Block Belief Propagation for Parameter Learning in Markov Random Fields

You Lu
Department of Computer Science
Virginia Tech
Blacksburg, VA
you.lu@vt.edu

Zhiyuan Liu
Department of Computer Science
University of Colorado Boulder
Boulder, CO
zhiyuan.liu@colorado.edu

Bert Huang
Department of Computer Science
Virginia Tech
Blacksburg, VA
bhuang@vt.edu

Abstract

Traditional learning methods for training Markov random fields require doing inference over all variables to compute the likelihood gradient. The iteration complexity for those methods therefore scales with the size of the graphical models. In this paper, we propose *block belief propagation learning* (BBPL), which uses block-coordinate updates of approximate marginals to compute approximate gradients, removing the need to compute inference on the entire graphical model. Thus, the iteration complexity of BBPL does not scale with the size of the graphs. We prove that the method converges to the same solution as that obtained by using full inference per iteration, despite these approximations, and we empirically demonstrate its scalability improvements over standard training methods.

Introduction

Markov random fields (MRFs) and conditional random fields (CRFs) are powerful classes of models for learning and inference of factored probability distributions (Koller and Friedman 2009; Wainwright and Jordan 2008). They have been widely used in tasks such as structured prediction (Taskar, Guestrin, and Koller 2004) and computer vision (Nowozin and Lampert 2011). Traditional training methods for MRFs learn by maximizing an approximate maximum likelihood. Many such methods use variational inference to approximate the crucial partition function.

With MRFs, the gradient of the log likelihood with respect to model parameters is the marginal vector. With CRFs, it is the expected feature vector. These identities suggest that each iteration of optimization must involve computation of the full marginal vector, containing the estimated marginal probabilities of all variables and all dependent groups of variables. In some applications, the number of variables can be massive, making traditional, full-inference learning too expensive in practice. This problem limits the application of MRFs in modern data science tasks.

In this paper, we propose *block belief propagation learning* (BBPL), which alleviates the cost of learning by computing approximate gradients with inference over only a small block of variables at a time. BBPL first separates the Markov network into several small blocks. At each iteration of learning, it selects a block and computes its marginals. It approximates the gradient with a mix of the updated and the previous marginals, and it updates the parameters of interest with this gradient.

Related Work

Many methods have been developed to learn MRFs. In this section, we cover only the BP-based methods.

Mean-field variational inference and belief propagation (BP) approximate the partition function with non-convex entropies, which break the convexity of the original partition function. In contrast, convex BP (Globerson and Jaakkola 2007; Heskes 2006; Schwing et al. 2011; Wainwright, Jaakkola, and Willsky 2005; Wainwright 2006) provides a strongly convex upper bound for the partition function. This strong convexity has also been theoretically shown to be beneficial for learning (London, Huang, and Getoor 2015). Thus, our BBPL method uses convex BP to approximate the partition function.

Regarding inference, some methods are developed to accelerate computations of the beliefs and messages. Stochastic BP (Noorshams and Wainwright 2013) updates only one dimension of the messages at each inference iteration, so its iteration complexity is much lower than traditional BP. Distributed BP (Schwing et al. 2011; Yin and Gao 2014) distributes and parallelizes the computation of beliefs and messages on a cluster of machines to reduce inference time. Sparse-matrix BP (Bixler and Huang 2018) uses sparse-matrix products to represent the message-passing indexing, so that it can be implemented on modern hardware. However, to learn MRF parameters, we need to run these inference algorithms for many iterations on the whole network until convergence at each parameter update. Thus, these methods are still impacted by the network size.

Regarding learning, many learning frameworks have been proposed to efficiently learn MRF parameters. Some approaches use neural networks to directly estimate the messages (Lin et al. 2015; Ross et al. 2011). Methods that truncate message passing (Domke 2011; 2013; Stoyanov, Ropson, and Eisner 2011) redefine the loss function in terms of the approximate inference results obtained after message passing is truncated to a small number of iterations. These methods can be inefficient because they require running in-

ference on the full network for a fixed number of iterations or until convergence. Moreover, they yield non-convex objectives without convergence guarantees.

Some methods restrict the complexity of inference on a subnetwork at each parameter update. Lifted BP (Kersting, Ahmadi, and Natarajan 2009; Singla and Domingos 2008) makes use of the symmetry structure of the network to group nodes into super-nodes, and then it runs a modified BP on this lifted network. A similar approach (Ahmadi, Kersting, and Natarajan 2012) uses a lifted online training framework that combines the advantages of lifted BP and stochastic gradient methods to train Markov logic networks (Richardson and Domingos 2006). These lifting approaches rely on symmetries in relational structure—which may not be present in many applications—to reduce computational costs. They also require extra algorithms to construct the lifted network, which increase the difficulty of implementation. Piecewise training separates a network into several possibly overlapping sets of pieces, and then uses the piecewise pseudolikelihood as the objective function to train the model (Sutton and McCallum 2009). Decomposed learning (Samdani and Roth 2012) uses a similar approach to train structured support vector machines. However, these methods need to modify the structure of the network by decomposing it, thus changing the objective function.

Finally, inner-dual learning methods (Bach et al. 2015; Hazan and Urtasun 2010; Hazan, Schwing, and Urtasun 2016; Meshi et al. 2010; Taskar et al. 2005) interleave parameter optimization and inference to avoid repeated inferences during training. These methods are fast and convergent in many settings. However, in other settings, key bottlenecks are not fully alleviated, since each partial inference still runs on the whole network, causing the iteration cost to scale with the network size.

