Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

The Linearization of Belief Propagation on Pairwise Markov Random Fields

Wolfgang Gatterbauer
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

Belief Propagation (BP) is a widely used approximation for exact probabilistic inference in graphical models, such as Markov Random Fields (MRFs). In graphs with cycles, however, no exact convergence guarantees for BP are known, in general. For the case when all edges in the MRF carry the same symmetric, doubly stochastic potential, recent works have proposed to approximate BP by linearizing the update equations around default values, which was shown to work well for the problem of node classification. The present paper generalizes all prior work and derives an approach that approximates loopy BP on any pairwise MRF with the problem of solving a linear equation system. This approach combines exact convergence guarantees and a fast matrix implementation with the ability to *model heterogenous networks*. Experiments on synthetic graphs with planted edge potentials show that the linearization has comparable labeling accuracy as BP for graphs with weak potentials, while speeding-up inference by orders of magnitude.

1 Introduction

Belief Propagation (BP) is an iterative message-passing algorithm for performing inference in graphical models (GMs), such as Markov Random Fields (MRFs). BP calculates the marginal distribution for each unobserved node, conditional on any observed nodes (Pearl 1988). It achieves this by propagating the information from a few observed nodes throughout the network by iteratively passing information between neighboring nodes. It is known that when the graphical model has a tree structure, then BP converges to the true marginals (according to exact probabilistic inference) after a finite number of iterations. In loopy graphs, convergence to the correct marginals is not guaranteed; in fact, it is not guaranteed at all, and using BP can lead to well-documented convergence problems (Sen et al. 2008). While there is a lot of research on convergence of BP (Elidan, McGraw, and Koller 2006; Ihler, Fisher III, and Willsky 2005; Mooij and Kappen 2007), exact criteria for convergence are not known (Murphy 2012), and most existing bounds for BP on general pairwise MRFs give only sufficient convergence criteria, or are for restricted cases, such as when the underlying distributions are Gaussians (Malioutov, Johnson, and Willsky 2006; Su and Wu 2015; Weiss and Freeman 2001).

Semi-supervised node classification. BP is also a versatile formalism for semi-supervised learning; i.e., assigning classes to unlabeled nodes while maximizing the number of correctly labeled nodes (Koller and Friedman 2009, ch. 4). The goal is to predict the most probable class for each node in a network *independently*, which corresponds to the Maximum Marginal (MM) assignment (Domke 2013; Weiss 2000). Let $\mathbb{P}$ be a probability distribution over a set of random variables $\mathbf{X} \cup \mathbf{Y}$. MM-inference (or "MM decoding") searches for the most probable assignment y_i for each unlabeled node Y_i *independently*, given evidence $\mathbf{X} = \mathbf{x}$:

$$\text{MM}(\mathbf{y}|\mathbf{x}) = \{\arg\max_{y_i} \mathbb{P}(Y_i = y_i | \mathbf{X} = \mathbf{x}) | Y_i \in \mathbf{Y}\}$$

Notice that this problem is simpler than finding the actual marginal distribution. It is also *different* from finding the Maximum A-Posteriori (MAP) assignment (the "most probable configuration"), which is the mode or the most probable *joint classification* of all non-evidence variables:[1]

$$\text{MAP}(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{y}} \mathbb{P}(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x})$$

Convergent message-passing algorithms. There has been much research on finding variations to the update equations of BP that guarantee convergence. These algorithms are often similar in structure to the non-convergent algorithms, yet it can be proven that the value of the variational problem (or its dual) improves at each iteration (Hazan and Shashua 2008; Heskes 2006; Meltzer, Globerson, and Weiss 2009). Another body of recent papers have suggested to solve the convergence problems of MM-inference by linearizing the update equations. Krzakala et al. study a form of linearization for unsupervised classification called "spectral redemption" in the stochastic block model. That model is unsupervised and has no obvious way to include supervision in its setup (i.e., it is not clear how to leverage labeled nodes). Donoho, Maleki, and Montanari propose "approximate message-passing" (AMP) as an iterative thresh-

[1] See (Murphy 2012, ch. 5.2.1) for a detailed discussion on why MAP has some undesirable properties and is not necessarily a "representative" assignment. While in theory it is arguably preferable to compute marginal probabilities, in practice researchers often use MAP inference due to the availability of efficient discrete optimization algorithms (Korč, Kolmogorov, and Lampert 2012).

olding algorithm for compressed sensing that is largely inspired by BP. Koutra et al. linearize BP for the case of two classes and proposed "Fast Belief Propagation" (FaBP) as a method to propagate existing knowledge of homophily or heterophily to unlabeled data. This framework allows one to specify a homophily factor h ($h > 0$ for homophily or $h < 0$ for heterophily) and to then use this algorithm with exact convergence criteria for binary classification. Gatterbauer et al. derive a multivariate ("polytomous") generalization of FaBP from binary to multiple labels called "Linearized Belief Propagation" (LinBP). Both aforementioned papers show considerable speed-ups for the application of node classification and relational learning by transforming the update equations of BP into an efficient matrix formulation. However, those papers solve only special cases: FaBP is restricted to two classes per node (de facto, one single score). LinBP can handle multiple classes, but is restricted to one single node type, one single edge type, and a potential that is symmetric and doubly stochastic (see Fig. 1).[2]

