The Linearization of Pairwise Markov Networks

Wolfgang Gatterbauer
Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213 USA

Abstract

Belief Propagation (BP) allows to approximate exact probabilistic inference in graphical models, such as Markov networks (also called Markov random fields, or undirected graphical models). However, no exact convergence guarantees for BP are known, in general. Recent work has proposed to approximate BP by linearizing the update equations around default values for the special case when all edges in the Markov network carry the same symmetric, doubly stochastic potential. This linearization has led to exact convergence guarantees, considerable speed-up, while maintaining high quality results in network-based classification (i.e. when we only care about the most likely label or class for each node and not the exact probabilities). The present paper generalizes our prior work on Linearized Belief Propagation (LinBP) with an approach that approximates Loopy Belief Propagation on any pairwise Markov network with the problem of solving a linear equation system.

1. Introduction

Belief Propagation (also called min-sum, or product-sum algorithm, or BP in short) is an iterative message-passing algorithm for performing inference on graphical models, such as factor graphs or Markov networks (MNs). It calculates the marginal distribution for each unobserved node, conditional on any observed nodes (Pearl, 1988). BP achieves this by propagating the information from a few explicitly labeled nodes throughout the network by iteratively passing information between neighboring nodes. It is also a very expressive formalism for semi-supervised learning or classification, i.e. assigning classes to unlabeled nodes. In that application, we are interested in the most likely “beliefs” (or labels) for all nodes in the graph. As such, BP has been used successfully in multiple settings for solving inference problems, such as error-correcting codes (Kschischang et al., 2001) or stereo imaging in computer vision (Felzenszwalb & Huttenlocher, 2006), fraud detection (McGlohon et al., 2009; Pandit et al., 2007), malware detection (Chau et al., 2011), graph similarity (Bayati et al., 2009), and pattern mining and anomaly detection (Kang et al., 2011).

It is known that when the graphical model has a tree structure, then BP reaches a stationary point (convergence to the true marginals) after a finite number of iterations. In loopy graphs, convergence to the correct marginals is not guaranteed. Furthermore, convergence in loopy graphs is not at all guaranteed, and using BP even for node classification leads to well-documented convergence problems (see (Sen et al., 2008) for a detailed discussion of the problems from a practitioner’s point of view). While there is a lot of work on convergence of BP, e.g., (Elidan et al., 2006; Ihler et al., 2005; Mooij & Kappen, 2007), exact criteria for convergence are not known (Murphy, 2012) and most existing bounds for BP on general pairwise Markov random fields give only sufficient convergence criteria.

In addition, various works attempts to speed up BP by exploiting the graph structure (Chechetka & Guestrin, 2010; Pandit et al., 2007), changing the order of message propagation (Elidan et al., 2006; Gonzalez et al., 2009; Mooij & Kappen, 2007), or using the MapReduce framework (Kang et al., 2011).

Two recent papers have suggested to solve the convergence and speed-up problems by linearizing the update equations: Koutra et al. (2011) linearized 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 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 (e.g., yes/no or male/female). Gatterbauer et al. (2015) derived a multivariate (“polytomous”) generalization of FaBP from binary to multiple labels called “Linearized Belief Propagation” (LinBP). Both papers have shown considerable speed-ups with high prediction accuracy through linearization of the BP update equations for the application of clas-
The Linearization of Pairwise Markov Networks

| BP | FaBP | LinBP (prior) | this paper |
|----|------|--------------|------------|
| # node types | arbitrary | 1 | arbitrary |
| # node classes | arbitrary 2 | const $k$ | arbitrary |
| # edge types | arbitrary | 1 | arbitrary |
| edge symmetry | arbitrary required | required | arbitrary |
| edge potential | arbitrary doubly stoch. | doubly stoch. | arbitrary |
| closed form | no | yes | yes |

Figure 1. The approach proposed in this paper combines the full expressiveness and generality of Belief Propagation (BP) on pairwise Markov networks with the computational advantages of Fast BP (FaBP) and Linearized (LinBP) for the semi-supervised case.

sification, i.e. scenarios where we are not interested in the exact final beliefs. We illustrate with an example.

**Example 1** (Node labeling functions). Assume that we have two different node labeling functions $f$ and $g$ that assign scores (weights) the three possible classes for each node in a network. Function $f$ follows a probabilistic semantics and returns a vector with probabilities that sum up to 1. For example, it may return $f(s) = \left[ \frac{1}{2}, \frac{1}{3}, \frac{1}{6} \right]$ for node $s$. In contrast, function $g$ returns relative weights that do not follow a probabilistic semantics, e.g., $g(s) = \left[ \frac{1}{12}, 0, \frac{1}{2} \right]$ for node $s$. Both labeling functions now allow us to determine class 3 as the most likely label for node $s$. For labeling (or classification) applications where we do not require a probabilistic semantics on the final labels, but are only interested in the top beliefs, we may choose $g$ if it comes with computational advantages (e.g., faster calculation, and guaranteed convergence).

Both above papers solve specialized cases: FaBP is restricted to two classes per node (de facto, one single score). LinBP can handle multiple classes, but is restricted to (i) only one node type, (ii) one edge type, and a potential that is (iii) symmetric, and (iv) doubly stochastic.

**Contributions.** This paper generalizes Linearized Belief Propagation (LinBP) to arbitrary pairwise Markov networks. LinBP transforms the parameters of any pairwise Markov network into an equation system that replaces multiplication with addition and that can thus be put into a matrix framework with a closed-form solution. Hence, unlike standard BP, LinBP (i) comes with exact convergence guarantees, (ii) allows closed-form solutions, (iii) gives a clear intuition about the algorithms, (iv) and has computational advantages. In addition over prior work (Fig.1) we remove any restrictions and solve the most general case of pairwise MRFs. This generalization is significant. For example, our prior LinBP formulation required the potential to be symmetric, which implied undirected edges (e.g. Alice being friend with Bob on Facebook is an undirected relation). We allow directed relations (e.g. Alice following Lady Gaga on Twitter has clearly different implications on Alice or Lady Gaga).

**Other related work.** Krzakala et al. (2013) study a form of linearization for unsupervised classification called “Spectral Redemption” in the stochastic block model. That model has no obvious way to include supervision in its setup. Donoho et al. (2009) propose “Approximate Message Passing” as an iterative thresholding algorithm for compressed sensing that is largely inspired by BP. While that work solves an overall different problem, their key insight appears to be a term in their update equations that is similar in nature to our echo cancellation term.