Contributions

In contrast to many existing approaches, our proposed method uses the same convex inference objective as traditional learning, providing the same benefits from learning with a strongly convex objective. BBPL thus scales well to large networks because it runs convex BP on a block of variables, fixing the variables in other blocks. This block update guarantees that the iteration complexity does not increase with the network size. BBPL preserves communication across the blocks, allowing it to optimize the original objective that computes inference over the entire network at convergence. The update rules of BBPL are similar in form to full BP learning, which makes it as easy to implement as traditional MRF or CRF training. Finally, we theoretically prove that BBPL converges to the same solution under mild conditions defined by the inference assumption. Our experiments empirically show that BBPL does converge to the same optimum as full BP.

Background

In this section, we introduce notation and background knowledge directly related to our work.

Convex Belief Propagation for MRFs

Let $x = [x_1, ..., x_n]$ be a discrete random vector taking values in $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$, and let $G = (V, E)$ be the corresponding undirected graph, with the vertex set $V = \{1, ..., n\}$ and edge set $E \subset V \times V$. Potential functions $\theta_s : \mathcal{X}_s \to \mathcal{R}$ and $\theta_{uv} : \mathcal{X}_u \times \mathcal{X}_v \to \mathcal{R}$ are differentiable functions with parameters we want to learn. The probability density function of a pairwise Markov random field (Wainwright and Jordan 2008) can be written as

$$p(x|\theta) = \exp\{\sum_{s \in V} \theta_s(x_s) + \sum_{(u,v) \in E} \theta_{uv}(x_u, x_v) - A(\theta)\}.$$

The log partition function

$$A(\theta) = \log \sum_X \exp\{\sum_{s \in V} \theta_s(x_s) + \sum_{(u,v) \in E} \theta_{uv}(x_u, x_v)\} \tag{1}$$

is intractable in most situations when G is not a tree. One can approximate the log partition function with a convex upper bound:

$$B(\theta) = \max_{\tau \in \mathbb{L}(G)} \{\langle \theta, \tau \rangle - B^*(\tau)\}, \tag{2}$$

where

$$\begin{aligned} \theta &= \{\theta_s | s \in V\} \cup \{\theta_{uv} | (u,v) \in E\}, \\ \tau &= \{\tau_s | s \in V\} \cup \{\tau_{uv} | (u,v) \in E\}, \\ \mathbb{L}(G) &:= \{\tau \in \mathbb{R}_+^d | \sum_{x_s} \tau_s(x_s) = 1, \\ &\qquad \sum_{x_v} \tau_{uv}(x_u, x_v) = \tau_u(x_u)\}. \end{aligned}$$

The vector τ is called the pseudo-marginal or belief vector. Specifically, τ_s is the unary belief of vertex s, and τ_{uv} is the pairwise belief of edge (u, v). The *local marginal polytope* $\mathbb{L}$ restricts the unary beliefs to be consistent with their connected pairwise beliefs. We consider a variant of belief propagation where $B^*(\tau)$ is strongly convex and has the following form:

$$B^*(\tau) = \sum_{s \in V} \rho_s H(\tau_s) + \sum_{(u,v) \in E} \rho_{uv} H(\tau_{uv}),$$

where ρ_s and ρ_{uv} are parameters known as counting numbers, and $H(.)$ is the entropy.

Equation 2 can be solved via convex BP (Meshi et al. 2009; Yedidia, Freeman, and Weiss 2005). Let λ_{uv} be the message from vertex u to vertex v. The update rules of messages and beliefs are as follows:

$$\lambda_{uv} = \rho_{uv} \log \sum_u \exp\{\frac{1}{\rho_{uv}}(\theta_{uv} - \lambda_{vu}) + \log \tau_u\}, \tag{3}$$

where

$$\tau_u \propto \exp\{\frac{1}{\rho_u}(\theta_u + \sum_{v \in N(u)} \lambda_{vu})\}, \tag{4}$$

and

$$\tau_{uv} \propto \exp\{\frac{1}{\rho_{uv}}(\theta_{uv} - \lambda_{uv} - \lambda_{vu}) + \log \tau_u \tau_v\}. \tag{5}$$

Other forms of convex BP, such as tree-reweighted BP (Wainwright, Jaakkola, and Willsky 2005), can also be used in our approach. Convex BP does not always converge on loopy networks. However, under some mild conditions, it is guaranteed to converge and can be a good approximation for general networks (Roosta, Wainwright, and Sastry 2008).

Learning Parameters of MRFs

In this subsection, we introduce traditional training methods for fitting MRFs to a dataset via a combination of BP and a gradient-based optimization. The learning algorithm is given a dataset with N data points, i.e., $w_1, ..., w_N$. It then learns θ by minimizing the negative log-likelihood:

$$
\begin{aligned}
L(\theta) &= -\frac{1}{N} \sum_n \log p(w_n|\theta) \\
&= -\frac{1}{N} \sum_n \theta^T w_n + A(\theta) \\
&\approx -\theta^T \bar{w} + B(\theta) \\
&= -\theta^T \bar{w} + \max_{\tau \in \mathbb{L}(G)} \{\langle \theta, \tau \rangle - B^*(\tau)\}, \quad (6)
\end{aligned}
$$

where $\bar{w} = 1/N \sum_n w_n$.