Contributions. This paper derives a linearization of BP for *arbitrary pairwise MRFs*, which transforms the parameters of an MRF into an equation system that replaces multiplication with addition. In contrast to standard BP, the derived update equations (i) come with exact convergence guarantees, (ii) allow a closed-form solution, (iii) keep the derived beliefs normalized at each step, and (iv) can thus be put into an efficient linear algebra framework. We also show empirically that this approach – in addition to its compelling computational advantages – performs comparably to Loopy BP for a large part of the parameter space. In contrast to prior work on linearizing BP, we remove any restriction on the potentials and solve the most general case for pairwise MRFs (see Fig. 1). Since it is known that any *higher-order MRF* can be converted to a pairwise MRF (Wainwright and Jordan 2008, Appendix E.3), the approach can be also be used for higher-order potentials. Our formalism can thus model *arbitrary heterogeneous networks*; i.e., such that have directed edges or have different types of nodes.[3] This generalization is not obvious and required us to solve several new algebraic problems: (i) Non-symmetric potentials modulate messages differently across both directions of an edge; each direction then requires different centering points (this is particularly pronounced for non-quadratic potentials; i.e., when nodes adjacent to an edge have different numbers of classes). (ii) Multiplying belief vectors with non-stochastic potentials doesn't leave them stochastic; an additional normalization would then not allow a closed-form matrix formulation as before; we instead derive a "bias term" that remains constant in the update equations and thus depends only on the network structure and the potentials but not the beliefs. (iii) Dealing with the full heterogenous case (multiple potentials, multiple node types, and different numbers of classes among nodes) requires a considerably more general formulation. The technical report on arXiv (Gatterbauer 2015) contains the full derivation of the results presented in this paper. An efficient Python implementation is available on Github (SSLH 2015).

	BP	**FaBP**	**LinBP**	*this work*
# node types	arbitrary	1	1	arbitrary
# node classes	arbitrary	2	const k	arbitrary
# edge types	arbitrary	1	1	arbitrary
edge symmetry	arbitrary	required	required	arbitrary
edge potential	arbitrary	doubly stoch.	doubly stoch.	arbitrary
closed form	no	yes	yes	yes

Figure 1: The approach proposed in this paper combines the full expressiveness and generality of Loopy Belief Propagation (BP) on pairwise MRFs with the computational advantages of Fast BP (Koutra et al. 2011) and Linearized BP (Gatterbauer et al. 2015).

2 BP for pairwise MRFs

A MRF is a factored representation of a joint distribution over variables $\mathbf{X}$. The distribution is defined using a set of factors $\{\phi_f \mid f \in F\}$, where each f is associated with the variables $\mathbf{X}_f \subset \mathbf{X}$, and ϕ_f is a function from the set of possible assignments of $\mathbf{X}_f$ to $\mathbb{R}^+$. The joint distribution is defined as: $\mathbb{P}(\mathbf{X} = \mathbf{x}) = \frac{1}{Z}\prod_{f\in F}\phi_f(\mathbf{x}_f)$ where Z is a normalization constant known as the partition function.

An important subclass of MRFs is that of *pairwise MRFs*, representing distributions where all of the interactions between variables are limited to those between pairs. More precisely, a pairwise MRF over a graph is associated with a set of node potentials and a set of edge potentials (Koller and Friedman 2009). The overall distribution is the normalized product of all of the node and edge potentials.

We next focus on the mechanics of BP. Consider a network of n nodes where each node s can be any of k_s possible classes (or values). A node s maintains a k_s-dimensional belief vector where each element j represents a weight proportional to the belief that this node belongs to class j. Let $\boldsymbol{x}_s$ be the vector of prior beliefs (also varyingly called local evidence or node potential) and $\mathbf{y}_s$ the vector of posterior (or implicit or final) beliefs at node s, and require that $\boldsymbol{x}_s$ and $\mathbf{y}_s$ are normalized to 1; i.e., $\sum_{j\in[k_s]} x_s(j) = \sum_{j\in[k_s]} y_s(j) = 1$. For example, a labeled node s of class i is represented by $x_s(j) = 1$ for $j = i$ and $x_s(j) = 0$ for $j \neq i$. Using $\mathbf{m}_{us}$ for the k_s-dimensional message that node u sends to node s, we can write the BP update equations (Murphy 2012; Weiss 2000) for the belief vector of a node s as:

$$y_s(j) \leftarrow \frac{1}{Z_s} x_s(j) \prod_{u\in N(s)} m_{us}(j) \qquad (1)$$

Here, we write Z_s for a normalizer that makes the elements of $\mathbf{y}_s$ sum up to 1. Thus, the posterior belief $y_s(j)$ is com-

[2] A potential is "doubly stochastic" if all rows and columns sum up to 1. As potentials can be scaled without changing the semantics of BP, this definition also extends to any potential where the rows and columns sum to the same value.