**Outline.** Sect. 2 provides necessary background on BP. Sect. 3 states our main results. Sect. 4 outlines the derivation of our results. Sect. 5 derives alternative, convenient formulations of important special cases, and Sect. 6 concludes. The appendix contains all proofs (Appendix A), a discussion of weighted edges (Appendix B), and a summary of the notation used throughout this paper (Fig.7).

## 2. Belief Propagation (BP) for Pairwise Markov Networks

An important subclass of undirected graphical models is that of pairwise Markov networks, representing distributions where all of the factors are over single variables or pairs of variables. More precisely, a pairwise Markov network over a graph $G$ is associated with a set of node potentials and a set of edge potentials (Koller & Friedman, 2009). The overall distribution is the normalized product of all of the potentials (both node and edge). 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 $j$. We denote by $x_s$ the vector of prior beliefs (also varying called explicit beliefs, or local evidence, or node potential) and $b_s$ the vector of posterior (or implicit or final) beliefs at node $s$, and require that $x_s$ and $b_s$ are normalized to 1, i.e. $\sum_j x_s(j) = \sum_j b_s(j) = 1$. Using $m_{us}$ for the $k_u$-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 each node as:

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

2Notice that we write $\sum_j$ as short form for $\sum_{j \in [k]}$ whenever $k$ is clear from the context (here and later, $[k] = \{1, 2, \ldots, k\}$).

Also notice that, commonly, the prior belief for class $j$ at node is denoted by $\phi_s(j)$, and the edge potential or evidence for value $i$ at node $t$ given value $j$ at node $s$ as $v_{st}(j, i)$ (Koller & Friedman, 2009). To increase readability (esp. with our bold vector and matrix notation) we decided to use $x_s(j)$ and $P_{st}(j, i)$ instead.
Here, we write \( Z_s \) for a normalizer that makes the elements of \( b_s \) sum up to 1. Thus, the posterior belief \( b_s(j) \) is computed by multiplying together the prior belief \( x_s(j) \) with the incoming messages \( m_{us}(j) \) from all neighbors \( u \in N(s) \setminus t \) and then normalizing so that the beliefs in all \( k_s \) classes sum up to 1. In parallel, each node sends messages to each of its neighbors:

\[
m_{st}(i) \leftarrow \frac{1}{Z_{st}} \sum_{j} P_{st}(j,i) \cdot x_s(j) \cdot \prod_{u \in N(s) \setminus t} m_{us}(j) \tag{2}
\]

where \( P_{st}(j,i) \) is a proportional “coupling weight” (or “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 \( P_{st} \) edge potential. Notice that scaling all elements of a message vector by the same constant does not affect the resulting beliefs since the normalizing factor in Eq. 1 makes sure that the beliefs are always normalized to 1, independent of the scaling of the messages. WLOG, we therefore use \( Z_{st} \) in Eq. 2 as a normalizer that makes the elements of \( m_{st} \) sum up to \( k_t \). This intermediate normalization of messages may seem redundant; it will, however, become essential for our later derivations.

BP repeatedly computes the above update equations for each node until the values (hopefully) converge. At iteration \( i \) of the algorithm, \( b_s(j) \) represents the posterior belief of \( j \) conditioned on the evidence that is \( i \) steps away in the network. Alternatively, the belief and message propagation updates can be written compactly in matrix notation by using the symbol \( \odot \) for the Hadamard product\(^3\) as

\[
\begin{align*}
b_s &\leftarrow \frac{1}{Z_s} (x_s \odot \prod_{u \in N(s)} m_{us}) \\
m_{st} &\leftarrow \frac{1}{Z_{st}} P_{st}^T (x_s \odot \prod_{u \in N(s) \setminus t} m_{us})
\end{align*}
\tag{3}
\tag{4}
\]

Note that the potential \( P_{st} \) is represented by a \( k_s \times k_t \)-dimensional matrix and that the transpose \( P_{st}^T = P_{ts} \) (see Fig. 2). This follows from the definition of a potential in a pairwise Markov network and the resulting derivation of belief propagation (Yedidia et al., 2003). Also notice that we could reduce the amount of necessary calculation by first multiplying all incoming messages at a node, and then dividing through the message a node indirectly sends to itself via a neighbor (we call this compensation “echo cancellation”). This approach is also called “message-passing with division” (Koller & Friedman, 2009) and can be made precise by defining a component-wise division operator by: \( Z = X \odot Y \Leftrightarrow Z(i,j) = X(i,j)/Y(i,j) \) where \( 0/0 = 0 \). Equation 4 can then be written more concisely as:

\[
m_{st} \leftarrow \frac{1}{Z_{st}} P_{st}^T (b_s \odot m_{ts}) \tag{5}
\]

3. Linearized Belief Propagation (LinBP) for Pairwise Markov Networks

This section gives a closed form description for the final beliefs after convergence of BP in arbitrary pairwise Markov networks under mild restrictions of all parameters. This is a strict and non-trivial generalization of our prior work (Gaterbauer et al., 2015) which addressed the case of a single, symmetric, doubly stochastic potential (Fig. 1). We need to solve here the following additional problems: (i) how to deal with non-symmetric potentials that modulate messages differently across both directions of an edge; (ii) how to deal with non-doubly stochastic matrices, i.e. matrices that are centered but whose rows or columns are not centered individually; (iii) how to deal with multiple potentials, (iv) multiple node types, and (v) different number of classes among nodes in the network? We still borrow the same formal setup and overall approach to center values around appropriate default values (using Maclaurin series) and to then restrict the parameters to small deviations from these defaults. The resulting equations replace multiplication with addition and can thus be put into a matrix framework with a closed form solution. This allows us to give exact convergence criteria based on the potentials. The approach is similar in spirit to the idea of writing any Markov network (with strictly positive density) as log-linear model, which allows techniques from matrix algebra to be applied. However, by starting from the update equations for loopy belief propagation, we copy its “solution” to the intractability problem: ignoring all dependencies between messages that have traveled over a path of length 2 or more.

Definition 2 (Centering). We call a vector or matrix \( x \) “centered around \( c \)” if the average of all entries is exactly \( c \) and each entry is close to \( c \).