Using $B(\theta)$ as the tractable approximation of $A(\theta)$, the traditional learning approach is to minimize $L(\theta)$ using gradient-based methods. Let θ_t be the parameter vector at iteration t, and let τ_t^* be the optimized τ corresponding to θ_t. Then gradient learning is done by iterating

$$
\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta L(\theta_t), \quad (7)
$$

where

$$
\nabla_\theta L(\theta_t) = -\bar{w} + \tau_t^*, \quad (8)
$$

and α_t is the learning rate. The traditional parameter learning process is described in Algorithm 1.

Algorithm 1 Parameter learning with full convex BP

1: Initialize θ_0.
2: While θ has not converged
3:     While λ has not converged
4:         Update τ with Equation 4.
5:         Update λ with Equation 3.
6:     end
7:     $\nabla L(\theta_t) = -\bar{w} + \tau_t^*$
8:     $\theta_{t+1} = \theta_t - \alpha_t \nabla L(\theta_t)$
9: end

Traditional parameter learning requires running inference on each full training example per gradient update, so we refer to it as *full BP learning*. These full inferences cause it to suffer scalability limitations when the training data includes large MRFs. The goal of our contributions is to circumvent these limitations.

Block Belief Propagation Learning

In this section, we introduce our proposed method, which has a much lower iteration complexity than full BP learning and is guaranteed to converge to the same optimum as full convex BP learning.

Algorithm Description

The full BP learning method in Algorithm 1 does not scale well to large networks. It needs to perform inference on the full network at each iteration, i.e., Step 2 to Step 5 in Algorithm 1. The iteration complexity depends on the size of network. When the network is large—e.g., a network with at least tens of thousands of nodes—the dimensions of τ and λ are also large. Updating all their dimensions in each gradient step creates a significant computational inefficiency.

To address this problem, we develop *block belief propagation learning* (BBPL). BBPL only needs to do inference on a subnetwork at each iteration, which means that, for templated models, the iteration complexity of BBPL does not depend on the size of the network. For non-templated models, the iteration complexity has a greatly reduced dependency on the network size. Moreover, we can prove the convergence of BBPL, guaranteeing that it will converge to the same optimum as full BP learning. Finally, BBPL's update rules are analogous to those of full BP learning, so it is as easy to implement as full BP learning.

BBPL first separates the vertex set V into D subsets $V_1, ..., V_D$. Then it separates the edge set E into D subsets of edges incident on the corresponding vertex subsets, i.e., $E_i = \{(u, v)|(u, v) \in E \text{ and } (u \in V_i \text{ or } v \in V_i)\}$. Thus, we have that $V = V_1 \cup V_2 \cup ... \cup V_D$, and $E = E_1 \cup E_2 \cup ... \cup E_D$. For shorthand, let $F_i = (E_i, V_i)$ denote the ith subnetwork.

At iteration t, BBPL selects a subgraph F_i and only updates each τ_u if $u \in V_i$, and τ_{uv} and only λ_{uv} if $(u, v) \in E_i$. It updates this block of messages and beliefs via belief propagation—i.e., Equations 3, 4, and 5—holding all other messages and beliefs fixed until convergence. Finally, BBPL uses the updated block of beliefs to update θ by computing an approximate gradient. Our empirical results show that either selecting subgraphs randomly or sequentially can lead to convergence, but sequentially selecting subgraphs can make the algorithm converge faster.

With a slight abuse of notation, let F_t be the subnetwork selected at iteration t. Let $\tau_t^{(F_t)}$ and $\lambda_t^{(F_t)}$ be the submatrices of τ_t and λ_t, respectively, that correspond to F_t. Let U_{F_t} be a projection matrix that projects the parameter from $\mathbb{R}^{|F_t|}$ to $\mathbb{R}^d$. The update rules for τ_t and λ_t are

$$
\tau_t = \tau_{t-1} + U_{F_t}(\tau_t^{(F_t)} - \tau_{t-1}^{(F_t)}), \quad (9)
$$

and

$$
\lambda_t = \lambda_{t-1} + U_{F_t}(\lambda_t^{(F_t)} - \lambda_{t-1}^{(F_t)}). \quad (10)
$$

After computing τ_t and λ_t, BBPL updates the parameters using the approximate gradient:

$$
g(\theta_t) = -\bar{w} + \tau_t. \quad (11)
$$

Note that the only difference between $g(\theta_t)$ and $g(\theta_{t-1})$ is the term $\tau_t^{(F_t)} - \tau_{t-1}^{(F_t)}$. Thus, given $g(\theta_{t-1})$, we can efficiently update our gradient estimate with

$$
g(\theta_t) = g(\theta_{t-1}) + U_{F_t}(\tau_t^{(F_t)} - \tau_{t-1}^{(F_t)}).
$$

This equation implies that computation of the gradient also does not depend on the size of the network. Changing the

gradient at the entries where the block marginal update was performed can be done in place. Thus, the complexity of the whole inference process and gradient computation depends only on the size of the subnetwork. The only step that requires time complexity that scales with the network size is the actual parameter update using the gradient, which we will later alleviate when using templated models. The complete algorithm is listed in Algorithm 2.

