Multifactor Expectation Maximization for Factor Graphs

Anonymous Author(s)
Affiliation
Address
email

Abstract

Factor graphs allow large probability distributions to be stored efficiently and facilitate fast computation of marginal probabilities, but the difficulty of training them has limited their use. Given a large set of data points, the training process should yield factors for which the observed data has a high likelihood. We present a factor graph learning algorithm which on each iteration merges adjacent factors, performs expectation maximization on the resulting modified factor graph, and then splits the joined factors using non-negative matrix factorization. We show that this multifactor expectation maximization algorithm converges to the global maximum of the likelihood for difficult learning problems much faster and more reliably than traditional expectation maximization.

1 Introduction

The computations performed by the brain depend, at least implicitly, upon probability distributions over large sets of variables. Sensory inputs rarely provide an unambiguous characterization of the outside world. In the face of ambiguous input data, tasks ranging from object identification in visual scenes to the planning of motor actions require probabilistic reasoning. Probabilistic inference on nontrivial probability distributions is thus an essential form of computation in the brain, as well as in artificial systems designed to interact with a complex sensory environment.

Factor graphs are an attractive formalism for probability distributions over many variables because they permit efficient storage and fast computation of quantities like marginal probabilities [1]. In a factor graph, the full probability distribution over a set of variables is approximated with the normalized product of many functions, each defined over only a small subset of the variables. Unfortunately, their use has been limited at least in part by the difficulty of choosing appropriate parameters on hard problems. In this paper, we present a method for training factor graphs, based upon the expectation maximization (EM) algorithm [2, 3] and non-negative matrix factorization (NMF) [4], which we have found to give greatly improved performance compared to traditional EM on difficult probability distributions. The main idea is to merge a set of adjacent factors, maximize the likelihood of the observed data over the new joint factor using expectation maximization, and then split the joint factor to return to the original graph topology via non-negative matrix factorization. This is the first application we know of where non-negative matrix factorization methods are used in conjunction with factor graphs. In section 2, we review expectation maximization for factor graphs. We derive multifactor expectation maximization as an extension of traditional EM and non-negative matrix factorization in section 3. Section 4 describes the specific probability distributions, factor graphs, and learning algorithm parameters we used to compare the performance of multifactor EM with single factor EM, and the results of these tests are presented in section 5.

2 Expectation maximization for factor graphs

Consider a set of functions f_a with domains $X_a \subset X = Y \cup H$, where Y is the set of observed variables and H is a set of hidden variables not available for direct observation. In a factor graph, these functions are called factors, and are combined to construct a probability distribution over X defined by $p(x) = \frac{1}{Z} \prod_a f_a(x_a)$, where Z is a normalizing factor called the partition function, defined as

$$Z = \sum_X \left(\prod_a f_a(X_a) \right). \tag{1}$$

Z assures that $\sum_X p(X) = 1$. For simplicity, we consider the case where all variables in X are discrete, although the generalization to continuous variables is straightforward.

Before a factor graph can be used to reason about the world, the functions $f_a(X_a)$ must be chosen. While in some settings, useful systems can be constructed by selecting the f_a manually using expert knowledge [5], the brain must instead learn from a finite set of experiences in a fundamentally unsupervised fashion. If we think of the brain as simply modeling its environment, this unsupervised learning could take the form of maximizing the likelihood of a set of observed data points $\mathbf{Y} = \{Y_i \in Y\}$. Maximizing the posterior probability of the functions $f_a(X_a)$ given the observed data is equivalent to maximizing the likelihood of the data if the prior probability over the possible $f_a(X_a)$ is flat.

One particularly effective method for maximizing the likelihood of a factor graph over its parameters is the expectation maximization algorithm [6]. The expectation maximization algorithm is an iterative method under which the log likelihood of the observed data ($\log p(\mathbf{Y}|\theta)$, where θ indicates the choice of all $f_a(X_a)$), provably converges to a stationary point, likely a local maximum, even though all operations are performed on a function of the total data (both observed and hidden) [2, 3]. Specifically, we maximize $E\left[\log p(\mathbf{X}) \mid \mathbf{Y}, \theta\right]$ with respect to one factor f_a (for a full derivation, see [8]). The result, also known as iterative proportional fitting (IPF) [7], is

$$f_a(x_a) \leftarrow \frac{f_a(x_a) \cdot \langle p(X_a = x_a | Y, \theta) \rangle}{p(X_a = x_a | \theta)}. \tag{2}$$