Definition 3 (Residual vector/matrix). If a vector \( x \) is centered around \( c \), then the residual vector around \( c \) is defined as \( x = [x_1 - c, x_2 - c, \ldots]^T \). Accordingly, we denote a matrix \( X \) as a residual matrix if each entry is the residual
The Linearization of Pairwise Markov Networks

| Expression / Maclaurin series / Approximation |
|------------------------------------------------|
| Logarithm                                      |
| \[
\ln(1 + \epsilon) \approx \epsilon - \frac{\epsilon^2}{2} - \frac{\epsilon^3}{3} - \cdots
\]
| Product                                        |
| \[
\frac{1}{1 + \epsilon_1} \frac{1}{1 + \epsilon_2} = \frac{1 + \epsilon_1 + \epsilon_2 + \epsilon_1 \epsilon_2}{1 + \epsilon_1 + \epsilon_2} \approx 1 + \epsilon_1 + \epsilon_2
\]
| Division                                       |
| \[
\frac{1}{1 + \epsilon_1} \frac{1}{1 + \epsilon_2} = \frac{1}{1 + \epsilon_1} \frac{1}{1 + \epsilon_2} \approx 1 + \epsilon_1 + \epsilon_2
\]

Figure 3. Table of our linearizing approximations.

After centering around the same value \( c \).

For example, we call the vector \( \mathbf{x} = [1.01, 1.02, 0.97]^T \) centered around \( c = 1 \).\(^4\) The residuals from 1 form the residual vector \( \hat{\mathbf{x}} = [0.01, 0.02, -0.03]^T \) around 1 (in other words, \( \mathbf{x} = \mathbf{1}_3 + \hat{\mathbf{x}} \) where \( \mathbf{1}_3 \) is the 3-dimensional vector with all entries equal to 1). Note that the entries in a residual vector or matrix always sum up to 0, by construction.

The main idea of our derivation relies on the following observation: if we start with centered messages around 1 and set \( \frac{1}{\mathbf{Z}_{ij}} = k_i \), then the updated messages from Eq.2 remain centered around 1 for subsequent iterations (as centering the messages around 1 assures that multiplying with default messages equal to 1 does not change anything, irrespective of the number of edges). Hence, our equations do not require further normalization. We also assume prior beliefs for a node \( s \) to be centered around \( \hat{c}_s \) and we make use of each of the linearizing approximations shown in Fig. 3 exactly once.\(^5\)

We will further require that, for every potential \( \mathbf{P} \), its entries are approximately equal. WLOG, we start with potentials that are all centered around 1.\(^6\) We then “row-recenter” the potentials before using them (we will supply the intuition for this operation later in Sect.4):

**Definition 4 (Row-recentered residual matrix).** Let \( \mathbf{P} \in \mathbb{R}^{\ell \times k} \) be centered around 1 and \( \hat{\mathbf{P}} \) be the residual matrix with \( \hat{\mathbf{P}}(j,i) \) being the sum of the residuals of row \( j \). Then the row-recentered residual matrix \( \hat{\mathbf{P}}' \) has entries

\[
\hat{\mathbf{P}}'(j,i) \equiv \frac{1}{k} \left( \hat{\mathbf{P}}(j,i) - \frac{c(j)}{k} \right)
\]

Before we can give our main result, we need some more notation. WLOG, let \( N = \{n\} \) be the set of all nodes. For each node \( u \in N \), let \( k_u \) be its number of classes. Let \( k_u = \frac{1}{k_u} \mathbf{1}_{k_u} \), i.e. the \( k_u \)-dimensional uniform stochastic column vector. Furthermore, let \( k_{\text{tot}} \equiv \sum_{u \in N} k_u \) be the total number of classes across nodes. To write all our re-

\(^4\) Notice that, unless otherwise stated, all vectors in this paper are assumed to be column vectors even if written as row vectors.

\(^5\) We thus call “nodes with prior beliefs”, those nodes for which the residuals have non-zero elements (\( \hat{\mathbf{x}}_s \neq 0 \)), i.e. there is local evidence and the prior beliefs deviate from the center \( \hat{c}_s \).

\(^6\) Notice that any potential in a Markov network can be scaled so that the average entry is 1 without changing the joint probability distribution. For example, given \( \mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \), we divide all entries by 6 to get \( \hat{\mathbf{P}} = \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \end{bmatrix} \), which is centered around 1.

\( \hat{\mathbf{P}}' := \begin{bmatrix} \hat{\mathbf{P}}_{11}' & \cdots & \hat{\mathbf{P}}_{1n}' \\ \vdots & \ddots & \vdots \\ \hat{\mathbf{P}}_{n1}' & \cdots & \hat{\mathbf{P}}_{nn}' \end{bmatrix} \)

We also combine all row-recentered residual matrices into one large and sparse \( [k_{\text{tot}} \times k_{\text{tot}}] \)-square matrix (notice that all entries for non-existing edges remain empty):

We can now state our main theorem:

**Theorem 5 (Linearized Belief Propagation).** Let \( \hat{\mathbf{b}}, \hat{\mathbf{x}}, \hat{\mathbf{k}} \) and \( \hat{\mathbf{P}}' \) be the above defined residual vectors and matrix. Then, the final belief assignment from BP is approximated by the following system of \( k_{\text{tot}} \) linear equations in \( \hat{\mathbf{b}} \):

\[
\hat{\mathbf{b}} = \hat{\mathbf{x}} + \hat{\mathbf{P}}'^T \hat{\mathbf{k}} + \hat{\mathbf{P}}'^T \hat{\mathbf{b}} - \hat{\mathbf{P}}'^T \hat{\mathbf{2}} \hat{\mathbf{b}}
\]

Notice that the 2\( n \)-term \( \hat{\mathbf{P}}'^T \hat{\mathbf{k}} \) is a vector that depends only on the structure of the network and the potentials, but not the beliefs. We thus sometimes prefer to write \( \hat{\mathbf{c}}' \equiv \hat{\mathbf{P}}'^T \hat{\mathbf{k}} \) emphasizing it to remain constant during the iterations. This term vanishes if all potentials are doubly stochastic. Also notice that the 4\( n \)-term is the “echo cancellation.”\(^7\)

**Example 6 (LinBP).** Consider the network Fig. 4 consisting of nodes \( N = \{1, 2, 3\} \). Node 1 has three classes, whereas nodes 2 and 3 have two classes. We have two edges, e.g., the edge between nodes 1 and 2 with a 3 \( \times \) 2

\(^7\) Notice that the original 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 any neighbor \( s \) (\( t \rightarrow s \rightarrow t \)).

---

Page 4 of 14
The Linearization of Pairwise Markov Networks

potential $P_{12}$. Figure 5 illustrates Eq. 6. Notice that $\hat{c}_i' = \hat{P}^T k$ with $k = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)^T$ and $k_{tot} = 3 + 2 + 2$.