Algorithm 2 Parameter estimation with block BP

1: Separate g into M subnetwork, i.e., $F_1, ..., F_M$.
2: While θ has not converged
3:     Select a subgraph F_t.
4:     While λ^{F_t} has not converged
5:         Update $\tau_t^{F_t}$ with Equation 4.
6:         Update $\lambda_t^{F_t}$ with Equation 3.
7:     end
8:     $g(\theta_t) = g(\theta_{t-1}) + U_{F_t}(\tau_t^{(F_t)} - \tau_{t-1}^{(F_t)})$.
9:     $\theta_{(t+1)} = \theta_{(t)} - \alpha_t g(\theta_t)$.
10: end

Convergence Analysis

In this subsection, we theoretically prove the convergence of BPPL. We first rewrite the learning problem as follows:

$$\min_{\theta} \max_{\tau \in \mathbb{L}(G)} -\theta^T \bar{w} + \theta^T \tau - B^*(\tau). \tag{12}$$

Equation 12 is a convex-concave saddle-point problem. We use (θ^*, τ^*) to represent its saddle point. Its corresponding primal problem is

$$\min_{\theta} L(\theta) = -\theta^T \bar{w} + B(\theta), \tag{13}$$

where $B(\theta)$ is defined in Equation 2. Equation 12 and Equation 13 have the same optimal θ.

Thus, we can prove the convergence of BBPL following the general framework for proving the convergence of saddle-point optimizations (Du and Hu 2018). We prove that under a mild assumption, i.e., Assumption 1, BBPL has a linear convergence rate.

Assumption 1 *When θ has not converged, the block coordinate update, i.e., Equation 9, satisfies the following inequality:*

$$||\tau_{t+1} - \tau_{t+1}^*|| \leq (1 - c)||\tau_t - \tau_{t+1}^*||,$$

where $0 < c < 1$.

Informally, we can interpret Assumption 1 to mean that, at iteration $t + 1$, the vector τ_{t+1} with new block $\tau_{t+1}^{(F_t)}$ will get closer to the optimum τ_{t+1}^*. This assumption is easy to satisfy when θ has not converged. Since the block $\tau_{t+1}^{(F_t)}$ of τ_{t+1} is updated with respect to θ_{t+1}, and τ_{t+1}^* is the true optimum with respect to θ_{t+1}, τ_{t+1} should be closer to τ_{t+1}^*. When θ converges, block BP becomes a block coordinate update method. Based on claims shown by Schwing et al. (2011), τ_t converges to τ^*.

The following theorem establishes the linear convergence guarantee of Algorithm 2.

Theorem 1 *The primal of BBPL, i.e., Equation 13, is β-strongly convex and η-smooth. When we use BBPL to learn the parameters, suppose BBPL satisfies Assumption 1. Define the Lyapunov function*

$$P_t = ||\theta_t - \theta^*|| + \gamma||\tau_t - \tau_t^*||.$$

When $0 < \alpha_t \leq \min\{\frac{c\beta}{2\eta^2 + \eta\beta + \beta^2}, \frac{2}{\eta+\beta}\}$, and $\gamma = \frac{\beta}{2(1-c)\eta^2}$, we have

$$P_{t+1} \leq (1 - \delta)P_t,$$

where $0 < \delta < 1$. This bound implies that $\lim_{t \to \infty} P_t = 0$, and θ_t will converge linearly to the optimum θ^*.

Proof sketch. The proof is based on the proof framework detailed by Du and Hu (2018). The proof can be divided to three steps. First, we bound the decrease of $||\theta_t - \theta^*||$. Second, we bound the decrease of $||\tau_t - \tau_t^*||$. Third, we prove that $P_{t+1} \leq (1 - \delta)P_t$. The complete proof is in the long version of this paper (Lu, Liu, and Huang 2018). $\square$

Generalization to Templated or Conditional Models

So far, we have described the BBPL approach in the setting where there is a separate parameter for every entry in the marginal vector. In templated or conditional models, the parameters can be shared across multiple entries. We describe here how BBPL generalizes to such models. For conditional models, we use MRFs to infer the probability of output variables Y given input variables X, i.e., $\Pr(Y|X)$. A standard modeling technique for this task is to encode the joint states of the input and output variables with some feature vectors, so the parameters are weights for these features. We are given a dataset of fully observed input-output pairs $S = \{(M_i, y_i)\}_{i=1}^N$, where $M_i \in \mathbb{R}^{K \times d}$ is the feature matrix, and $y_i \in \mathbb{R}^d$ is the label vector (i.e., the one-hot encoding of the ground-truth variable states). The negative log-likelihood of a conditional random field is defined as

$$L(\tilde{\theta}) = -\frac{1}{N} \sum_{i=1}^N \log L_i(\tilde{\theta}, M_i, y_i), \tag{14}$$

where $\tilde{\theta} \in \mathbb{R}^K$ is the parameter we want to learn, and

$$L_i(\tilde{\theta}, M_i, y_i) = \tilde{\theta}^T M_i y_i - B(\tilde{\theta}^T M_i). \tag{15}$$

The definition of $B(\tilde{\theta}^T M_i)$ is similar to that for MRFs. We can interpret $\tilde{\theta}^\top M_i$ to be the expanded, or "grounded," potential vector. The product with feature matrix M_i maps from the low-dimensional $\tilde{\theta}$ to a possibly high-dimensional, full potential vector θ. To be precise, the definition is

$$B(\tilde{\theta}^T M_i) = \max_{\tau_i \in \mathbb{L}(G)} \{\tilde{\theta}^T M_i \tau_i - B^*(\tau_i)\}.$$

For each data point i, we can still use BBP to update $\tau_{i,t}$ at iteration t, i.e, lines 4–9 in Algorithm 2. Let $\bar{w} = \frac{1}{N} \sum_i M_i y_i$. The approximate gradient is then

$$g(\tilde{\theta}_t) = \bar{w} - \frac{1}{N} \sum_{i=1}^N M_i \tau_{i,t}. \tag{16}$$

Given $g(\tilde{\theta}_{t-1})$, we compute $g(\tilde{\theta}_t)$ with the following rule:

$$g(\tilde{\theta}_{t-1}) = g(\tilde{\theta}_{t-1}) + \frac{1}{N}\sum_{i=1}^{N} U_{i,F_t}(\tau_{i,t}^{(F_t)} - \tau_{i,t-1}^{(F_t)}).$$

Thus, the iteration complexity of inference only depends on the subnetwork size, rather than the size of the whole network.