3 Multifactor expectation maximization

Each iteration of the EM algorithm optimizes an entire factor given the other factors [7, 8], but in practice, learning remains difficult. One possible reason is that factor graphs with hidden variables have many symmetries. In particular, the values of each hidden variable can be permuted (i.e., all factors connected to the same hidden variable can be multiplied by a permutation matrix) without affecting the probability distribution over the observed variables. In part because of these symmetries, the fitness landscape is on average very flat, and many iterations are required before the collection of factors connected to a single hidden variable reaches a consensus on how each value of the hidden variable should be used [9]. As shown in Figure 4(c), as the number of possible values of the variables is increased while holding the structure of the graphical model and the probability distribution on which it is trained otherwise constant, the number of iterations required for the EM algorithm to converge increases roughly exponentially. The large number of updates required for training with EM is particularly troublesome in loopy graphical models, where it is computationally expensive to compute the marginal probabilities necessary for each EM update [1].

Rather than considering each factor separately, we can group adjacent factors before performing the EM algorithm. Specifically, we replace these adjacent factors with an equivalent new factor over the variables to which they were connected (other than their shared variables), as in Figure 1. We consider here two adjacent factors f_w and f_h defined over pairs variables, $\{i, j\}$ and $\{j, k\}$ respectively, where variable j is only shared by f_w and f_h. This is equivalent to the more general case where f_w and f_h have an arbitrary number of arguments if i and k are understood to be the Cartesian products of the unshared variables of f_w and f_h, respectively. The values of the entries of this new factor are equal to the inner product of the original factors:

$$f_v(i, k) = \sum_j f_w(i, j) \cdot f_h(j, k). \tag{3}$$

Equation (3) can be written in matrix notation as $V = W \cdot H$, where W and H are the matrices containing the entries of the two factors f_w and f_h, with the shared variable j along the columns and rows respectively, and with i and k comprising the other dimension of each matrix (respectively). We use the EM algorithm on this modified factor graph to calculate the desired new values for the joint factor V, which then must be factored to obtain values for the original separate factors. So long as the error of the final factorization step is small, this procedure retains all of the performance guarantees of expectation maximization [2, 3]. Furthermore, after each update the shared variable j will be used optimally, given the values of the factors other than f_w and f_h, because NMF minimizes the error metric defined in equation (4). Although the singular value decomposition is known to provide the optimal factorization with respect to the Frobenius norm, it yields negative entries, which are incompatible with the probabilistic interpretation of factor graphs [1]. We thus wish to restrict our factorization to matrices with positive entries, so we use NMF to split the joint factors.

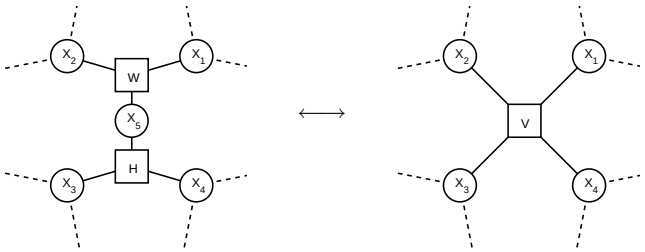

Figure 1: Schematic joining of two adjacent factors into a multifactor. Squares represent factors and circles represent variables.

A substantial literature exists on non-negative matrix factorization, and many algorithms have been proposed to optimize various error metrics [10, 11]. One of the most parsimonious algorithms is the multiplicative update rule of Lee and Seung [4], which minimizes a generalization of the Kullback-Leibler divergence:

$$D_{LS}(A||B) = \sum_{ik} \left(A_{ik} \log \frac{A_{ik}}{B_{ik}} - A_{ik} + B_{ik} \right). \tag{4}$$

This generalized KL divergence between a matrix V and its non-negative factorization $W \cdot H$ is locally minimized by the iterated application of the update rules:

$$H_{jk} \leftarrow H_{jk} \frac{\sum_i V_{ik} W_{ij} / (W \cdot H)_{ik}}{\sum_i W_{ij}} \tag{5}$$

$$W_{ij} \leftarrow W_{ij} \frac{\sum_k V_{ik} H_{jk} / (W \cdot H)_{ik}}{\sum_k H_{jk}} \tag{6}$$