Further notice that the matrix $\hat{P}^{T^2}$ is block-diagonal (entries represent the total echo that a node receives through all its neighbors). We next illustrate the update equation for node 2. Here, we write $\langle \cdot \rangle_2$ for the projection of a stacked vector on the entries for node 2, e.g., $\langle \hat{b} \rangle_2 = b_2$:

$$\langle \hat{x} \rangle_2 = \hat{x}_2$$

$$\langle \hat{c}_i' \rangle_2 = \frac{1}{3} \hat{P}^T_{12} \hat{x}_1 + \frac{1}{2} \hat{P}^T_{23} \hat{x}_2$$

$$\langle \hat{P}^T b \rangle_2 = \hat{P}_{12}^T \hat{b}_1 + \hat{P}_{23}^T \hat{b}_3$$

$$\langle \hat{P}^T b \rangle_2 = \left( \sum_{j} \hat{P}_{i j}^T \hat{b}_j \right) \hat{x}_i$$

3.1. Closed-form solution for LinBP

In practice, we will solve the system of linear equations defined by Eq. 6 iteratively (see Sect. 3.2). However, we can also give a closed-form solution by simple algebraic manipulation and solving for $\hat{b}$. This closed-form will allow us later to study the convergence of the iterative updates.

**Corollary 7** (LinBP in closed-form). The closed-form solution for LinBP (Eq. 6) is given by:

$$\hat{b} = (I_{k_{tot}} - \hat{P}^T + \hat{P}^{T^2})^{-1} (\hat{x} + \hat{c}_i')$$  \hspace{1cm} (7)

3.2. Iterative Updates and convergence

While Eq. 6 gives an implicit definition of the final beliefs after convergence, it can also be used iteratively, allowing an iterative calculation of the final beliefs. Starting with an arbitrary initialization of $\hat{b}$ (e.g., all values zero), we repeatedly compute the right hand side of the equations and update the values of $\hat{b}$ until the process converges:

$$\hat{b}^{(i+1)} \leftarrow (\hat{x} + \hat{c}_i') + (\hat{P}^T - \hat{P}^{T^2}) \hat{b}^{(i)}$$  \hspace{1cm} (8)

These particular update equations allow us to give a sufficient and necessary criterion for convergence via the spectral radius of a matrix.\(^8\)

\(^8\)The spectral radius $\rho(\cdot)$ is the supremum among the absolute

**Proposition 8** (LinBP convergence). The update equation of LinBP (Eq. 8) converges if and only if:

$$\rho(\hat{P}^T - \hat{P}^{T^2}) < 1$$  \hspace{1cm} (9)

Thus, the updates converge towards the closed-form solution, and the final beliefs of each node can be computed via elegant matrix operations and optimized solvers, while the implicit form gives us guarantees for the convergence of this process.

4. Derivation of LinBP for pairwise MNs

This section outlines the derivation of Theorem 5; all proofs are in the appendix. Our approach is to center the elements of all message and belief vectors around their natural default values, i.e. the elements of $m_{ix}$ around 1, and the elements of $x_s$ and $b_s$ around $\frac{1}{k_s}$: $m(i) = 1 + \hat{m}(i), x_s(j) = \frac{1}{k_s} + \hat{x}_s(j)$, and $b_s(j) = \frac{1}{k_s} + \hat{b}_s(j)$.

WLOG, we start from a potential $P \in \mathbb{R}^{\ell \times k}$ that is centered around 1. We appropriately recenter a potential differently across both directions of an edge so that most of the residual terms for the belief update equations for each direction of an edge cancel each other out and the resulting equations simplify (Lemma 11 will provide some intuition why our chosen center points are the natural choice to simplify all later derivations). Definition 4 provided the definition for the residual matrix in one direction, row-recentering. Adding to that definition, the row-recentered stochastic matrix $P'$ is centered around $\frac{1}{k}$ and has entries $P'(j, i) = \hat{P}'(j, i) + \frac{1}{k}$. Both matrices are indicated with a single apostrophe ' . Analogously, let $c(i) = \sum_i \hat{P}(j, i)$ be the residual sum of column $i$. Then, a column-recentered residual matrix $P''$ has entries $P''(j, i) \equiv \frac{1}{\ell} \hat{P}(j, i) - \frac{c(i)}{\ell}$ and the column-recentered stochastic matrix $P'''$ has entries $P'''(j, i) \equiv \frac{1}{\ell} + \hat{P}(j, i)$. The resulting recentered residual potentials are coupling matrices that make explicit the relative attraction and repulsion of neighboring nodes. For example, the sign of $P''(j, i)$ tells us if the class $j$ attracts or repels class $i$ in a neighbor, and the magnitude of $P''(j, i)$ indicates the extent. Subsequently, this centering allows us to rewrite belief propagation in terms of the residuals.

Notice that column-recentering and row-recentering are connected via the transpose. Hence, message modulation across one direction of an edge is is not simply the transpose of the modulation across the other direction (!)

**Corollary 9** (Row-recentering vs. column-recentering). It holds: $(\hat{P}'')^T = (\hat{P}')'$. In particular, $\hat{P}_{st}''' = \hat{P}_{ts}''$. We also write the row and columns as $\ell$-dimensional values of the eigenvalues of the enclosed matrix.
vector \( r = P 1_k \) and \( k \)-dimensional vector \( c = P^T 1_k \), respectively. And we write \( s = \sum_j r(j) = 1^T r = 1^T P 1_k \) for the sum of all entries in a matrix. We illustrate recentering next with a detailed example.

**Example 10 (Recentering).** Figure 6 shows the \( 3 \times 2 \) matrix \( P \) that is centered around 1 (i.e. each entry is close to 1 and the average is exactly 1) together with the row sums \( r(j) \) and the column sums \( c(i) \). \( P \) is then the residual matrix. Notice that the centered residual matrices \( P' \) and \( P'' \) have zero row sums \( \hat{r}(j)' \) or column sums \( \hat{c}(i)' \), respectively. As consequence, the row-recentered matrix \( P' \) and column-recentered matrix \( P'' \) are row-stochastic or column-stochastic, respectively.