Regarding convergence, note that each matrix M_i is constant, and its norm is bounded by its eigenvalues. Using the same proof method as that of Theorem 1, we can straightforwardly prove that it still has a linear convergence rate.

The same formulation—using a matrix M to map from a low-dimensional parameter vector $\tilde{\theta}$ to a possibly high-dimensional, full potential vector—can also be used to describe templated MRFs, where the same potential functions may be used in multiple parts of the graph. Therefore the same techniques and analysis also apply for templated MRFs.

Empirical Study

In this section, we empirically analyze the performance of BBPL. We design two groups of experiments. In the first group of experiments, we evaluate the sensitivity of our method on the block size and empirically measure its convergence on synthetic networks. In the second group of experiments, we test our methods on a popular application of MRFs: image segmentation on a real image dataset.

Baselines We compare BBPL to other methods that exactly optimize the full variational likelihood. We use full convex BP and inner-dual learning (Bach et al. 2015; Hazan, Schwing, and Urtasun 2016) as our baselines. Full BP learning runs inference to convergence each gradient step, and inner-dual learning accelerates learning by performing only one iteration of inference per learning iteration.

Metrics We evaluate the convergence and correctness of our method by measuring the objective value and the distance between the current parameter vector θ and the optimal θ^*. To compute the objective value, we store the parameters θ_t obtained by full BP learning, BBPL, or inner-dual learning during training. We then plug each into Equation 15 to compute the variational negative log-likelihood. To compute the distance between θ_t and θ^*, we first run convex BP learning until convergence to get θ^*, and then we use the ℓ_2 norm to measure the distance.

Experiments on Synthetic Networks

We generate two types of synthetic MRFs: grid networks and Barabási-Albert (BA) random graph networks (Albert and Barabási 2002). The nodes of the BA networks are ordered according to their generating sequence. We generate true unary and pairwise features from zero-mean, unit-variance Gaussian distributions. The unary feature dimension is 20 and the pairwise feature dimension is 10, and each variable has 8 possible states. Once the true models are generated, we draw 20 samples for each dataset with Gibbs sampling.
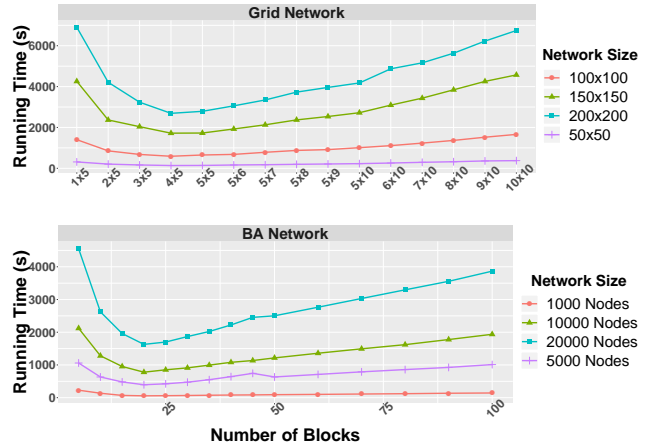

Figure 1: Sensitivity of BBPL to block size. The label 5×5 in the top plot represents that each grid is divided to 5 rows and 5 columns of sub-networks. The label 20 in the bottom plot represents that the network is divided into 20 networks. The results show that BBPL converges the fastest when each block's size is about 20 to 25 times smaller than the network.

Sensitivity to block size We conduct experiments on grid networks of four different sizes, i.e., 50×50, 100×100, 150×150, and 200×200, and on BA networks with four different sizes, i.e., 1,000 nodes, 5,000 nodes, 10,000 nodes, and 20,000 nodes. For the grid networks, we vary the number of blocks from 25 (5×5) to 100 (10×10). For example, when the network size is 200×200 and the number of blocks is 5×5, we separate the network into 25 sub-networks, and each sub-network's size is 40×40. For the BA networks, we vary the number of blocks from 5 to 100. We separate the nodes of each BA network based on their indices. For example, for a node set $V = \{1, 2, ..., 10\}$, if we want to separate it into two subsets, we will let $V_1 = 1, ..., 5$, and $V_2 = \{6, ..., 10\}$.

The results are plotted in Figure 1. The trends indicate that when each block is between 20 to 25 times smaller than the network, BBPL converges the fastest. When the block size is too small, the algorithm needs more iterations to converge. When the block size is too large, per-iteration complexity will be large. Blocks that are either too large or too small can reduce the benefits of BBPL.

