Stochastic Belief Propagation on Trees

John Halloran
EE596a Final Project, Winter 2013

Abstract

For exact inference, Belief Propagation (BP) on trees requires $O(Td^2)$ operations, where $T$ is the number of variables and $d$ is the cardinality of all hidden variables. This quadratic complexity becomes prohibitive when $d$ is large. Stochastic Belief Propagation (SBP) [1] is an approximate inference algorithm which utilizes subtle changes to original BP in order to achieve $O(1/\sqrt{\tau})$ error in $O(\tau Td)$ time over all trees. The SBP algorithm is described herein and illustrated for HMMs, and interesting avenues for further work on SBP are discussed.

1 Introduction and Notation

It is well known that for trees consisting of $T$ hidden variables each with cardinality $d \in \mathbb{N}$, Belief Propagation (BP) requires $O(Td^2)$ time and $O(Td)$ memory for exact inference. As $d$ grows, this computational complexity becomes prohibitive. While many approximate techniques exist, such as the various forms of beam-width pruning, such techniques do not offer theoretical guarantees for general graphs. While approximate, variational techniques exist [2], such techniques do not guarantee almost sure convergence to the BP fixed point. Stochastic Belief Propagation (SBP) [1] is an approximate inference algorithm which is a simple variant of BP and offers theoretical guarantees for all trees (in fact, even for Loopy Belief Propagation over non-trees which fit several constraints, but discussion is limited to trees herein). Indeed, as we will see, SBP is such a simple variant of BP, many exact inference BP engines may adapt it with minimal change to the BP implementation itself. SBP consists of iterating BP many times, similar to Loopy Belief Propagation (LBP). Unlike LBP, however, only a single damped column of an edge potential is used to update a message, per iteration, in SBP. Thus, in $\tau$ iterations, SBP achieves $O(1/\sqrt{\tau})$ Maximum Absolute Distortion (MAD) error in $O(\tau Td)$ time.

Consider a Markov Random Field (MRF) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes corresponding to the elements of the random vector $Q = (Q_1, Q_2, \ldots, Q_n)$, such that $\mathcal{G}$ is a tree. For the vertex set $[n] = \{1, 2, \ldots, n\}$, let a variable $u \in [n]$ serve as an index such that $Q_u \in \mathcal{V}$. Denote the neighbors of node $Q_u$ as $\mathcal{N}(u)$. Denote the edge potentials of $\mathcal{G}$ as $\psi_{u,v}(Q_u, Q_v) = \psi_{u,v}, \forall (u, v) \in \mathcal{E}$, and node potentials of $\mathcal{G}$ as $\psi_u, \forall u \in \mathcal{V}$. In order to keep track of the BP messages, denote the set of directed edges in $\mathcal{G}$ as $\overrightarrow{\mathcal{E}}$, such that for $u, v \in [n], u \leftarrow v \in \overrightarrow{\mathcal{E}}$, $m_{uv}$ denotes the message passed from node $Q_v$ to $Q_u$. 
Denote the collection of all BP messages flowing in the tree as $m = \{m_{uv}\}_{u \leftarrow v \in \mathcal{E}}$. For analysis purposes, we view $m \in \mathbb{R}^{\mathcal{E}}$ as a column vector consisting of the concatenation of all messages. We will consider running BP multiple iterations, so let $m^t$ be the collection of updated messages at iteration $i$. Define a function $F(m^t) = m^{t+1}$, which updates all messages given the previous iteration of messages. The goal of BP, then, is to find an $m^*$ such that $F(m^t) = m^*$, where $m^*$ is called a BP fixed point. For trees, $F(m^0) = m^1 = m^*$, and $m^*$ is unique.

2 Stochastic Belief Propagation for Hidden Markov Models

Consider the HMM consisting of $T$ frames depicted in figure 1(a) with potentials $p(Q_t|Q_{t-1})$, $p(x_t|Q_t)$, for $t \in [T]$, where $Q_1 = \emptyset$. Consider the problem of calculating posteriors $p(Q_t|X_{1:t} = x_{1:t})$. To ease discussion of the SBP algorithm for HMMs, we collapse the observed variables into their hidden parents, such that the potentials of the resulting chain (depicted in figure 1(b)) are $\psi_{t-1,t}(Q_{t-1} = i, Q_t = j) = p(Q_t = j|Q_{t-1} = i)p(x_t|Q_t = j) \forall i, j \in [d]$. We may compute the posteriors using the values of the $\alpha$ recursion, $\alpha_t(j) = p(Q_1 = j)p(x_1|Q_1 = j)$ and

$$\alpha_{t+1}(j) = \sum_i \alpha_t(i)p(Q_{t+1} = j|Q_t = i)p(x_{t+1}|Q_{t+1} = j)$$

and we have $p(Q_t|x_{1:t}) = \frac{\alpha_{t+1}}{\sum_j \alpha_{t+1}(j)}$. Thus, $\alpha_{t+1} = \psi_{t,t+1}\alpha_t$, so that $\alpha_{t+1}$ is simply a linear combination of the columns of $\psi_{t,t+1}$ requiring $O(d^2)$ operations. Furthermore, when $\alpha_t$ is appropriately normalized, $\alpha_{t+1}$ is simply an expectation. Conditioned on $\alpha_t$ and the elements of $\psi_{t,t+1}$ itself, different vectors of $\psi_{t,t+1}$ may dominate the computation of $\alpha_{t+1}$. With this intuition, we may consider iteratively updating $\alpha_{t+1}$ using a single column of $\psi_{t,t+1}$ chosen with some probability. Denote the $\alpha_{t+1}$ value after iteration $i$ as $\alpha_{i+1}^i$. Such a sampling scheme reduces the complexity of a single update to $O(d)$ operations, with the sacrifice of exactness. However, one might imagine that iteratively, $\alpha_{i+1}^i$ might approach $\alpha_{t+1}$, and the rate at which it does, as well as the error one is willing to incur, would allow us to approximate $\alpha_{t+1}$ in $(iTd)$ operations, where $id \ll d^2$.