The following lemma provides the mathematical justification for our particular choice of recentering:

**Lemma 11 (Recentering).** Consider the update equation

\[
y \leftarrow \frac{1}{Z} P^T x
\]

with \( x \) being a \( l \)-dimensional stochastic vector, \( P \in \mathbb{R}^{l \times k} \) being centered around 1, and \( Z \) a normalizer that makes the elements of the resulting \( k \)-dimensional vector \( y \) sum up to \( k \). Then, the update equation can be approximated with the row-centered stochastic matrix \( \hat{P}' \) by

\[
y \leftarrow k \hat{P}'^T x
\]

Lemma 11 implies that by recentering the coupling matrix, we can replace the normalizer with a constant, which considerably simplifies our later derivations. The proof also shows that the approximation becomes exact if each row in \( P \) is centered around 1.

**Example 12 (Recentering (continued)).** Consider matrix \( P \in \mathbb{R}^{3 \times 2} \) in Fig. 6 and assume \( x = [0.1, 0.1, 0.8]^T \). Then \( y = [0.99021, 1.00979]^T \) for Eq. 10, but \( y = [0.99, 1.01]^T \) with Eq. 11. Thus, the residuals are \( \pm 0.00979 \) and \( \pm 0.01 \), respectively, and the relative difference \( \approx 2\% \).

By using the previous lemma and focusing on the residuals only, we can next transform the belief update equations from multiplication into addition:

**Lemma 13 (Centered BP).** By appropriately centering the coupling matrix, beliefs and messages, the equations for belief propagation can be approximated by:

\[
\hat{b}_s(j) \leftrightarrow \hat{x}_s(j) + \frac{1}{k_s} \sum_{u \in N(s)} \hat{m}_{us}(j)
\]

(12)

\[
\hat{m}_{st}(i) \leftrightarrow \frac{k_t}{k_s} \hat{c}_{st}(i)' + k_t \sum_j \hat{P}_{st}(j,i) \left( \hat{b}_s(j) - \frac{1}{k_s} \hat{m}_{ts}(j) \right)
\]

(13)

Eq.12 and Eq.13 can be written in matrix notation as:

\[
\hat{b}_s \leftarrow \left( \hat{x}_s + \frac{1}{k_s} \sum_{u \in N(s)} \hat{m}_{us} \right)
\]

(14)

\[
\hat{m}_{st} \leftarrow \frac{k_t}{k_s} \hat{c}_{st}' + k_t \hat{P}_{st}^T \left( \hat{b}_s - \frac{1}{k_s} \hat{m}_{ts} \right)
\]

(15)

An alternative way to write the message updates is

\[
\hat{m}_{st} \leftarrow \frac{k_t}{k_s} \hat{c}_{st}' + k_t \hat{P}_{st}^T \left( \hat{x}_s + \frac{1}{k_s} \sum_{u \in N(s)} \hat{m}_{us} \right)
\]

(16)

We invite the reader to compare Eq.14, Eq.15 (and Eq.16) with the original BP update equations Eq. 3, Eq. 5 (and Eq. 4). Notice that the first term \( \frac{k_t}{k_s} \hat{c}_{st}' \) is the result of allowing non-doubly stochastic potentials.

From Lemma 13, we can derive a closed-form equation for the message in steady-state of belief propagation:

**Lemma 14 (Steady state messages).** After convergence of belief propagation, message propagation can be approximated in terms of the steady centered beliefs as:

\[
\hat{m}_{st} = \frac{k_t}{k_s} \hat{c}_{st}' + k_t \hat{P}_{st}^T \left( \hat{x}_s + \frac{1}{k_s} \sum_{u \in N(s)} \hat{m}_{us} \right)
\]

(17)

Finally, by using or matrix notation introduced in Sect. 3, we can transform and write Eq. 17 for all nodes and edges together as one large equation system and get Theorem 5.
5. Special formulations

In this section, we derive alternative formulations of Theorem 5 for important special cases: edges with repeated potentials (Sect. 5.1), constant number of classes for all nodes in the network (Sect. 5.2), and one single symmetric, doubly stochastic potential (Sect. 5.3).

5.1. Edge types with repeated potentials

In many realistic scenarios, the number of edges is usually larger than the number of different edge types or edge potentials. For example, assume a set $T$ of different node types. We then have a $|T|$-partite network and each node with type $y \in T$ can be one of $k_y$ classes. Further assume that the couplings along an edge only depend on the types at both ends of the edge. Then there are max $|T|(|T| - 1)$ different row-recentered potentials irrespective of the size of the network (recall that we have one row-recentered potential for every edge direction, thus two for every edge type). For large graphs, our most general formulation of LinBP (Eq. 6) would require us to repeatedly store the same information in $\hat{P}'$. In the following, we reformulate the update equations so that every different row-recentered edge potential appears only once in the equations.

One complication in the following formulation is that different node types may have different numbers of classes. We are addressing this issue by creating separate matrices that contain the beliefs of nodes with the same number of classes. Concretely, let $N = [n]$ be the set of all nodes and let $K$ be the set of number of classes across all nodes. Let $N_k \subseteq N(k \in K)$ denote the set of nodes with $k$ classes so that all nodes are partitioned into groups $N_{k_1}, N_{k_2}, \ldots, N_{k_{|K|}}$. Let $n_k = |N_k|$ denote the size of group $k$. We assume a numbering of nodes such that $N_{k_1} = \{1, 2, \ldots, n_{k_1}\}, N_{k_2} = \{n_{k_1} + 1, n_{k_1} + 2, \ldots, n_{k_1} + n_{k_2}\}$, and so on. Given this convention, each node $s$ has a unique order $o_s$ within its group. For each $k \in K$, we create two $n_k \times k$ matrices $\hat{X}_k$ and $\hat{C}_k$ that contain the posterior and prior beliefs of all nodes with $k$ classes with different row-recentered potentials $\hat{P}_s$ and $\hat{P}_t$.

For each potential $P \in \mathbb{R}^{k \times k}$, we create two centered residual potentials $\hat{P}' \in \mathbb{R}^{k \times k}$ and $(\hat{P}')' = \hat{P}^\top \in \mathbb{R}^{k \times k}$ that correspond to the two modulations across the two directions of an edge. For notational convenience, we treat them as two distinct potentials and ignore their common ancestry. For example, $P_{12} \in \mathbb{R}^{3 \times 2}$ leads to $\hat{P}'_{12} \in \mathbb{R}^{3 \times 2}$ and $\hat{P}'_{21} = P_{12} \in \mathbb{R}^{2 \times 3}$. For each newly created row-recentered potential $\hat{P}' \in \mathbb{R}^{k \times k}$, we create two new matrices: the adjacency matrix $A_{\hat{P}'} \in \mathbb{R}^{n_k \times n_k}$ with $A_{\hat{P}'}(o_s, o_t) = 1$ if node $s$ with $\ell$ classes is connected to node $t$ with $k$ classes via an edge potential $\hat{P}'$; and the block-diagonal in-degree matrix $D_{\hat{P}'} \in \mathbb{R}^{n_k \times n_k}$ with $D_{\hat{P}'}(o_t, o_t) = d$ if there are $d$ number of nodes $s$ that are connected to $t$ via an edge potential $\hat{P}'$ (notice that $\hat{P}'$ works along the direction $s \rightarrow t$).