Surprisingly, we can also derive equations (5) and (6) directly from the EM update given in equation (2) applied to the sub-graph consisting solely of the factors represented by W and H, if we assume that V_{ik} contains the expected marginal probability of the variable combination $\{i, k\}$ given the observed data (i.e., $\langle p(i, k|Y, \theta) \rangle$). Following equation (2), the EM update for f_h can be written as

$$f_h(j,k) \leftarrow f_h(j,k) \cdot \frac{\langle p(j,k|Y,\theta) \rangle}{p(j,k|\theta)} \tag{7}$$

$$\leftarrow f_h(j,k) \cdot \frac{\sum_i \langle p(i,k|Y,\theta) \rangle \, p(j|i,k)}{\sum_i p(i,j,k|\theta)} \tag{8}$$

$$\leftarrow H_{jk} \cdot \frac{\sum_i V_{ik} W_{ij} H_{jk} / (W \cdot H)_{ik}}{\sum_i W_{ij} H_{jk} / \sum_{ik} (W \cdot H)_{ik}} \tag{9}$$

$$\leftarrow H_{jk} \cdot \frac{\sum_i V_{ik} W_{ij} / (W \cdot H)_{ik}}{\sum_i W_{ij}} \cdot Z \tag{10}$$

where Z is analogous to equation (1). Equation (10) is a scaled version of Lee and Seung's algorithm, and since the probability distribution of a factor graph is not affected when its factors are multiplied by constants, the two are equivalent. The equivalence of the updates for W can

be demonstrated via a similar set of manipulations. The ability to rederive equations (5) and (6) from expectation maximization should assuage any concerns regarding the arbitrary nature of the generalized Kullback-Leibler divergence in equation (4).

It is important to note that the update of f_h given in equations (5) and (10), applied to the joint factor, is not equivalent to the EM update using equation (2) applied to the unmodified graph. When performing the EM update on the joint factor V using equation (2), we obtain $V = (W \cdot H)_{ik} \cdot \frac{\langle p(i,k|Y,\theta)\rangle}{p(i,k|\theta)}$, rather than just $\langle p(i,k|Y,\theta)\rangle$, as we assumed for the above derivation. The term $p(i,k|\theta)$ in the denominator depends upon all of the factors, not just f_w and f_h, so the difference is not local to the updated factors. The application of NMF after performing EM on a modified graph with a joint factor, effectively an EM update nested inside another EM update, is thus not equivalent to the original EM algorithm.

We can extend Lee and Seung's non-negative matrix factorization to cases where more than two factors are connected to the variable subsumed inside the new joint factor. In this case, V is not intrinsically the product of two matrices. Rather, when factors $A^j_{i_j,x}$, $1 \leq j \leq n$ with inputs i_j and x are all connected to variable x, the tensor $V_{i_1,i_2,\ldots,i_n}$ representing the new joint factor is defined by $V_{i_1,i_2,\ldots,i_n} = \sum_x \prod_j A^j_{i_j,x}$. The update of any single matrix A^j can be performed by rewriting this tensor as an ordinary matrix product $\sum_x A^j_{i_j,x} B^j_{x,i_1\times\cdots\times i_{j-1}\times i_{j+1}\times\cdots\times i_n}$, where the second matrix is defined as $B^j_{x,i_1\times\cdots\times i_{j-1}\times i_{j+1}\times\cdots\times i_n} = \prod_{k\neq j} A^k_{i_k,x}$. Lee and Seung's update is guaranteed to reduce the KL-like error function when performed on each of W and H separately. Since this error function only depends upon $A^j \cdot B^j = V$, the update of any factor A^j using the B^j as the second factor will indeed reduce an appropriately extended KL-like error function. This algorithm is once again a form of EM.

This method can also be trivially extended by joining a larger tree of adjacent factors into a joint factor, and then performing a series of non-negative matrix factorizations to extract the values of the original factors one-by-one. However, the size of the matrices with which we must work grows exponentially with the number of variables which feed into the joint factor, so this extension will probably be of limited use in practice.