Convergence analysis For the grid networks, we vary the network size from 10×10 to 200×200, and we set the number of blocks to 4×5. For the BA networks, we vary the network size from 100 to 20,000, and we set the number of blocks to 20. Figure 2 and Figure 3 empirically show the convergence of BBPL on different networks. Figure 4 plots the running time comparison on networks with all sizes. From Figure 2 we can see that BBPL converges to the same solution as full BP learning, while requiring significantly less time. Figure 4 shows that on all networks, BBPL is the fastest. The acceleration is more significant when the network is large. For example, when the grid network's size is 200×200, BBPL is about three times faster than the inner-
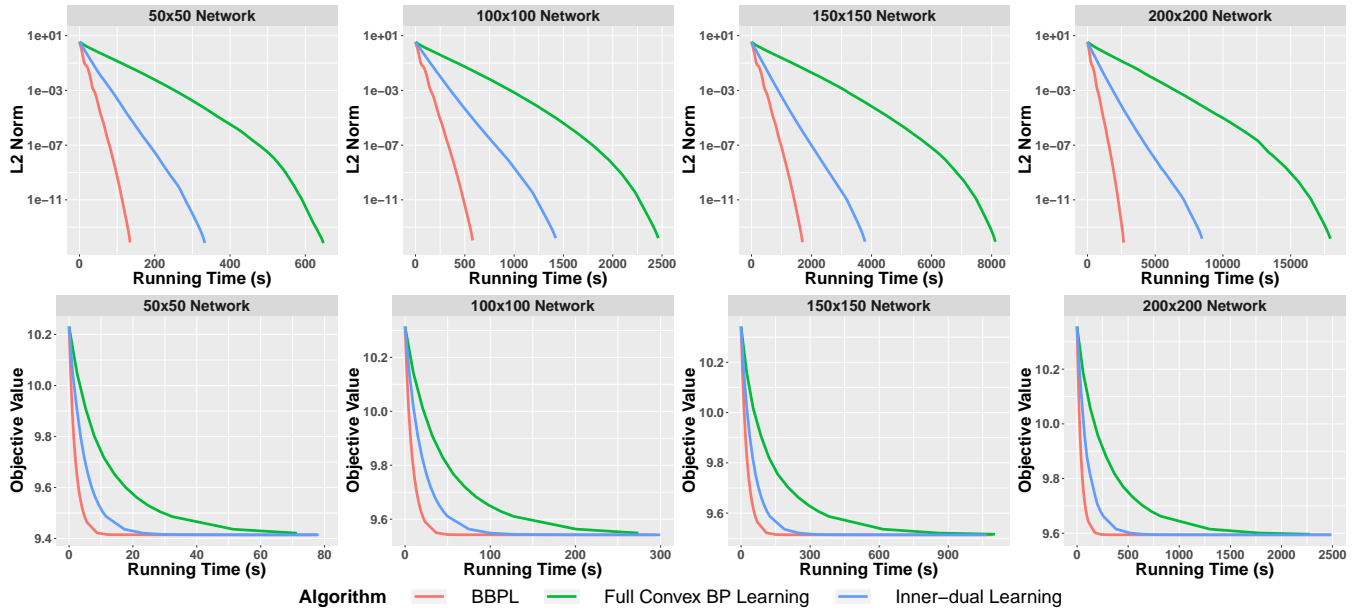
Figure 2: Convergence on grid networks. The top row plots the ℓ_2 distance from the optimum, and the second row plots the objective values. The ℓ_2 plots show the full running time of each method, but for clarity, we zoom into the plots of objective values by truncating the x-axis.
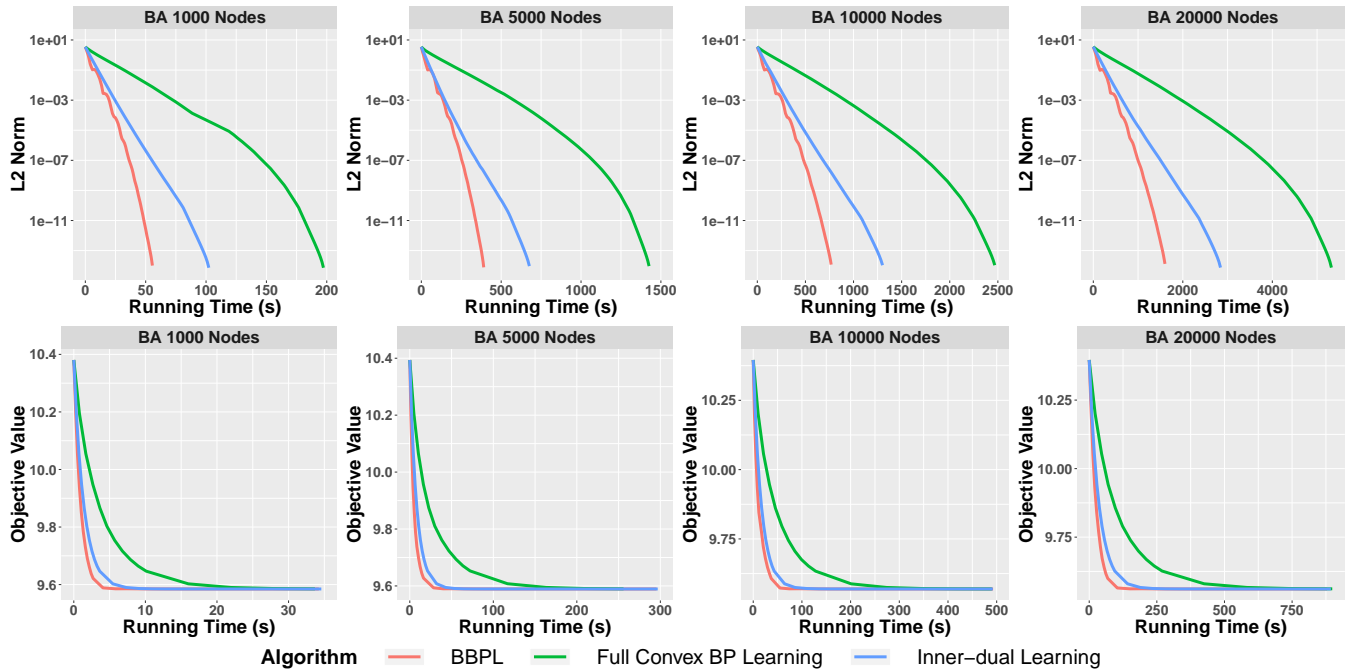

