An Investigation into Concurrent Expectation Propagation

David Hall and Alex Kantchelian
EECS CS Division
University of California, Berkeley
{dlwh, akant}@cs.berkeley.edu

Abstract

As statistical machine learning becomes more and more prevalent and models become more complicated and fit to larger amounts of data, approximate inference mechanisms become more and more crucial to their success. Expectation propagation (EP) is one such algorithm for inference in probabilistic graphical models. In this work, we introduce a robustified version of EP which helps ensure convergence under a relaxed memory consistency model. The resulting algorithm can be efficiently implemented on a GPU in a straightforward way. Using a 2D Ising spin glass model, we evaluate both the original EP algorithm and our robustified version in terms of convergence, if any, and precision on a classic single core processor. We also compare the naive parallelized version of the original EP algorithm against the parallelized robustified EP on both a multicore CPU and a GPU.

1 Introduction

. . . EP updates are typically defined sequentially, and there is reason to believe this is intentional. While some (XXX) have employed parallel updates when using EP, they have done so uncritically, finding that it works. Here, we undertake to explore . . .

This paper is organized as follows. First, we describe necessary background on probabilistic graphical models. Second, we describe different inference algorithms in general and Expectation Propagation in particular. Third, we motivate and present Convex Expectation Propagation. Finally, we conduct our experiments on synthetic models in a Ising spin-glass model under a variety of parameter conditions.

Figure 1: An Ising model, a commonly employed synthetic class of graphical model. XXX

2 Graphical Models

3 Inference Algorithms

4 Convex Expectation Propagation

As we discussed, EP is typically defined sequentially. While it has been infrequently used in a parallel setting (XXX), to our knowledge no one has critically examined its performance when parallelized.

Like almost all approximate inference algorithms, EP is known to have multiple local optima. (The main exceptions to our knowledge are Tree-Reweighted Belief Propagation (XXX) and LP relaxation approaches (XXX)) In practice, sequential EP seems to have little issue with these local optima, as the errors found with EP are typically much lower than with other algorithms. Moreover, the algorithm always seems to converge.

However, when doing parallel updates, EP is basically using "stale" information. That is, when doing each update, the different components have no knowledge of the changes about to be performed by the other graphs. Because different subgraphs might be locally attracted to different optima, we claim that parallel-update EP is less likely to converge, because it might alternate between these optima.

The reason that Tree-Reweighted Belief propagation is exempted from this critique is that it attempts to optimize a *convex* relaxation of the objective function. Convexity assures, among many other things, that there is at most one optimum.[1] Thus, even if the updates in the Tree-Reweighted case are done in parallel, they cannot be attracted to different local optima because they simply do not exist.

We propose, therefore, to modify EP by "convexifying" its updates. Here, we derive the algorithm starting from

[1] In principle the optimum could be at $\pm\infty$.

the modified objective function. Our proof—given sufficient background—is fairly straightforward, and follows the derivation of standard EP given in XXX closely.

4.1 The EP Objective

We begin by motivating the EP objective function. First, we use standard results to rewrite the log-partition function $A(\theta)$:

$$\begin{aligned} A(\theta) &= \log \sum_x \langle \theta, \phi(x) \rangle \\ &= \max_{\mu \in \mathcal{M}} \{ \langle \theta, \mu \rangle + H(\mu) \} \end{aligned} \tag{1}$$

with summation replaced by integration as appropriate. Here, μ is a function defining marginal distributions and H is the standard entropy functional in base e. There are two problems to computing this function directly. First, there are potentially exponentially many constraints defining the set $\mathcal{M}$. Second, the term $H(\mu)$ might not decouple in a way that admits efficient dynamic programming. We will relax both of these conditions.

First, we split the graphical model with features ϕ into a number of components. Specifically, there are the *core* features ϕ^0, and then a set of *augmented* features ϕ^i. In the Ising model, the core features would be the singleton potentials, while the ϕ^i correspond to non-overlapping subsets of the various edges. Clearly, taking all of these features together yields the original model, while taking just the core features and one of the ϕ^i yields a relaxed approximation to the full model, which we will call an *augmented* model.

The key insight behind standard EP is to approximate the entropy of the full model as follows:

$$\begin{aligned} H(\mu) &\approx H(\mu^0) + \sum_i \big(H((\mu^0, \mu^i)) - H(\mu^0) \big) \\ &\triangleq F(\mu^0, \ldots, \mu^n) \end{aligned} \tag{2}$$

Intuitively, this expression says that the full entropy function H can be approximated as the entropy of the core model, plus a set of "corrections" from each of the augmented models.

What remains is to relax the set $\mathcal{M}$ to one with fewer constraints. This can be achieved by specifying that each of the marginals (μ^0, μ^0) must live in a set $\mathcal{M}^i$ that specifies constraints that affect only the variables and potentials used in that augmented model. All together, we obtain a new objective:

$$A(\theta) \approx \max_{(\mu^0, \mu^i) \in \mathcal{M}^i} \{ \langle \theta^0, \mu^0 \rangle + \sum_i \langle \theta^i, \mu^i \rangle + F(\mu^0, \ldots, \mu^n) \} \tag{3}$$

By following a derivation similar to what we present in the next section, we can arrive at the EP updates described previously.