[3] Notice that an *underlying directed network* is still modeled as an *undirected Graphical Model* (GM). For example, while the "friendship" relation on Facebook is undirected, the "follower" relation on Twitter is directed and has different implications on the two nodes adjacent to a directed "links to"-edge. Yet, the resulting GM is still undirected, but now has *asymmetric potentials*.

puted by multiplying the prior belief $x_s(j)$ with the incoming messages $m_{us}(j)$ from all neighbors $u \in N(s)$, and then normalizing so that the beliefs in all k_s classes sum to 1. In parallel, each node sends messages to each of its neighbors:

$$m_{st}(i) \leftarrow \frac{1}{Z_{st}} \sum_j \psi_{st}(j,i)\; x_s(j) \prod_{u \in N(s) \setminus t} m_{us}(j) \qquad (2)$$

Here, $\psi_{st}(j,i)$ is a proportional "coupling weight" (or "compatibility," "affinity," "modulation") that indicates the relative influence of class j of node s on class i of node t. Thus, the message $m_{st}(i)$ is computed by multiplying together all incoming messages at node s – except the one sent by the recipient t – and then passing through the $\boldsymbol{\psi}_{st}$ edge potential. Notice that we use Z_{st} in Eq. (2) as a normalizer that makes the elements of $\mathbf{m}_{st}$ sum up to k_t at each iteration. As pointed out by Murphy, Weiss, and Jordan; Pearl, normalizing the messages has no effect on the final beliefs; however, this intermediate normalization of messages will become crucial in our derivations. BP then repeatedly computes the above update equations for each node until the values (hopefully) converge. At iteration r of the algorithm, $y_s(j)$ represents the posterior belief of j conditioned on the evidence that is r steps away in the network.

3 Linearizing BP over any pairwise MRF

This section gives a closed form description for the final beliefs after convergence of BP in *arbitrary pairwise MRFs* under a certain limit consideration of all parameters. This is a strict and non-trivial generalization of recent works (Fig. 1). The difficulty of our generalization lies in technical details: non-symmetric potentials require different centering points for messages across different directions of an edge; non-stochastic potentials require different normalizers for different iterations (and for different potentials in the networks) which does not easily lead to a simple matrix formulation; and *the full heterogenous case* (e.g., different number of classes k for different nodes) requires a considerably more general derivation and final formulation.

Our approach is conceptually simple: we center all matrix entries around *well-chosen* default values and then focus only on the deviations from these defaults using Maclaurin series at several steps in our derivation. The resulting equations replace multiplication with addition and can thus be put into the framework of matrix-vector multiplication, which can leverage existing highly-optimized code. It also allows us to give exact convergence criteria for the resulting update equations and a closed form solution (that would require the inversion of a large matrix). The approach is similar in spirit to the idea of writing any MRF (with strictly positive density) as log-linear model. However, by starting from the update equations for loopy BP, we solve the intractability problem by ignoring all dependencies between messages that have traveled over a path of length 2 or more.

Definition 1 (Centering)**.** *We call a vector* $\mathbf{x}$ *or matrix* $\mathbf{X}$ *"centered around* c *with standard deviation* v*" if the average entry* $\mu(\mathbf{x}) = c$ *and standard deviation* $\sigma(\mathbf{x}) = v$*.*

Definition 2 (Residual vector/matrix)**.** *If a vector* $\mathbf{x}$ *is centered around* c*, then the "residual vector"* $\hat{\mathbf{x}}$ *around* c *is defined as* $\hat{\mathbf{x}} = [x_1 - c, x_2 - c, \ldots]^\intercal$*. Accordingly, we denote a matrix* $\hat{\mathbf{X}}$ *as a "residual matrix" if each entry is the residual after centering around* c*.*

For example, the vector $\mathbf{x} = [1.1, 1.2, 0.7]^\intercal$ is centered around $c = 1$, and the residuals from 1 form the residual vector $\hat{\mathbf{x}} = [0.1, 0.2, -0.3]^\intercal$; i.e., $\mathbf{x} = \mathbf{1}_3 + \hat{\mathbf{x}}$, where $\mathbf{1}_3$ is the 3-dimensional vector with all entries equal to 1. By definition of a normalized vector, beliefs for any node s are centered around $\frac{1}{k_s}$, and the residuals for prior beliefs have non-zero elements (i.e., $\hat{\boldsymbol{x}}_s \neq \mathbf{0}_{k_s}$) only for nodes with local evidence (nodes "with explicit beliefs"). Further notice that the entries in a residual vector or matrix always sum up to 0 (i.e., $\sum_i \hat{x}(i) = 0$). This is done by construction and will become important in the derivations of our results.

The main idea of our derivation relies then on the following observation: if we start with messages and potentials with rows and columns centered around 1 with small enough standard deviations, then the normalizer of the update equation Eq. (2) is independent of the beliefs and *remains constant* as $Z_{st} = k_t^{-1}$. Importantly, the resulting equations *do not require further normalization*. The derivation further makes use of certain linearizing approximations that result in a well-behaved linear equation system. We show that the MM solutions implied by this equation system are *identical* to those from the original BP update equations in case of nearly uniform priors and potentials. For strong priors and potentials (e.g., $\left[\begin{smallmatrix} 1 & 100 \\ 100 & 1 \end{smallmatrix}\right]$), the resulting solutions are not identical anymore, yet serve as reasonable approximations in a wide range of problem parameters (see Section 4). WLOG, we start with potentials that are centered around 1 and then re-center the potentials before using them:[4]

Definition 3 (Row-recentered residual matrix)**.** *Let* $\boldsymbol{\psi} \in \mathbb{R}^{\ell \times k}$ *be centered around 1 and* $\hat{\boldsymbol{\psi}}$ *be the residual matrix around 1. Furthermore, let* $\hat{r}(j) \coloneqq \sum_i \hat{\psi}(j,i)$ *be the sum of the residuals of row* j*. Then the "row-recentered residual matrix"* $\hat{\boldsymbol{\psi}}'$ *has entries* $\hat{\psi}'(j,i) \coloneqq \frac{1}{k}\big(\hat{\psi}(j,i) - \frac{\hat{r}(j)}{k}\big)$*.*