Figure 3: Results on BA networks. The top row again plots the ℓ_2 distance from the optimum, and the bottom row plots the objective zoomed to show behavior early during optimization.

dual method and seven times faster than full convex BP. When the BA network size is 20,000, BBPL is about two times faster than inner-dual learning and three times faster than full convex BP.
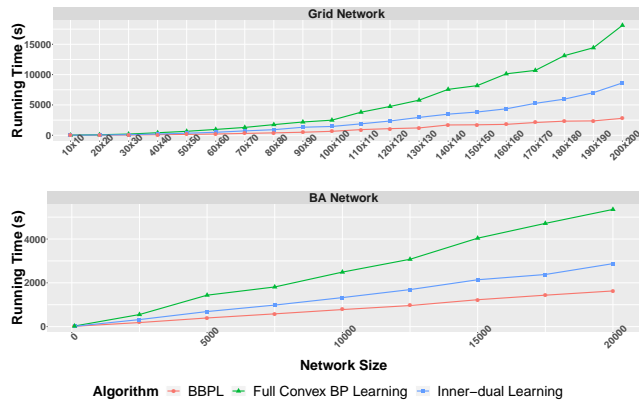

Figure 4: Comparisons of running time on networks with different sizes. BBPL converges much faster than the other two methods, especially on large networks where the improvement in running time is more significant.

Experiments on Image Dataset

For our real data experiments, we use the *scene understanding* dataset (Gould, Fulton, and Koller 2009) for semantic image segmentation. Each image is 240×320 pixels in size. We randomly choose 50 images as the training set and 20 images as the test set. We extract unary features from a fully convolutional network (FCN) (Long, Shelhamer, and Darrell 2015). We add a linear transpose layer between the output layer and the last deconvolution layer of the FCN. This transpose layer does not impact the FCN's performance. Let $x \in \mathbb{R}^n$ be the output of the deconvolution layer and $y \in \mathbb{R}^c$ be the number of classes of the dataset. Then the input of the transpose layer is x and its output is y. We use x as the MRF's unary features. We use FCN-32 models to generate the features and we fine-tuned its parameters from the pre-trained VGG 16-layer network (Simonyan and Zisserman 2014). Our pairwise features are based on those of Domke (2013): for edge features of vertex s and t, we compute the ℓ_2 norm of unary features between s and t and discretize it to 10 bins.

We train MRFs on this segmentation task. Since these MRFs are large, full BP learning cannot run in a reasonable amount of time. Instead, we first run BBPL until convergence, and then we run inner-dual learning and full BP learning for the same amount of time. Finally, we compare their objective values during optimization. This evaluation scheme was necessary because the baseline methods need several days to converge on these large networks. The results are plotted in Figure 5, and Table 1 shows the number of iterations each algorithm runs when it stops. BBPL's per-iteration complexity is much lower than the other methods, so it runs the most number of iterations in the same time. Following the trends seen in the synthetic experiments,

BBPL again reduces the objective much faster than the two other methods. Sample segmentation results are in the long version of this paper (Lu, Liu, and Huang 2018).

Table 1: Number of iterations each algorithm runs.

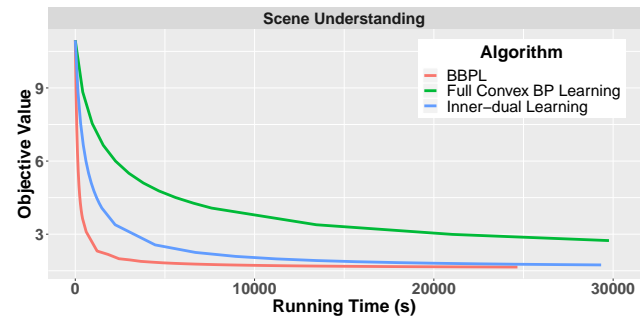| BBPL	Inner-dual Learning	Full BP Learning
601	163	21

Figure 5: The learning objective during training on the scene understanding dataset. We run the three methods for the same amount of time. Our BBPL method is much faster than the two other methods.

Conclusion

In this paper, we developed block belief propagation learning (BBPL), for training Markov random fields. At each learning iteration, our method only performs inference on a subnetwork, and it uses an approximation of the true gradient to optimize the parameters of interest. Thus, BBPL's iteration complexity does not scale with the size of the network. We theoretically prove that BBPL has a linear convergence rate and that it converges to the same optimum as convex BP. Our experiments show that, since BBPL has much lower iteration complexity, it converges faster than other methods that run (truncated or complete) inference on the full MRF each learning iteration.

For future work, we plan to use the scalability of BBPL to analyze large-scale networks. Further speedups may be possible. Even though BBPL only needs to run inference on a subnetwork, it still needs to run many iterations of belief propagation until convergence. We plan to develop a more efficient learning method that can stop inference in a fixed number of iterations, combining the benefits of BBPL and inner-dual learning. Finally, our proof depends on Assumption 1, which has been made in the literature about belief propagation-like algorithms but has not been proven. We aim to both prove its validity for existing methods and use it to help derive new inference methods suitable for learning.