4.2 Convex EP

The entropy functional is well-known to be convex, but the approximate entropy functional F defined in Equation 2 is in general not convex (XXX cite). However, if we were to restrict ourselves to just one augmented model, then F would be convex, because

$$\begin{aligned} F(\mu^0, \mu^i) &= H(\mu^0) + \big(H((\mu^0, \mu^i)) - H(\mu^0) \big) \\ &= (H((\mu^0, \mu^i)) \end{aligned} \tag{4}$$

Moreover, if we exploit the fact that any convex combination of convex functions is convex, we obtain:

$$\begin{aligned} G(\mu^0, \mu^1, \ldots, \mu^n) &= \sum_i \rho_i \left(H(\mu^0) + \big(H((\mu^0, \mu^i)) - H(\mu^0) \big) \right) \\ &= H(\mu^0) + \sum_i \rho_i \left(H((\mu^0, \mu^i)) - H(\mu^0) \right) \end{aligned} \tag{5}$$

is convex. Here ρ_i parameterizes the convex combination. In general, we require that $\sum_i \rho_i = 1$, though for specific cases (such as the Ising model) we can employ other choices that maintain convexity and might be more accurate. Hence, we have a new approximate objective:

$$A(\theta) \approx \max_{(\mu^0, \mu^i) \in \mathcal{M}^i} \{ \langle \theta^0, \mu^0 \rangle + \sum_i \langle \theta^i, \mu^i \rangle + G(\mu^0, \ldots, \mu^n) \} \tag{6}$$

What remains is to actually derive the algorithm. To do that, we first employ a mathematical sleight-of-hand. We define new marginals η^i defined over the same space as μ^0 and constrain them to equal μ^0. We modify G as follows:

$$\begin{aligned} &G(\mu^0, (\eta^0, \mu^1), \ldots, (\eta^n, \mu^n)) \\ &\qquad = H(\mu^0) + \sum_i \rho_i \left(H((\eta^i, \mu^i)) - H(\eta^i) \right) \end{aligned}$$

This trick allows us to relax the constraint that $\eta_i = \mu_i$ and enforce it with Lagrange multipliers.

Given this modified G, we define the Lagrangian:

$$\begin{aligned} &L(\mu^0, (\eta^i, \mu^i), \lambda^i) \\ &\quad = \langle \theta^0, \mu^0 \rangle + \sum_i \langle \theta^i, \mu^i \rangle + G(\mu^0, (\eta^1, \mu^1), \ldots, (\eta^n, \mu^n)) \\ &\qquad + \sum_i \langle \lambda^i, \mu^0 - \eta^i \rangle \rangle \end{aligned} \tag{7}$$

Taking gradients and setting them to 0 we obtain:

$$\begin{aligned}
0 &= \nabla_{\mu^0} L(\cdot) \\
&= \theta^0 + \nabla H(\mu^0) + \sum_i \lambda_i \\
&= \theta^0 - \log \mu^0 + \sum_i \lambda_i + \text{const} \\
\log \mu^0 &= \theta^0 + \sum_i \lambda_i + \text{const} \\
\mu^0(x) &\propto \exp(\left\langle \theta^0 + \sum_i \lambda_i, \phi^0(x) \right\rangle) \\
&\triangleq q(x)
\end{aligned}$$

Because μ^0 is a probability distribution, μ^0 is a distribution that has the same form as the base distribution.

Taking derivatives with respect to each (η^i, μ^i) we have:

$$\begin{aligned}
0 &= \nabla_{(\eta^i,\mu^i)} L(\cdot) \\
&= \theta^i + \rho_i \nabla H\left((\eta^i, \mu^i)\right) - \rho_i \nabla H(\eta^i) - \lambda_i \\
&= \theta^i - \rho_i \log \left(\overset{\mu^i}{\eta^i}\right) + \rho_i \log \eta^i - \lambda_i \\
\rho_i \log \left(\overset{\mu^i}{\eta^i}\right) &= \theta^i - \lambda_i + \rho_i \log \eta^i
\end{aligned}$$

Using the fact that $\eta^i = \mu^0$ at the optimum and doing some algebra, we have:

$$\begin{aligned}
\rho_i \log \left(\overset{\mu^i}{\eta^i}\right) &= \theta^i - \lambda_i + \rho_i \log \mu^0 \\
\rho_i \log \left(\overset{\mu^i}{\eta^i}\right) &= \theta^i - \lambda_i + \rho_i \cdot (\theta^0 + \sum_\ell \lambda_\ell) \\
\log \left(\overset{\mu^i}{\eta^i}\right) &= \frac{1}{\rho}_i \theta^i + \theta^0 + \sum_{\ell \neq i} \lambda_\ell + (1 - \frac{1}{\rho}_i) \lambda_i \\
q_i(x) \propto \exp(&\left\langle \theta^0 + \sum_{\ell \neq i} \lambda_\ell + (1 - \frac{1}{\rho}_i) \lambda_i, \phi^0(x) \right\rangle \\
&+ \frac{1}{\rho}_i \langle \theta^i, \phi^i(x) \rangle)
\end{aligned}$$

This implies a procedure very similar to EP that is actually a special instance of power EP(XXX cite). Identifying $\tilde{f}_i(x) = \exp(\langle \lambda_i, \phi^0(x) \rangle)$ and $f_i(x) = \exp(\langle \theta^i, \phi^i(x) \rangle)$ we can define convex EP as:

1. Initialize $q(x) \propto f_0(x) \prod_i \tilde{f}_i(x)$
2. Create $q^{\setminus i}(x) = q(x) f(x)^{1/\rho_i - 1}$
3. Minimize KL $\left(q^{\setminus i}(x) f(x)^{\frac{1}{\rho}_i} || q(x)\right)$
4. Update $\tilde{f}_i(x) = q(x)/q^{\setminus i}(x)$

Note that this algorithm has a kind of hysteresis: powers of the f_i are subtracted out rather than all of them. This is an interesting effect.

5 Experiments

6 Conclusion