuracy as exact training with less than a quarter of the training time. At right, a rescaled graph of the same experiment, with several heuristic beam search techniques added. These other heuristics either slower or less robust than sparse BP.

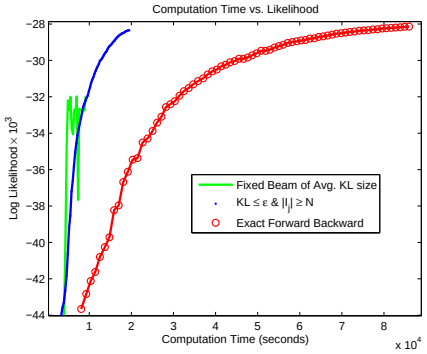

Figure 3: Learning curves of beam search algorithms for CRF training on the NetTalk data set. Sparse BP (final test performance 91.7) performs equivalently to exact training (91.6) using only a quarter of the training time. A fixed-size beam yields unstable results (final testing accuracy 85.7).

present run time, model likelihood and accuracy results for the 52 state CRF trained using the NetTalk data with training and testing partitions as described in Section 4.1. We initialize the CRF parameters using a subset 12% of the data, before training on the full data until convergence. We used the beam methods during the complete optimization run and during this initialization period. During these subset initializations, our experiments with a threshold beam set such that it explored an average of 36 states produced initial parameter estimates which had a test set accuracy of 67%. Our KL method, a fixed size beam of average KL size and exact forward backward all had accuracies of 74%. Further, during the complete run, the threshold beam gradient estimates were so noisy that our L-BFGS optimizer was unable to take a complete step. In the experiments of Figure 3, $\epsilon = .005$ and $N = 10$. Exact forward backward training produced a test set accuracy of 91.6%. In these experiments fixed beam optimization using the average size of our KL beam ($N = 20$) terminated normally but very noisy intermediate gradients were found in the terminating iteration. The result was a much lower accuracy of 85.7%. In contrast, our KL beam achieved an accuracy of 91.7% in a less than 25% of the time it took to exactly train the CRF using forward backward.

5 Related Work

Beam search is a standard method, that is used especially frequently in the speech community. In the graphical models community, there is some old work on zero-compression in clique trees, due to Jensen and Anderson[1]. Their technique considered every potential in a clique tree, and set the smallest potential values to zero, with the constraint that the total mass of the potential does not fall below some fixed value δ. This is clearly related to our technique, but there is an important differences: They pruned the potentials of the model once before performing inference, whereas we dynamically prune the beliefs during inference, and indeed the beam can change during inference as new information arrives from other parts of the model. Also, Jordan et al. [3], in their work on hidden Markov decision trees, introduce a variational algorithm that uses a delta on a single best state sequence, but they provide no experimental evaluation of this technique.

6 Conclusions

We have presented a principled method for significantly speeding up decoding and learning tasks in HMMs and CRFs. We also have presented experimental work illustrating the utility of our approach. As future work, we believe a promising avenue of exploration would be to explore adaptive strategies involving interaction of our L-BFGS optimizer, detecting excessively noisy gradients and automatically setting ϵ values. While the results we have presented here were applied to experiments with HMMs and chain structured CRFs, we believe this approach should be more generally applicable.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval, in part by The Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grants #IIS-0326249 and #IIS-0427594, and in part by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under contract number NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

References

[1] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 1999.

[2] X. Huang, A. Acero, and H. W. Hon. *Spoken Language Processing: A Guide to Theory Algorithms and System Development*, chapter 12. Prentice Hall, New Jersey, 2001.

[3] Michael I. Jordan, Zoubin Ghahramani, and Lawrence K. Saul. Hidden Markov decision trees. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Proc. Conf. Advances in Neural Information Processing Systems, NIPS*, volume 9. MIT Press, 1996.

[4] J. Norcedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

[5] Mosur K. Ravishankar. *Effcient Algorithms for Speech Recognition*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1996.

[6] T.J. Sejnowski and C.R. Rosenberg. Nettalk: a parallel network that learns to read aloud. *Cognitive Science*, 14:179–211, 1990.

[7] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology-NAACL 2003*, Edmonton, Canada, 2003.

[8] Charles Sutton, Khashayar Rohanimanesh, and Andrew McCallum. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML)*, 2004.

[1] This paper was originally presented in UAI 1990, but now appears to be unavailable. The technique is discussed somewhat in [1]).

[9] Monika Woszczyna. *Fast Speaker Independent Large Vocabulary Continuous Speech Recognition*. PhD thesis, Universität Karlsruhe, 1998.