4 Methods

For simplicity and uniformity of comparison, we investigated the relative performance of single factor and multifactor EM on factor graphs in which all factors were of degree three and all variables could take on one of four values, except where noted. For a factor graph with N input variables $Y_1, \ldots, Y_N$ of size M ($Y_i \in \{1, \ldots, M\}$), we trained the factor graph to represent a probability distribution where all input combinations satisfying $\left(\sum_{i=1}^N Y_i\right) \mod M = 0$ had equal probability $M^{-(N-1)}$, while all other input combinations had 0 probability. This is in many ways the most difficult probability distribution that can be represented by such a graphical model, as only maximal-order statistics are present. If even a single input variable Y_i is unknown, the conditional probability of the other input variables $p(Y_j|\mathbf{Y} \setminus \{Y_i, Y_j\})$ is uniform. A change in one input variable affects the probability distribution of all other variables, regardless of distance on the graph. If each hidden variable H_i is connected to exactly two factors f_L and f_R, then H_i must encode $\left(\sum_{j\in L} Y_j\right) \mod M$, where L is the set of input variables connected to H_i through f_L, or equivalently $M - \left(\sum_{j\in R} Y_j\right) \mod M$, where R is the set of input variables connected to H_i through f_R. If the hidden variables can take on only M values, then every value of each hidden variable must be used to transmit information, and the entropy of these variables must be maximal. Due to this structure of the probability distribution, all of the factors must be learned simultaneously; it is not possible to correctly learn a subset of the factors first, and then use them to learn the other factors.

To probe the performance of the two algorithms in the face of increasingly large graphs and thus increasingly complex probability distributions, we observed their convergence properties on graphs with between three and nine factors. These graphs may seem small, but the results in Figure 4(e,f) suggest that the performance on these small graphs should be similar to that on much larger graphs for probability distributions where correlations are mostly local and low-order. The exact graphs

used in the simulations consisted of the subsets of the graph shown in Figure 2 including factors 1 through n (for $3 \leq n \leq 9$), as well as all variables connected to these factors. Variables connected to only one factor are inputs, with values set according to the desired probability distribution. Both single and multifactor EM updates require knowledge of the marginal probabilities of the set of variables connected to each (multi-)factor given the parameters of the factor graph ($p(X_a = x_a|\theta)$), as well as the expected values of these marginal probabilities with the input variables sampled from the probability distribution on which the factor graph is being trained ($\langle p(X_a = x_a|Y, \theta)\rangle$). We calculated the intrinsic marginal probabilities exactly using belief propagation [1], and estimated the expected marginal probabilities given the external inputs using the average of the marginals over a batch of trials, with the inputs drawn from the desired probability distribution.

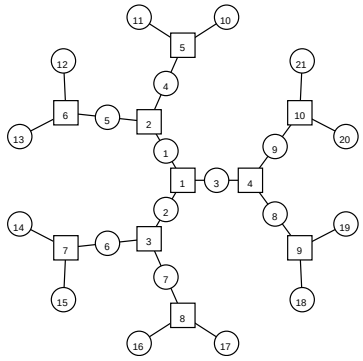

Figure 2: A factor graph with 10 factors, subsets of which were used in evaluating multifactor EM.

Since the single factors of traditional EM and the joint factors of multifactor EM have 64 and 256 distinct input combinations respectively with four values per variable, we used batches of size 1000 and 4000 (respectively) to estimate the expected marginals given the input distribution. Batch sizes were increased to 2000 and 10000 for single and multifactor EM respectively on runs where variables could take on six values. The batch size does not significantly affect the EM algorithm so long as it is sufficiently large, so we report all convergence times in terms of the number of batches. It is relevant to note that the required size of the batches is proportional to the size of the domain of the (multi-)factors, so multifactor EM will in general require larger batches and thus longer running times for the same number of batches. As will be shown in Figure 4(a,b), multifactor EM for graphs with four values per variable, for which the multifactor batches should be four times larger than the single factor batches, is between 6.7 and 371 times faster than single factor EM, with an average of 100. Figure 4(c,d) suggests that this convergence time ratio grows exponentially quickly as the variable size increases, whereas the difference in batch sizes grows only polynomially. Furthermore, the required batch size can be considerably reduced if the marginal probabilities are approximated by low-pass filtering a sequence of estimates calculated with much smaller batches [9]. Thus, there is strong evidence that the larger batches required for multifactor EM are more than compensated for by the much faster convergence time afforded by the multifactor method.