Before we can state our main result, we need some additional notation. WLOG, let $[n]$ be the set of all nodes. For each node $s \in [n]$, let k_s be the number of its possible classes. Let $\mathbf{k}_s \coloneqq \frac{1}{k_s}\mathbf{1}_{k_s}$, i.e., the k_s-dimensional uniform stochastic column vector. Furthermore, let $k_{\text{tot}} \coloneqq \sum_{s \in [n]} k_s$ be the sum of classes across nodes. To write all our resulting equations as one large equation system, we stack the individual explicit ($\hat{\boldsymbol{x}}$) and implicit ($\hat{\mathbf{y}}$) residual belief vectors together with the $\mathbf{k}_s$-vectors one underneath the other to form three k_{tot}-dimensional stacked column vectors. We also combine all row-recentered residual matrices into one large but sparse $[k_{\text{tot}} \times k_{\text{tot}}]$-square block matrix

[4] Without changing the joint probability distribution, every potential in a MRF can be scaled so that the average entry is 1. For example, given $\boldsymbol{\psi} = \left[\begin{smallmatrix} 4 & 6 & 5 \\ 6 & 8 & 7 \end{smallmatrix}\right]$, we scale by $\frac{1}{6}$ to get $\boldsymbol{\psi} = \left[\begin{smallmatrix} \frac{2}{3} & 1 & \frac{5}{6} \\ 1 & \frac{4}{3} & \frac{7}{6} \end{smallmatrix}\right]$, which has the identical semantics but is now centered around 1.

(notice that all entries for non-existing edges remain empty):

$$\hat{\mathbf{y}} \coloneqq \begin{bmatrix} \hat{\mathbf{y}}_1 \\ \vdots \\ \hat{\mathbf{y}}_n \end{bmatrix},\ \hat{\boldsymbol{x}} \coloneqq \begin{bmatrix} \hat{\boldsymbol{x}}_1 \\ \vdots \\ \hat{\boldsymbol{x}}_n \end{bmatrix},\ \mathbf{k} \coloneqq \begin{bmatrix} \mathbf{k}_1 \\ \vdots \\ \mathbf{k}_n \end{bmatrix},\ \hat{\boldsymbol{\psi}}' \coloneqq \begin{bmatrix} \hat{\boldsymbol{\psi}}'_{11} & \dots & \hat{\boldsymbol{\psi}}'_{1n} \\ \vdots & \ddots & \vdots \\ \hat{\boldsymbol{\psi}}'_{n1} & \dots & \hat{\boldsymbol{\psi}}'_{nn} \end{bmatrix}$$

We can now state our main theorem:

Theorem 4 (Linearizing Belief Propagation)**.** *Let $\hat{\mathbf{y}}$, $\hat{\boldsymbol{x}}$, $\hat{\mathbf{k}}$, and $\hat{\boldsymbol{\psi}}'$ be the above defined residual vectors and matrix. Let ϵ be a bound on the standard deviation of all non-zero entries of $\hat{\boldsymbol{\psi}}'$ and $\hat{\boldsymbol{x}}$, $\sigma(\hat{\boldsymbol{\psi}}') < \epsilon$ and $\sigma(\hat{\boldsymbol{x}}) < \epsilon$. Let $\mathbf{y}_v^{BP}$ be the final belief assignment for any node v after convergence of BP. Then, for $\lim_{\epsilon \to 0^+}$, $\arg\max_i y_v^{BP}(i) = \arg\max_i \hat{y}_v^{Lin}(i)$, where $\hat{\mathbf{y}}_v$ results from solving the following system of k_{tot} linear equations in $\hat{\mathbf{y}}$:*

$$\hat{\mathbf{y}} = \underbrace{\hat{\boldsymbol{x}}}_{1^{st}} + \underbrace{\hat{\boldsymbol{\psi}}'^{\intercal}\mathbf{k}}_{2^{nd}} + \underbrace{\hat{\boldsymbol{\psi}}'^{\intercal}\hat{\mathbf{y}}}_{3^{rd}} - \underbrace{\hat{\boldsymbol{\psi}}'^{\intercal 2}\hat{\mathbf{y}}}_{4^{th}} \tag{3}$$

In other words, the MM node labeling from BP can be approximated by solving a linear equation system if each of the potentials and each of the beliefs are reasonably tightly centered around their average values. Notice that the 2nd term $\hat{\boldsymbol{\psi}}'^{\intercal}\mathbf{k}$ is a "bias" vector that depends only on the structure of the network and the potentials, but *not* the beliefs. We thus sometimes prefer to write $\hat{\mathbf{c}}'_* \coloneqq \hat{\boldsymbol{\psi}}'^{\intercal}\mathbf{k}$ to emphasize that it remains constant during the iterations. This term vanishes if all potentials are doubly stochastic. Also notice that the 4th term is what was called the "echo cancellation" in (Gatterbauer et al. 2015).[5] Simple algebraic manipulations then lead a closed-form solution by solving Eq. (3) for $\hat{\mathbf{y}}$:

$$\hat{\mathbf{y}} = \big(\mathbf{I}_{k_{\text{tot}}} - \hat{\boldsymbol{\psi}}'^{\intercal} + \hat{\boldsymbol{\psi}}'^{\intercal 2}\big)^{-1}\big(\hat{\boldsymbol{x}} + \hat{\mathbf{c}}'_*\big) \tag{4}$$