Proposition 15 (LinBP with edge types). Let $\hat{P}'$ bet the set of all row-partitioned potentials, $\mathcal{P}^{r \times k} \subset \mathcal{P}'$ be the subset with dimensions $\ell \times k$, and let $\hat{B}_k, \hat{X}_k, \hat{A}_{\hat{P}'}', D_{\hat{P}'}$ be the previously defined partitioned matrices for all $k \in K$, and $\hat{P}' \in \mathcal{P}'$. The LinBP update equations can then be written $\forall k \in K$ as follows:

$$\hat{B}_k \leftarrow \hat{X}_k + \hat{C}_k + \sum_{\hat{P}'} (A_{\hat{P}'} \hat{B}_k \hat{P}' - D_{\hat{P}'} \hat{B}_k \hat{P}_s)$$

with $\hat{C}_k = \sum 1/2 \hat{A}^T_{\hat{P}'} [1]_{n_k \times \ell} \hat{P}'$ and $\hat{P}_s = \hat{P}'^\top \hat{P}'$.

Example 16 (Example 6 continued). We use our example from Fig. 5 to illustrate Proposition 15. Let $b_{2j}$ be the belief of node $s$ in class $j$. We create two belief matrices $\hat{B}_2, \hat{B}_3, \hat{B}_3$ with corresponding echo cancellation potentials (e.g., $\hat{P}_{12} = \hat{P}_{21} \hat{P}_{12}$), and appropriate adjacency and in-degree matrices. For example, $\hat{P}_{12} \in \mathbb{R}^{3 \times 2}$ has $A_{\hat{P}_{12}} = [1 \ 0]$ where the first entry indicates an edge from node 1 to node 2. We illustrate next in detail:

$$A_{\hat{P}_{12}} = \begin{bmatrix} 1 & \frac{3}{2} \\ 3 & 1 \end{bmatrix}, A_{\hat{P}_{13}} = \begin{bmatrix} \frac{3}{2} & 1 \\ 0 & 1 \end{bmatrix}, A_{\hat{P}_{23}} = \begin{bmatrix} 2 & \frac{3}{2} \\ 3 & 1 \end{bmatrix}, A_{\hat{P}_{31}} = \begin{bmatrix} \frac{3}{2} & 1 \\ 0 & 1 \end{bmatrix},$$

$$D_{\hat{P}_{12}} = \begin{bmatrix} 2 & \frac{3}{2} \\ 3 & 1 \end{bmatrix}, D_{\hat{P}_{21}} = \begin{bmatrix} \frac{3}{2} & 1 \\ 0 & 1 \end{bmatrix}, D_{\hat{P}_{23}} = \begin{bmatrix} 2 & \frac{3}{2} \\ 3 & 1 \end{bmatrix}, D_{\hat{P}_{31}} = \begin{bmatrix} \frac{3}{2} & 1 \\ 0 & 1 \end{bmatrix}.$$
5.2. Networks with constant number of classes

Proposition 15 simplifies considerably when all nodes have the same number of classes:

**Corollary 17 (LinBP with constant k).** Let \( k \) be the number of classes for each node in the graph, \( \mathcal{P}' \) be the set of row-recentered residual edge potentials (all with \( k \times k \) dimensions), \( \mathbf{B} \) and \( \mathbf{X} \) the \( n \times k \) dimensional final and explicit belief matrices, and \( \mathbf{A}_{\mathbf{P}} \) and \( \mathbf{D}_{\mathbf{P}}^{\text{in}} \) the adjacency and in-degree matrices for each potential \( \mathbf{P} \in \mathcal{P}' \). The update equations for LinBP can then be simplified to:

\[
\hat{\mathbf{B}} \leftarrow \hat{\mathbf{X}} + \hat{\mathbf{C}}' + \sum_{\mathbf{P}' \in \mathcal{P}'} (\mathbf{A}_{\mathbf{P}}^\top \hat{\mathbf{B}} \mathbf{P}' - \mathbf{D}_{\mathbf{P}}^{\text{in}} \hat{\mathbf{B}} \mathbf{P}' ) \quad (19)
\]

with \( \hat{\mathbf{C}}' = \frac{1}{k} \sum_{\mathbf{P}' \in \mathcal{P}'} \mathbf{A}_{\mathbf{P}}^\top \hat{\mathbf{P}} \mathbf{P}' \) and \( \hat{\mathbf{P}}_s = \hat{\mathbf{P}}^\top \mathbf{P}' \).

Also the convergence criterion and the closed-form solution allow very concise formulations. For that we need to introduce two new notions: Let \( x_j \) denote the \( j \)-th column of matrix \( \mathbf{X} \) (i.e. \( \mathbf{X} = \{ x_{ij} \} = [x_1 \ldots x_n] \) and let \( \mathbf{X} \) and \( \mathbf{Y} \) be matrices of order \( m \times n \) and \( p \times q \), respectively. First, the vectorization of a matrix \( \mathbf{X} \) stacks its columns one underneath the other to form a single column vector:

\[
\text{vec}(\mathbf{X}) = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}
\]

Second, the Kronecker product of \( \mathbf{X} \) and \( \mathbf{Y} \) is the \( mp \times nq \) matrix defined by

\[
\mathbf{X} \otimes \mathbf{Y} = \begin{bmatrix} x_{11} \mathbf{Y} & \ldots & x_{1n} \mathbf{Y} \\ \vdots & \ddots & \vdots \\ x_{mn} \mathbf{Y} & \ldots & x_{mn} \mathbf{Y} \end{bmatrix}
\]

With these notations, Proposition 8 now becomes

**Proposition 18 (LinBP convergence with constant k).** Update Eq.19 converges if and only if \( \rho(\mathbf{M}) < 1 \) for

\[
\mathbf{M} = \sum_{\mathbf{P}' \in \mathcal{P}'} (\mathbf{A}_{\mathbf{P}}^\top \otimes \hat{\mathbf{P}}^\top \mathbf{P}' - \mathbf{D}_{\mathbf{P}}^{\text{in}} \otimes \hat{\mathbf{P}}^\top \mathbf{P}' )
\]