Most simulations were run for 10^8 iterations or 10^5 batches for single factor EM, and $5{\cdot}10^7$ iterations or $1.25 \cdot 10^4$ batches for multifactor EM. Single factor EM was run for $2 \cdot 10^8$ iterations for graphs with eight factors or six values per variable, and only $5 \cdot 10^7$ iterations for graphs with 3 factors. Multifactor EM was run for up to 10^8 iterations for graphs with eight or nine factors or six values per variable. Multifactor EM was run for fewer iterations only because it converged more rapidly. The finite length of our simulations creates an arbitrary but unavoidable maximum bound on the measured convergence time. Care was taken to allow the simulations to run long enough so that most runs converged well before the end of the simulation, as shown in Figure 3.

In both the single and multifactor EM algorithm, we initialized the factors according to an exponential distribution. Since factor graphs are invariant with respect to constant scaling of their parameters, the exponential distribution is scale-free in the context of factor graphs ($p_a(x \cdot b) = ae^{-a \cdot b \cdot x} \propto p_{ab}(x)$). Our algorithm thus has no parameters other than batch size and number of NMF iterations, both of which need only be chosen to be sufficiently large. In similar experiments with a slightly different probability distribution, the convergence of single factor EM was found to be dependent upon

the use of a distribution where small values are much more likely than large values [9]. Intuitively, this breaks some of the symmetries between the possible solutions. However, in our present simulations, we found that performance improved only slightly when using an exponential distribution rather than a uniform distribution for the initial parameter values (data not shown).

For the multifactor EM algorithm, we iterated equations (5) and (6) 500 times on each update. Empirically, we have found this more than sufficient to reach convergence even on random matrices. The initial values of W and H in the NMF iterations were chosen to be the previous values of the factors. This assures that the structure of the factors does not needlessly change drastically from update to update and may lead the NMF algorithm to converge more quickly, but it is not necessary. We have also had success initializing W and H to random values (drawn from the exponential distribution) in the NMF algorithm (data not shown).

5 Results

Figure 3 shows the log likelihood averaged over the last $4 \cdot 10^4$ trials versus the number of batches required to reach this likelihood for graphs of various sizes, using both single and multifactor EM. The bands of points in these graphs correspond to the various local maxima found by the algorithms. Although the EM algorithm did occasionally jump from a local maximum to the global maximum, the significant number of trials in which the EM algorithm remained trapped in a local maximum despite the large number of total training cycles suggests that these local maxima can be stable.

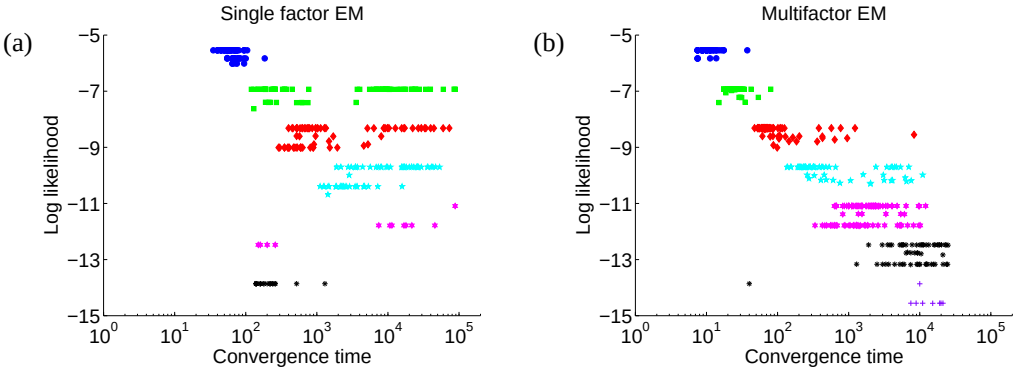

Figure 3: Number of iterations required to reach the final likelihood versus the log likelihood for (a) single factor and (b) multifactor EM. Runs in which the log likelihood failed to improve at all are not plotted. From top to bottom, blue squares = 3 factors; green circles = 4 factors; red diamonds = 5 factors; cyan five-pointed stars = 6 factors; magenta six-pointed stars = 7 factors; black asterisks = 8 factors; blue +'s = 9 factors.