Iterative updates and convergence

The complexity of inverting a matrix is cubic in the number of variables, which makes direct application of Eq. (4) difficult. Instead, we use Eq. (3), which gives an implicit definition of the final beliefs, iteratively. Starting with an arbitrary initialization of $\hat{\mathbf{y}}$ (e.g., all values zero), we repeatedly compute the right hand side of the equations and update the values of $\hat{\mathbf{y}}$ until the process converges:[6]

Proposition 5 (Update equations)**.** *The positive fix points for Eq.* (3) *can be calculated iteratively with the following update equations starting from $\hat{\mathbf{y}}^{(0)} = \mathbf{0}$:*

$$\hat{\mathbf{y}}^{(r+1)} \leftarrow \big(\hat{\boldsymbol{x}} + \hat{\mathbf{c}}'_*\big) + \big(\hat{\boldsymbol{\psi}}'^{\intercal} - \hat{\boldsymbol{\psi}}'^{\intercal 2}\big)\hat{\mathbf{y}}^{(r)} \tag{5}$$

These particular update equations allow us to give a sufficient and necessary criterium for convergence via the spectral radius ρ of a matrix.[7]

Corollary 6 (Convergence)**.** *The update Eq.* (5) *converges if and only if $\rho\big(\hat{\boldsymbol{\psi}}' - \hat{\boldsymbol{\psi}}'^2\big) < 1$.*

Thus, the updates converge towards the closed-form solution, and the final beliefs of each node can be computed via efficient matrix operations with optimized packages, while the implicit form gives us guarantees for the convergence of this process.[8] In order to apply our approach to problem settings with spectral radius bigger than one (and thus direct application of Eq. (5) would not work), we propose to modify the model by *weakening the potentials*. In other words, we multiply $\hat{\boldsymbol{\psi}}'$ with a factor that guarantees convergence. We call the multiplicative factor which exactly separates convergence from divergence, the "*convergence boundary*" ϵ_*. Choosing any ϵ with $s \coloneqq \frac{\epsilon}{\epsilon_*}$ and $s < 1$ guarantees convergence. We call any choice of s the "*convergence parameter*."

Definition 7 (Convergence boundary ϵ_*)**.** *For any $\hat{\boldsymbol{\psi}}'$, the convergence boundary $\epsilon_* > 0$ is defined implicitly by $\rho\big(\epsilon_*\hat{\boldsymbol{\psi}}' - \epsilon_*^2\hat{\boldsymbol{\psi}}'^2\big) = 1$.*

Computational complexity

Naively materializing $\hat{\boldsymbol{\psi}}'$ would lead to a space requirement of $\mathcal{O}(n^2 k_{\max}^2)$ where n is the number of nodes and $k_{\max}$ the max number of classes per node. However, by using a sparse matrix implementation, both the space requirement and the computational complexity of each iteration are only proportional to the number of edges: $\mathcal{O}(mk_{\max}^2)$. The time complexity is identical to the one of message-passing *with division*, which avoids redundant calculations and is faster than standard BP on graphs with high node degrees (Koller and Friedman 2009). However, the ability to use existing highly-optimized packages for efficient matrix-vector multiplication will considerably speed-up the actual calculations.

4 Experiments

Questions. Our experiments will answer the following 3 questions: (1) What is the effect of the convergence parameter s on accuracy and number of required iterations until convergence? (2) How accurate is our approximation under

[5] Notice that the BP update equations send a message across an edge that excludes information received across the same edge from the other direction: "$u \in N(s) \setminus t$" in Eq. (2). In a probabilistic scenario on tree-based graphs, this echo cancellation is required for correctness. In loopy graphs (without well-justified semantics), this term still compensates for the message a node t would otherwise send to itself via a neighbor s, i.e., via the path $t \to s \to t$.

[6] Interestingly, our linearized update equations, Eq. (5), are reminiscent of the update equations for the mean beliefs in Gaussian MRFs (Malioutov, Johnson, and Willsky 2006; Su and Wu 2015; Weiss and Freeman 2001). Notice however, that whereas the update equations are exact in the case of continuous Gaussian MRFs, our equations are approximations for the general discrete case.

[7] The "spectral radius" $\rho(\cdot)$ of a matrix is the supremum among the absolute values of its eigenvalues.

[8] The intuition behind these equivalences can be illustrated by comparing to the geometric series $S = 1 + x + x^2 + \dots$ and its closed form $S = (1-x)^{-1}$. Whereas for $|x| < 1$, the series converges to its closed-form, for $|x| > 1$, it diverges, and the closed-form is meaningless.

varying conditions: (i) the density of the network, (ii) the strength on the interaction, and (iii) the fraction of labeled nodes? (3) How fast is the linearized approximation as compared to standard Loopy BP?