Furthermore, let \( \hat{\mathbf{b}} \equiv \text{vec}(\hat{\mathbf{B}}^\top) \), \( \hat{\mathbf{x}} \equiv \text{vec}(\hat{\mathbf{X}}^\top) \), and \( \hat{\mathbf{c}}_s' \equiv \text{vec}(\hat{\mathbf{C}}_s^\top) \). The closed-form solution of Eq. 19 is given by:

\[
\hat{\mathbf{b}} = (\mathbf{I}_{nk} - \mathbf{M})^{-1} (\hat{\mathbf{x}} + \hat{\mathbf{c}}_s') \quad (20)
\]

Notice that Eq. 20 is a special case of Corollary 7 that factors out repeated edge potentials. This concise factorization with the Kronecker product is only possible for constant \( k \).
The Linearization of Pairwise Markov Networks

References

Bayati, Mohsen, Gerritsen, Margot, Gleich, David, Saberi, Amin, and Wang, Ying. Algorithms for large, sparse network alignment problems. In ICDM09, pp. 705–710, 2009.

Chau, Duen Horng,Nachengen, Carey, Wilhelm, Jeffrey, Wright, Adam, and Faloutsos, Christos. Polonium: Terascale graph mining and inference for malware detection. In SDM, pp. 131–142, 2011. (former title: Large Scale Graph Mining and Inference for Malware Detection).

Chechetka, Anton and Guestrin, Carlos. Focused belief propagation for query-specific inference. In UAI, pp. 165–173, 2006. http://arxiv.org/pdf/1206.6837.

Felzenszwalb, Pedro F. and Huttenlocher, Daniel P. Efficient belief propagation for early vision. Int. J. Comput. Vision, 70(1):41–54, October 2006. ISSN 0920-5691. doi: 10.1007/s11263-006-7899-4. URL http://dx.doi.org/10.1007/s11263-006-7899-4.

Gatterbauer, Wolfgang, Günemann, Stephan, Koutra, Danai, and Faloutsos, Christos. Linearized and single-pass belief propagation. PVLDB, 8(5):581–592, 2015. URL http://www.vldb.org/pvldb/vol8/p581-gatterbauer.pdf.

Gonzalez, Joseph, Low, Yucheng, and Guestrin, Carlos. Residual splash for optimally parallelizing belief propagation. Journal of Machine Learning Research - Proceedings Track, 5:177–184, 2009.

Henderson, Harold V. and Searle, S. R. The vec-permutation matrix, the vec operator and kronecker products: a review. Linear and Multilinear Algebra, 9 (4):271–288, 1981. doi: 10.1080/03081088108817379.

Ihler, Alexander T., Fisher III, John W., and Willsky, Alan S. Loopy belief propagation: Convergence and effects of message errors. Journal of Machine Learning Research, 6:905–936, 2005. URL http://www.jmlr.org/papers/v6/ihler05a.html.

Kang, U., Chau, Duen Horng, and Faloutsos, Christos. Mining large graphs: Algorithms, inference, and discoveries. In ICDE, pp. 243–254, 2011.

Koller, Daphne and Friedman, Nir. Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning. The MIT Press, 2009. ISBN 0262013193, 9780262013192.

Koutra, Danai, Ke, Tai-You, Kang, U., Chau, Duen Horng, Pao, Hsing-Kuo Kenneth, and Faloutsos, Christos. Unifying guilt-by-association approaches: Theorems and fast algorithms. In ECML/PKDD (2), pp. 245–260, 2011.

Kschischang, Frank R., Frey, Brendan J., and Loeliger, Hans-Andrea. Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory, 47(2): 498–519, 2001.

McGlohon, Mary, Bay, Stephen, Anderle, Markus G., Steier, David M., and Faloutsos, Christos. SNARE: a link analytic system for graph labeling and risk detection. In KDD, pp. 1265–1274, 2009.

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

Murphy, Kevin P. Machine learning: a probabilistic perspective. Adaptive computation and machine learning series. MIT Press, Cambridge, MA, 2012. ISBN 9780262018029.

Pandit, Shashank, Chau, Duen Horng, Wang, Samuel, and Faloutsos, Christos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In WWW, pp. 201–210, 2007.

Pearl, Judea. Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann Publishers, San Mateo, Calif., 1988. ISBN 0934613737.

Roth, W. E. On direct product matricies. Bull. Amer. Math. Soc., 40:461–468, 1934.

Saad, Y. Iterative methods for sparse linear systems. SIAM, 2nd ed edition, 2003. ISBN 0898715342. URL http://www.loc.gov/catdir/enhancements/fy0665/2002044644-d.html.
The Linearization of Pairwise Markov Networks

| n | number of nodes |
|---|----------------|
| s, t, u | indices used for nodes |
| N(s) | list of neighbors for node s |
| k, ℓ | number of classes |
| i, j, g | indices used for classes |
| x, | k-dimensional prior (or explicit) belief vector of node s |
| b, j | k-dimensional posterior (implicit, final) belief vector of node s |
| m | k-dimensional message vector from node s to node ℓ (s → ℓ) |
| P | ℓ × k potential (or coupling matrix): P(j, i) indicates the influence of class j by node i |

For example, \( \hat{r}(j)_{2 \times 3} = \begin{bmatrix} -0.06 & 0 & 0.06 \\ -0.06 & 0 & 0.06 \end{bmatrix} \) for \( P \) in Fig. 6.