Figure 4(a,b) shows the median time to convergence and the probability of convergence to the global maximum as a function of graph size for the single and multifactor EM algorithms. A run was said to have converged to the global maximum if the average log likelihood reached 99% of the actual log probability of the data points in the target distribution ($\log(M^{-(N-1)})$); the time to convergence is the number of batches required to reach this point. Runs that failed to converge were not included in the statistics. Single factor EM failed to converge even once in 74 runs on the eight factor graph with up to $2 \cdot 10^5$ batches, so its convergence time is not indicated in Figure 4(a). We report the median time to convergence rather than the average time to convergence because the median reduces the impact of outliers, readily visible in Figure 3, which otherwise obscure the trends in the graphs. Note that the average, represented by an asterisk, often falls outside the second and third quartiles of the data, indicated by the whiskers around the median. Multifactor EM converged faster and more reliably for all graph sizes tested. In particular, multifactor EM was 371 times faster on the four-factor graph, and an average of 100 times faster over all of the graph sizes.

Figure 4(c,d) shows the median time to convergence and the probability of convergence to the global maximum for the five factor graph, as a function of the size of the variables. The log of the time required to converge to the global maximum was roughly linear with respect to the size of the

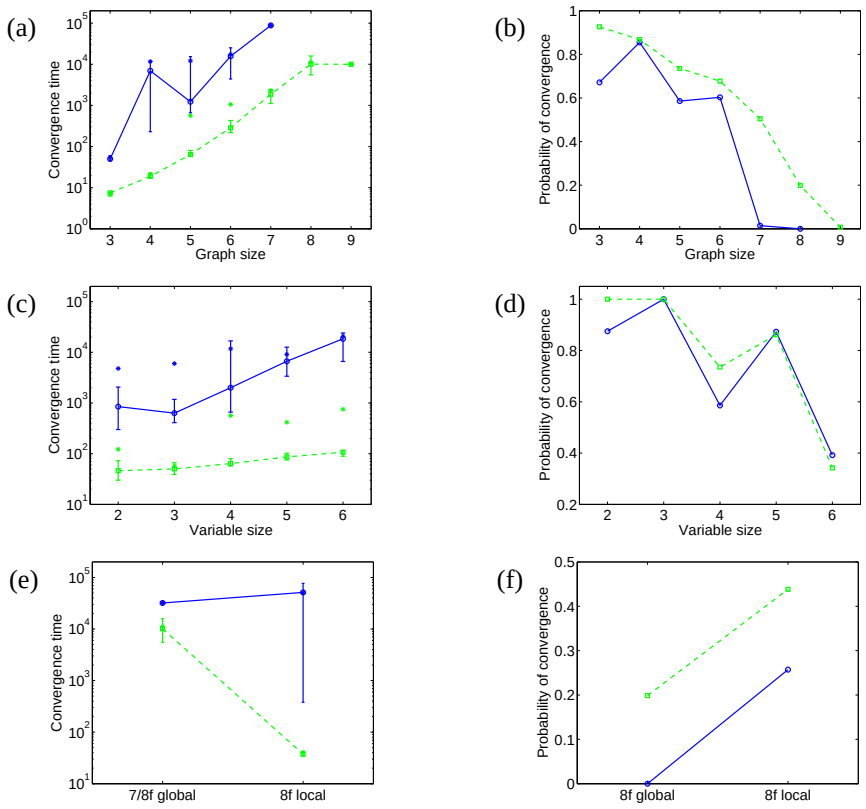

Figure 4: (a,b) Number of factors in the graph (with variable size four), (c,d) number of variables in the graph (with five factors), and (e,f) global or local correlations in the probability distribution versus (a,c,e) median convergence time and (b,d,f) probability of convergence. See text for details. Blue circles/solid line = single factor EM; green squares/dashed line = multifactor EM. In (a,c,e), the whiskers indicate the second and third quartiles of the data, while the asterisk plots the average.

variables, suggesting that the time to convergence is exponential in the variable size. The steeper growth of the log convergence time for single factor EM relative to multifactor EM implies that multifactor EM will be exponentially faster as the number of variables increases. Although the requisite batch size of multifactor EM grows faster than that for single factor EM, these terms are only polynomial in the size of the variables, so multifactor EM should be asymptotically faster. The probability of convergence did not exhibit a consistent trend with respect to variable size, presumably reflecting high-order interactions in the system.