Experimental protocol. We define "accuracy" as the fraction of unlabeled nodes that receive correct labels. In order to evaluate the accuracy of a method, we need to use graphs with known label ground truth (GT). As we are interested in the accuracy as a function of various parameters, we need graphs with *controlled GT*. We thus decided to compare BP against its linearization on synthetic graphs with known GT, which allows us to measure the accuracy as result of *systematic parameter changes*. The well-studied stochastic block-model (Airoldi et al. 2008) leads to networks with degree distributions that are not similar to those found in most empirical network data. Our synthetic graph generator is thus a variant thereof with two important differences: (1) we actively control the degree distributions in the resulting graph; and (2) we "plant" exact graph properties (instead of fixing a property only in expectation). In other words, our generator preserves desired degree distribution and compatibilities between classes. The online appendix (Gatterbauer 2015) contains all details. We focus on the scenario of a network with one non-symmetric potential along each edge. The generator creates a graph using a tuple of parameters $(n, m, \boldsymbol{\alpha}, \boldsymbol{\psi}, \mathsf{dist})$, where n is the number of nodes, m is the number of edges, $\boldsymbol{\alpha}$ is the node label distribution with $\alpha(i)$ being the fraction of nodes of class i, $\boldsymbol{\psi}$ is the edge potential, and dist is a chosen degree distribution (e.g., uniform or power law with chosen coefficient).

Parameter choices. Throughout our experiments, we use $k = 3$ classes and the potential $\boldsymbol{\psi} = \left[\begin{smallmatrix} 1 & h & 1 \\ 1 & 1 & h \\ h & 1 & 1 \end{smallmatrix}\right]$, parameterized by a value h representing the ratio between min and max entries. Dividing by $(2 + h)$ centers it around 1. Thus parameter h models the *strength of the potential*, and we expect higher values of h to make our approximation less suitable. Notice that this matrix is not symmetric and shows very different modulation behavior across both directions of an edge. We create graphs with n nodes and assign the same fraction of nodes to one of the 3 classes: $\boldsymbol{\alpha} = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$. We also vary the parameters m and $d = \frac{m}{n}$ as the average in- and outdegree in the graph, and we assume a power law distribution with coefficient 0.3. We then keep a fraction f of node labels and measure accuracy on the remainder.

Computational setup. All methods are implemented in Python and use the optimized SciPy library (Jones et al. 2001) to handle sparse matrix operations. The experiments are run on a 2.5 Ghz Intel Core i5 with 16G of main memory and a 1TB SSD hard drive. To allow comparability across implementations, we limit evaluation to one processor. For timing BP, we use message-passing with division which is faster than standard BP on graphs with high node degree (Koller and Friedman 2009). To calculate the approximate spectral radius of a matrix, we use a method from the PyAMG library (Bell, Olson, and Schroder 2011) that implements a technique described in (Bai et al. 2000). Our code, including the data generator, is inspired by Scikit-learn (Pedregosa et al. 2011) and is available on Github to encourage reproducible research (SSLH 2015).

Question 1. *What is the effect of scaling parameter s on accuracy and number of iterations for convergence?*

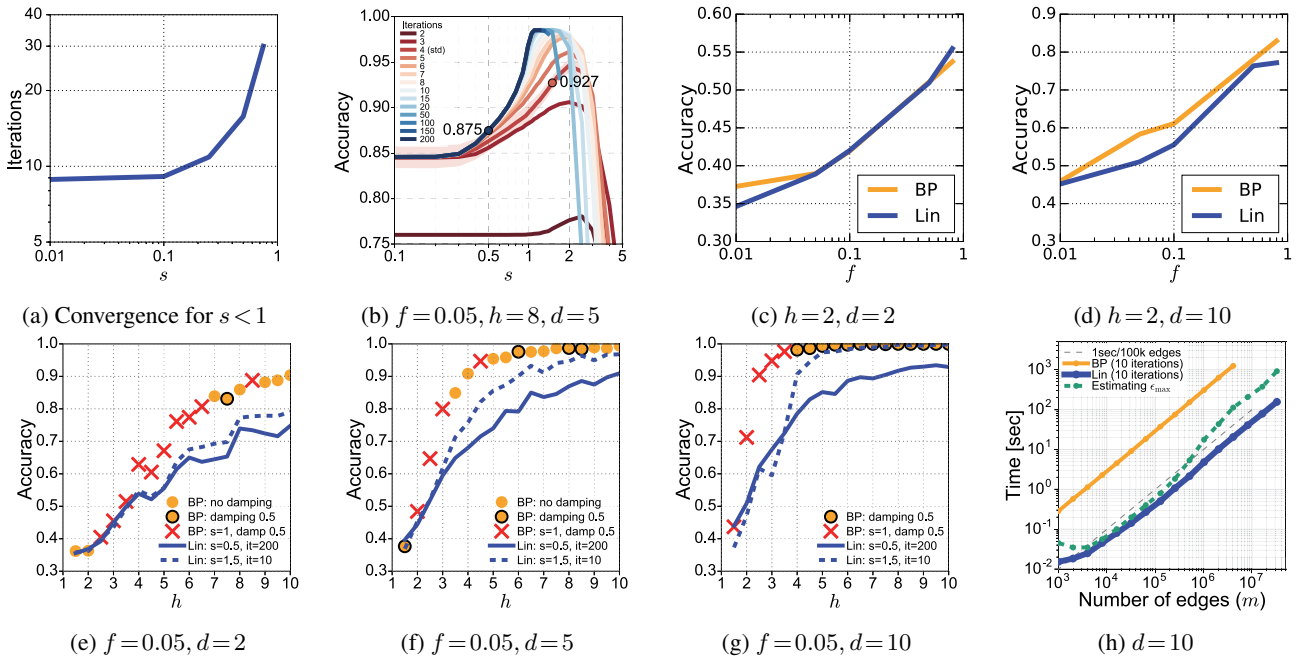
Result 1. Our scaling parameter s gives an exact criterion for our approach to converge. In contrast, BP often does not converge and requires a lot of fine-tuning; e.g., damping or even scaling of the potential. The accuracy of the linearization is highest for s close or slightly above 1 and by not iterating until convergence.