![HMM(left) and equivalent Markov Chain(right)](figure.png)
2.1 Stochastic Belief Propagation

Define the matrix $\Gamma_{uv}$ such that, for $j = 1, \ldots, d$, $\Gamma_{uv}(i,j) = \psi_{uv}(i,j)$, and the vector $\theta_{uv}(j) = \left(\sum_{i=1}^{d} \psi_{uv}(i,j)\right) \psi_u$. Note that $\Gamma_{uv}$ is simply a column normalized variant of $\psi_{uv}$, and $\theta_{uv}(j)$ is a prior for the column vector $\psi_{uv}(i,j)$ which influences which column is chosen to update a message during an iteration. $\Gamma_{uv}$ and $\theta_{uv}$ are assumed to be computed offline, requiring $O(d^2)$ operations.

**Algorithm 1 Stochastic Belief Propagation**

1: Initialize $m^0 \in \mathcal{R}_d^D$
2: for $i = 0, 1, \ldots, \tau$ do
3:     for $u \leftarrow v \in \mathcal{E}$ do
4:         For $j \in [d]$, compute $M_{uv}^i = \prod_{w \in \mathcal{N}(u) \setminus \{v\}} m_{uw}^i(j)$
5:         Construct distribution $p_{uv}^i(j) \propto M_{uv}^i(j)\beta_{vu}(j)$, for $j \in [d]$
6:         Pick random index $j_{iu}^{i+1} \in [d] \sim p_{uv}^i$
7:         For $\lambda^i \in (0, 1)$, update $m_{uv}^{i+1} = (1 - \lambda^i) m_{uv}^i + \lambda^i \Gamma_{vu}(\cdot, j_{iu}^{i+1})$
8:     end for
9: end for

Line 4 of algorithm 1 is simply the gathering of a node’s incoming messages, as in normal BP. From line 5, $p_{uv}^i(j)$ is the distribution from which we randomly generate the index of the column vector to update the current message $m_{vu}^{i+1}$ in line 7. Note that $p_{uv}^i(j)$ is a function of both the incoming messages and the prior on the edge potentials, $\theta_{uv}$, and may be computed in $O(d)$ time per message per iteration. $\lambda^i$ is a learning rate analogous to gradient based methods. In [1], it is proven that with a learning rate $\lambda^i = 1/(t+1)$ and for some constant vector $C$, in $\tau$ iterations SBP achieves error

$$E[|m^\tau - m^*|] \leq \frac{1}{\sqrt{\tau}}C \Rightarrow \max_{i \in [d]} E[|m^\tau(j) - m^*(j)|] \leq \frac{c}{\sqrt{\tau}}$$

for some constant $c$.

Note that this gives us a bound on the expected MAD value, i.e. the largest absolute difference between a message element’s exact and approximated values.

For the HMM $\alpha$ recursion and in the interest of computing $p(Q_t|x_{1:t})$, SBP simplifies in several ways. Firstly, the messages we keep track of are simply the alpha values, such that $m^i = [(\alpha_1^i)^\dagger \ (\alpha_2^i)^\dagger \ \ldots \ (\alpha_t^i)^\dagger]^\dagger$(note that $\alpha_1^i = \alpha_0^i$), where $\dagger$ is the transpose operator. As such, the incoming messages on line 4 simplifies to $M_{uv}^i = \alpha_{u,t+1}^i$, for $u = t, v = t + 1$. Finally, we need only perform this computation for edges $t + 1 \leftarrow t \in \mathcal{E}, t \in [d-1]$. 

3
2.2 SBP results for computing posteriors in an HMM

SBP results are displayed in figure 2 for an HMM with $T = 100, d \in \{128, 256, 512, 1028\}$. The observations were sampled from a Gaussian, and the assumed conditional distribution over the observed variables was Gaussian. The HMM transition matrix, initial distribution, and conditional Gaussian parameters were all randomly generated. The MAD is defined as $\max_{i \in [d]} \mathbb{E}||m^\tau(j) - m^*(j)||$ and normalized Mean Squared Error (MSE) is defined as $\frac{||m^\tau - m^*||_2^2}{||m^*||_2^2}$. The exact BP posteriors were calculated by running the standard HMM $\alpha$ recursion, which allows us to compare SBP runtime 2(c). Figure 2(b) shows that in only 10 iterations, SBP achieves normalized MSE no greater than 3 significant digits. For $d = 1024$, this results in a 3 order of magnitude speed-up.

![Graphs showing average MAD, normalized MSE, and inference time](image)

Figure 2: $T = 100, d \in \{128, 256, 512, 1028\}$. Results averaged over 5 runs of SBP.

3 Conclusions and Discussion

SBP was defined and examined for the case of computing posteriors in an HMM, and shown to quickly approach the exact posterior values in terms of MAD and normalized MSE. The empirical success and theoretical guarantees of such a simple BP variant begs the question of whether a Viterbi analogue exists, an analogue which has yet to be found. Furthermore, the learning rate $\lambda^t = 1/(t + 1)$ and proof of convergence rate in [1] for trees are suspiciously of a similar flavor to the proof of convergence for gradient descent [3]. It would be interesting to show that SBP is equivalent to gradient descent.

References