Both algorithms failed to find the global maximum when the graphs became sufficiently large. However, as discussed above, the parity-mod-M distribution used in these experiments contained only maximal-order correlations. Most probability distributions of practical interest, such as natural visual and auditory stimuli, have considerable local correlations. For instance, visual stimuli tend to be composed of objects with smooth, continuous boundaries. Auditory stimuli usually exhibit harmonics and correlations in the power of nearby frequencies. To investigate the effect of such local structure in the probability distribution, we trained an eight-factor factor graph on the distribution with $\left(\sum_{i \in A_j} Y_i\right) \mod M = 0$, for $A_1 = \{10, 11, 12, 13\}$, $A_2 = \{14, 15, 16, 17\}$, and $A_3 = \{8, 9, 10, 11, 14, 15\}$, in the subset of the graph in Figure 2 consisting of factors 1-8 and the connected variables. As can be seen in Figure 4(e,f), this probability distribution was considerably easier to learn than the original parity-mod-M distribution, and the performance advantage of multifactor EM was once again evident. Since single factor EM never converged on the eight factor graph with the original global parity-mod-M distribution, in Figure 4(e) we plot the median convergence time for single factor EM on the seven factor graph as a lower bound.

6 Conclusions

Philosophically, this method for training trees of connected factors is similar to Wainwright's tree reparameterization [12], in that we are taking an iterative, distributed update algorithm and effectively (although not exactly) rescheduling the updates to reach convergence on a subset of the graph before updating other portions of the graph. The efficacy of multifactor EM can be understood in terms of the properties of non-negative matrix factorization [13]. In choosing the model parameters for two adjacent factors, we are effectively defining equivalence classes of input combinations to the joint factor. These equivalence classes are represented by a common internal message between the two factors. Non-negative matrix factorization can be understood as performing a biclustering operation, and thus intuitively should choose these equivalence classes efficiently.

The merging and splitting steps of multifactor EM can also provide a convenient mechanism for incrementally changing the structure of the factor graph. Factors joined into a multifactor need not be split, and the NMF algorithm can be applied to any factor in the graph to yield a larger graph composed of smaller factors. The methods described in this paper allow the parameter changes due to these topology modifications to be calculated efficiently, and obviate the need to recalculate the parameters of factors other than those directly modified.

As shown in Figure 4(e,f), EM in general and multifactor EM in particular perform much more efficiently when the probability distribution to be learned has correlations within local regions of the factor graph. This property is likely shared by probability distributions of practical interest, such as those found in sensory processing. We are currently investigating the application of multifactor EM to large loopy graphs encoding real-world probability distributions. Since the multifactor EM algorithm depends only on the marginal probabilities of the variables connected to the multifactors, regardless of inference technique, we are optimistic regarding its applicability to these problems.

References

[1] B.J. Kschischang, F.R. Frey and H.A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. on Information Theory*, 47(2):498–519, 2001.

[2] S.L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19, 1995.

[3] G.J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley-Interscience, 1996.

[4] D.D. Lee and H.S. Seung. Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, 13:556–562, 2001.

[5] M.A. Shwe, B. Middleton, D.E. Heckerman, M. Henrion, E.J. Horvitz, H.P. Lehmann, and G.F. Cooper. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base. I. The probabilistic model and inference algorithms. *Methods Inf Med*, 30(4):241–55, 1991.

[6] B.J. Frey and N. Jojic. A comparison of algorithms for inference and learning in probabilitic graphical models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(9):1392–1416, 2005.

[7] J.N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

[8] K.P. Murphy. *Dynamic Bayesian networks: Representation, inference, and learning*. PhD thesis, University of California, Berkeley, 2002.

[9] A. Anonymous. The cortex as a graphical model. Master's thesis, California Institute of Technology, 2006.

[10] D.D. Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[11] M.W. Berry, M. Browne, A.N. Langville, V.P. Pauca, and R.J. Plemmons. Algorithms and Applications for Approximate Nonnegative Matrix Factorization. *Submitted to Computational Statistics and Data Analysis*, 2006.

[12] M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Trans. on Information Theory*, 49(5):1120–1146, 2003.

[13] J.-P. Brunet, P. Tamayo, T.R. Golub, and J.P. Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the National Academy of Sciences (USA)*, 101(12):4164–4169, 2004.