Figure 2a shows the number of required iterations to reach convergence and confirms our theoretical results from Corollary 6. In case the convergence condition does not hold, we scale the centered potential by a value ϵ, resulting from $\epsilon = s \cdot \epsilon_*$ with $s < 1$. This action weakens the potentials, but preserves the relative affinities (we also use the same approach to help BP find a fixed point if it does not converge within 200 iterations). Figure 2b shows what happens to accuracy if we run the iterative updates a fixed number of times as a function of s. Notice that even *considerably* scaling a potential does not entirely change the model and still gives reasonable approximations. The figure fixes a number of iterations, but then varies again ϵ via s. Also interestingly, almost all of the performance gains from the linearized update equations come from running just a few iterations, and convergence for optimal labeling is not necessary; instead, by choosing $s \approx 1$ (at the *exact boundary of convergence*) or even $s > 1$ and iterating only a few times, we can maximize the expected accuracy. For the remaining accuracy experiments, we use $s = 0.5$ and run our algorithm to convergence.

Question 2. *How accurate is our approximation, and under which conditions is it reasonable?*

Result 2. The linearization gives comparable labeling accuracy as LBP for graphs with weak potentials. The performance deteriorates the most in *dense* networks with *strong potentials*.

We found that h, d and f have important influence on the labeling accuracy of BP and its linearization (whereas n, dist and $\boldsymbol{\alpha}$ influence only to a lesser extent). Figures 2c and 2d show accuracy as a function of the fraction f of labeled nodes. Notice that we chose the best BP was able to perform (over several choices of ϵ and damping factors to make it converge) whereas for LinBP we consistently chose $s = 0.5$ as proposed in (Gatterbauer et al. 2015). Figures 2e to 2g show labeling quality as a function the strength h of the potential. For strong potentials ($h > 3$), BP gives better accuracy *if* it converges. In practice, BP often did not converge within 200 iterations even for weak potentials (bordered data points required dampening; red crosses required additional entry-wise scaling of the potential with our convergence boundary ϵ_*). In our experiments, BP often did not converge despite using damping, surprisingly often when h is not big. It is known that if the potentials are close to indifference then loopy BP usually converges. In this case, our formalism is equivalent to loopy BP (this follows from our linearization). Thus, whenever loopy BP did not converge,



(a) Convergence for $s<1$ (b) $f=0.05, h=8, d=5$ (c) $h=2, d=2$ (d) $h=2, d=10$

(e) $f=0.05, d=2$ (f) $f=0.05, d=5$ (g) $f=0.05, d=10$ (h) $d=10$

Figure 2: Experimental results for BP and its linearization (abbreviated here as "Lin"): f represents the fraction of labeled nodes, h the strength of potentials, and d the average node in- and outdegree. All graphs except for (h) have $n=1000$ nodes. (a): The *convergence parameter* s exactly determines convergence for the linearization. (b): Accuracy increases for s close to 1 and *few iterations*. (c, d): f only marginally affects the relative accuracy between BP and its linearization. (e, f, g): For strong potentials, BP gives better accuracy *if it converges*. In practice, BP often did not converge within 200 iterations even for weak potentials and required a lot of fine-tuning (damping and/or entry-wise scaling of the potential with our *convergence boundary* ϵ_* to $s=1$). (h): Each iteration of our approach is 50 times faster than an implementation of BP with division. In addition, deploying a proper damping strategy often requires 100s of iterations, which can bring up the total speed-up to a factor 1000 for some of the above data points. (Each data point results from at least 10 samples).

we simply exponentiated the entries of the potential with a varying factor ϵ until BP converged. Thus for high h, BP *can* perform better than the linearization, but only after a lot of fine-tuning of parameters. In contrast, for our formulation we know exactly the boundary of convergence.

Overall, the linearization gives comparable results to the original BP for small potentials, and BP performance is better than the linearization only either for strong potentials with $h \geq 3$ and dampening (see a few yellow dots without borders as exceptions) or after fine-tuning BP after using our own convergence boundary and scaling the potentials, or after a lot of manual fine-tuning.

Question 3. *How fast is the linearized approximation as compared to BP?*

Result 3. The linearization is around 100 times faster than BP per iteration and often needs 10 times fewer iterations until convergence. In practice, this can lead to a speed-up of 1000 times.

A key advantage of the linearization is that it has *predictable* convergence and comes with considerable speed-ups. Figure 2h shows that our approach scales linearly in the number of edges and is 50 times faster than regular loopy BP *per iteration*; an iteration on a graph with 3 million nodes and 30 million edges takes less than 2 sec. Calculating the exact convergence boundary via a spectral radius calculation can take more time (approx. 1000 sec for the same graph). Notice that any dampening strategy for BP results in increased number of iterations and needs to overcome the additional slow-down of further iterations. Also recall that on each circled point in Figs. 2e to 2g, BP did *not converge* within 200 iterations and required dampening; each red cross required *additional scaling* of the potentials with our calculated ϵ_* in order to make BP converge.