Acknowledgments

References

Ahmadi, B.; Kersting, K.; and Natarajan, S. 2012. Lifted online training of relational models with stochastic gradient methods. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*.

Albert, R., and Barabási, A.-L. 2002. Statistical mechanics of complex networks. *Reviews of Modern Physics* 74:47.

Bach, S.; Huang, B.; Boyd-Graber, J.; and Getoor, L. 2015. Paired-dual learning for fast training of latent variable hinge-loss MRFs. In *International Conference on Machine Learning*.

Bixler, R., and Huang, B. 2018. Sparse matrix belief propagation. In *Conference on Uncertainty in Artificial Intelligence*.

Domke, J. 2011. Parameter learning with truncated message-passing. In *Computer Vision and Pattern Recognition*.

Domke, J. 2013. Learning graphical model parameters with approximate marginal inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(10):2454–2467.

Du, S. S., and Hu, W. 2018. Linear convergence of the primal-dual gradient method for convex-concave saddle point problems without strong convexity. *arXiv preprint arXiv:1802.01504*.

Globerson, A., and Jaakkola, T. 2007. Approximate inference using conditional entropy decompositions. In *Artificial Intelligence and Statistics*.

Gould, S.; Fulton, R.; and Koller, D. 2009. Decomposing a scene into geometric and semantically consistent regions. In *International Conference on Computer Vision*.

Hazan, T., and Urtasun, R. 2010. A primal-dual message-passing algorithm for approximated large scale structured prediction. In *Advances in Neural Information Processing Systems*.

Hazan, T.; Schwing, A. G.; and Urtasun, R. 2016. Blending learning and inference in conditional random fields. *The Journal of Machine Learning Research* 17:8305–8329.

Heskes, T. 2006. Convexity arguments for efficient minimization of the Bethe and Kikuchi free energies. *Journal of Artificial Intelligence Research* 26:153–190.

Kersting, K.; Ahmadi, B.; and Natarajan, S. 2009. Counting belief propagation. In *Uncertainty in Artificial Intelligence*.

Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT press.

Lin, G.; Shen, C.; Reid, I.; and van den Hengel, A. 2015. Deeply learning the messages in message passing inference. In *Advances in Neural Information Processing Systems*.

London, B.; Huang, B.; and Getoor, L. 2015. The benefits of learning with strongly convex approximate inference. In *International Conference on Machine Learning*.

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *Computer Vision and Pattern Recognition*, 3431–3440.

Lu, Y.; Liu, Z.; and Huang, B. 2018. Block belief propagation for parameter learning in markov random fields. *arXiv preprint arXiv:1811.04064*.

Meshi, O.; Jaimovich, A.; Globerson, A.; and Friedman, N. 2009. Convexifying the Bethe free energy. In *Conference on Uncertainty in Artificial Intelligence*.

Meshi, O.; Sontag, D.; Jaakkola, T.; and Globerson, A. 2010. Learning efficiently with approximate inference via dual losses. In *International Conference on Machine Learning*.

Noorshams, N., and Wainwright, M. J. 2013. Stochastic belief propagation: A low-complexity alternative to the sum-product algorithm. *IEEE Transactions on Information Theory* 59:1981–2000.

Nowozin, S., and Lampert, C. H. 2011. Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision* 6:185–365.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62:107–136.

Roosta, T. G.; Wainwright, M. J.; and Sastry, S. S. 2008. Convergence analysis of reweighted sum-product algorithms. *IEEE Transactions on Signal Processing* 56:4293–4305.

Ross, S.; Munoz, D.; Hebert, M.; and Bagnell, J. A. 2011. Learning message-passing inference machines for structured prediction. In *Computer Vision and Pattern Recognition*.

Samdani, R., and Roth, D. 2012. Efficient decomposed learning for structured prediction. *arXiv preprint arXiv:1206.4630*.

Schwing, A.; Hazan, T.; Pollefeys, M.; and Urtasun, R. 2011. Distributed message passing for large scale graphical models. In *Computer Vision and Pattern Recognition*.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Singla, P., and Domingos, P. M. 2008. Lifted first-order belief propagation. In *Association for the Advancement of Artificial Intelligence*.

Stoyanov, V.; Ropson, A.; and Eisner, J. 2011. Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In *International Conference on Artificial Intelligence and Statistics*, 725–733.

Sutton, C., and McCallum, A. 2009. Piecewise training for structured prediction. *Machine Learning* 77(2–3):165–194.

Taskar, B.; Chatalbashev, V.; Koller, D.; and Guestrin, C. 2005. Learning structured prediction models: A large margin approach. In *International Conference on Machine Learning*.

Taskar, B.; Guestrin, C.; and Koller, D. 2004. Max-margin markov networks. In *Advances in Neural Information Processing Systems*.

Wainwright, M. J., and Jordan, M. I. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning* 1:1–305.

Wainwright, M. J.; Jaakkola, T. S.; and Willsky, A. S. 2005. A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory* 51:2313–2335.

Wainwright, M. J. 2006. Estimating the "wrong" graphical model: Benefits in the computation-limited setting. *Journal of Machine Learning Research* 7:1829–1859.

Yedidia, J. S.; Freeman, W. T.; and Weiss, Y. 2005. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory* 51:2282–2312.

Yin, J., and Gao, L. 2014. Scalable distributed belief propagation with prioritized block updates. In *International Conference on Conference on Information and Knowledge Management*.