5 Conclusions

We have derived a linearization of BP for arbitrary pairwise MRFs for the purpose of node labeling with MM-inference. The approach transforms the parameters of an MRF into a linear equation system that can be solved with simple iterative updates. These updates come with exact convergence guarantees, allow a closed-form solution, keep the derived beliefs normalized at each step, and can thus be put into an efficient linear algebra framework that does not require normalization at each step. Experiments on carefully controlled synthetic data with known ground truth show that our approach performs comparably with Loopy BP for weak potentials and comes with a predictable behavior, compelling

computational advantages, and an easy implementation with only few lines of code. An unexplored application of the linearization may be speeding-up convergence of regular BP by starting from good approximations of its fixed points.

Acknowledgements. This work was supported in part by NSF grant IIS-1553547. I would like to thank Christos Faloutsos for very convincingly persuading me of the power of linear algebra and continued support. I am also grateful to Stephan Günneman, Vladimir Kolmogorov, and Christoph Lampert for a number of insightful comments.

References

Airoldi, E. M.; Blei, D. M.; Fienberg, S. E.; and Xing, E. P. 2008. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research* 9:1981–2014.

Bai, Z.; Demmel, J.; Dongarra, J.; Ruhe, A.; and van der Vorst, H. 2000. *Templates for the solution of algebraic eigenvalue problems*. SIAM.

Bell, W. N.; Olson, L. N.; and Schroder, J. B. 2011. PyAMG: Algebraic multigrid solvers in Python v2.0.

Domke, J. 2013. Learning graphical model parameters with approximate marginal inference. *IEEE Trans. Pattern Anal. Mach. Intell.* 35(10):2454–2467.

Donoho, D. L.; Maleki, A.; and Montanari, A. 2009. Message-passing algorithms for compressed sensing. *PNAS* 106(45):18914–18919.

Elidan, G.; McGraw, I.; and Koller, D. 2006. Residual belief propagation: Informed scheduling for asynchronous message passing. In *UAI*, 165–173.

Gatterbauer, W.; Günnemann, S.; Koutra, D.; and Faloutsos, C. 2015. Linearized and single-pass belief propagation. *PVLDB* 8(5):581–592.

Gatterbauer, W. 2015. The linearization of belief propagation on pairwise markov random fields. CoRR abs/1502.04956. (http://arxiv.org/abs/1502.04956).

Hazan, T., and Shashua, A. 2008. Convergent message-passing algorithms for inference over general graphs with convex free energies. In *UAI*, 264–273.

Heskes, T. 2006. Convexity arguments for efficient minimization of the Bethe and Kikuchi free energies. *J. Artif. Intell. Res. (JAIR)* 26:153–190.

Ihler, A. T.; Fisher III, J. W.; and Willsky, A. S. 2005. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research* 6:905–936.

Jones, E.; Oliphant, T.; Peterson, P.; et al. 2001. SciPy: Open source scientific tools for Python.

Koller, D., and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. MIT Press.

Korč, F.; Kolmogorov, V.; and Lampert, C. 2012. Approximating marginals using discrete energy minimization. In *ICML Workshop on Inferning: Interactions between Inference and Learning*.

Koutra, D.; Ke, T.-Y.; Kang, U.; Chau, D. H.; Pao, H.-K. K.; and Faloutsos, C. 2011. Unifying guilt-by-association approaches: Theorems and fast algorithms. In *ECML/PKDD (2)*, 245–260.

Krzakala, F.; Moore, C.; Mossel, E.; Neeman, J.; Sly, A.; Zdeborová, L.; and Zhang, P. 2013. Spectral redemption in clustering sparse networks. *PNAS* 110(52):20935–20940.

Malioutov, D. M.; Johnson, J. K.; and Willsky, A. S. 2006. Walk-sums and belief propagation in Gaussian graphical models. *Journal of Machine Learning Research* 7:2031–2064.

Meltzer, T.; Globerson, A.; and Weiss, Y. 2009. Convergent message passing algorithms – a unifying view. In *UAI*, 393–401.

Mooij, J. M., and Kappen, H. J. 2007. Sufficient conditions for convergence of the sum-product algorithm. *IEEE Transactions on Information Theory* 53(12):4422–4437.

Murphy, K. P.; Weiss, Y.; and Jordan, M. I. 1999. Loopy belief propagation for approximate inference: An empirical study. In *UAI*, 467–475.

Murphy, K. P. 2012. *Machine learning: a probabilistic perspective*. Adaptive computation and machine learning series. MIT Press.

Pearl, J. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Gallagher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI Magazine* 29(3):93–106.

SSLH. 2015. A Python package for Semi-Supervised Learning with Heterophily. http://github.com/sslh/sslh/.

Su, Q., and Wu, Y. 2015. On convergence conditions of Gaussian belief propagation. *IEEE Transactions on Signal Processing* 63(5):1144–1155.

Wainwright, M. J., and Jordan, M. I. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning* 1(1-2):1–305.

Weiss, Y., and Freeman, W. T. 2001. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation* 13(10):2173–2200.

Weiss, Y. 2000. Correctness of local probability propagation in graphical models with loops. *Neural Computation* 12(1):1–41.