(i) Eq. 10: We calculate \( y \) in two steps that treat the normalization separately: first \( z = P^T x \), and then \( y = \frac{1}{z} z \).

\[
\begin{align*}
z &= P^T x \\
&= \left( \begin{bmatrix} 1 \end{bmatrix}_k \times \ell \ + \ \frac{1}{k} \hat{r}(j)_{k \times \ell} \ + \ k \hat{P}^T \hat{x} \right) \cdot \left( \frac{1}{\ell} \begin{bmatrix} 1 \end{bmatrix}_\ell + \hat{x} \right) \\
&= 1_k + \frac{1}{k} \hat{r}(j)_{k \times \ell} \ + \ k \hat{P}^T \hat{x} \\
&= 1_k + \frac{1}{k} \hat{r}x_k + k \hat{c} + k \hat{P}^T \hat{x} \\
\end{align*}
\]

We next calculate the value of the normalizer. Recall that the normalizer makes the entries of the resulting vector sum up to \( k \).

\[
\begin{align*}
Z &= \frac{1}{k} 1^T z \\
&= \frac{1}{k} \left( k + \hat{r}x + k^2 + \frac{1}{\ell} 1^T \hat{c} + k^2 \cdot \frac{1}{\ell} P^T \hat{x} \right) \\
&= 1 + \frac{\hat{r}x}{\ell}
\end{align*}
\]

We see that the normalizer is not a constant but also depends on \( P \) and \( x \). However, notice that if each row of \( P \) is centered exactly around 1, then \( \hat{r}(j) = 0 \) for all rows and, thus, \( Z = 1 \). In the following, we approximate \( 1/(1+\epsilon) \approx (1-\epsilon) \) and \( (1+\epsilon_1)(1+\epsilon_2) \approx (1+\epsilon_1+\epsilon_2) \).

\[
\[ y = \left( 1_k + \frac{1}{k} \hat{r}x_k + k \hat{c} + k \hat{P}^T \hat{x} \right) \left( 1 + \frac{\hat{r}x}{\ell} \right)^{-1} \\
&= 1_k + \frac{1}{k} \hat{r}x_k + k \hat{c} + k \hat{P}^T \hat{x} \\
&= 1_k + 1_k \hat{r}x_k + k \hat{c} + k \hat{P}^T \hat{x} - \frac{1}{k} \hat{r}x_k \\
&= 1_k + \frac{k}{\ell} \hat{c} + k \hat{P}^T \hat{x}
\]

Notice that the last equation is exact if \( \hat{r}(j) = 0 \) for all rows.

(ii) Eq. 11: Here we get the same result much faster:

\[
\begin{align*}
y &= k \hat{P}^T x \\
&= \left( \begin{bmatrix} 1 \end{bmatrix}_k \times \ell \ + \ \frac{1}{\ell} \begin{bmatrix} 1 \end{bmatrix}_\ell + \hat{x} \right) \\
&= 1_k + \frac{k}{\ell} \hat{c} + k \hat{P}^T \hat{x}
\end{align*}
\]

It follows that Eq. 11 is an approximation of Eq. 10, in general, and both equations are equivalent if each row in \( P \) is centered exactly around 1.

A. Proofs

A.1. Lemma 11: Recentering

**Proof Lemma 11.** Our proof will express both equations (Eq. 10 with \( P \) and Eq. 11 with \( P' \)) in terms of the residual matrix \( P' \) and show that they lead to the same equation. From Definition 4 and the definitions in Sect. 4, we know that \( P(j, i) = 1 + \hat{P}(j, i) \) and \( \hat{P}(j, i) = k P'(j, i) + \frac{\hat{r}(j)}{k} \).

Therefore, \( P(j, i) = 1 + k \hat{P}(j, i) + \frac{\hat{r}(j)}{k} \). Similarly, \( P'(j, i) = \frac{1}{k} + P'\). We will write \[ x \] \[ x \] for a \( k \times \ell \)-dimensional matrix with all entries equal to \( x \). We will also write \[ \hat{r}(j) \] \[ k \times \ell \] as a short form for a \( k \times \ell \)-dimensional matrix where the entries of the \( j \)-th column are equal to the \( j \)-th residual row sum for matrix \( \hat{P} \).

For example, \[ \hat{r}(j)_{2 \times 3} = \begin{bmatrix} -0.06 & 0 & 0.06 \\ -0.06 & 0 & 0.06 \end{bmatrix} \) for \( P \) in Fig. 6.
Also notice that, since \( y(j) = 1 + \hat{y}(j) \), we can express the update equation in terms of residuals as
\[
\hat{y} = \frac{k}{\ell} \hat{c}' + k \hat{P}'^\top \hat{x}
\]
Further notice that if each column in the original potential is centered around 1, then the term \( \hat{c}' \) disappears. □

A.2. Lemma 13: Centered BP

Proof Lemma 13. (i) Equation 12: Substituting the expansions into the belief updates Eq. 1 leads to
\[
\frac{1}{k_s} + \hat{b}_s(j) \leftarrow \frac{1}{Z_s} \cdot \left( \frac{1}{k_s} + \hat{x}_s(j) \right) \cdot \prod_{u \in N(s)} \left( 1 + \hat{m}_s(j) \right)
\]
\[
\ln \left( 1 + k_s \hat{b}_s(i) \right) \leftarrow -\ln Z_s + \ln(1 + k_s \hat{x}_s(j)) + \sum_{u \in N(s)} \ln(1 + \hat{m}_s(j))
\]
\[
k_t \hat{b}_s(j) \leftarrow -\ln Z_s + k_t \hat{x}_s(j) + \sum_{u \in N(s)} \hat{m}_s(j)
\]
(23)
For the last step, we use the approximation \( \ln(1 + \epsilon) \approx \epsilon \) for small \( \epsilon \). Summing both sides over \( j \) gives us:
\[
k_s \sum_j \hat{b}_s(j) \leftarrow -k_s Z_s + k_s \sum_j \hat{x}_s(j) + \sum_{u \in N(s)} \hat{m}_s(j)
\]
Hence, we see that \( \ln Z_s \) needs to be 0, and therefore our normalizer is actually a normalization constant and independent for all nodes \( Z_s = 1 \). Plugging \( Z_s = 1 \) back into Eq.23 leads to Eq.12:
\[
\hat{b}_s(j) \leftarrow \hat{x}_s(j) + \frac{1}{k_s} \sum_{u \in N(s)} \hat{m}_s(j)
\]

(ii) Equation 13: Using Lemma 11, we can write Eq.2 as follows (recall that \( k_t \) and \( H' \) take care of the normalization):
\[
m_s(i) \leftarrow k_t \sum_j P'_s(j,i) x_s(j) \prod_{u \in N(s) \setminus t} m_u(j)
\]
(24)
By further using Eq.12, we get:
\[
\leftarrow k_t \sum_j P'_s(j,i) \frac{x_s(j) \prod_{u \in N(s)} m_u(j)}{m_s(j)}
\]
\[
\leftarrow k_t \sum_j P'_s(j,i) \frac{\hat{b}_s(j)}{m_s(j)}
\]
(25)
Then, using the centering together with the approximation
\[
\frac{\frac{\hat{x} + \epsilon_1}{\hat{x} + \epsilon_2}}{\frac{1}{\hat{x} + \epsilon_2}} \approx \frac{1}{k} + \frac{\epsilon_1}{k} - \frac{1}{k} \epsilon_2 \text{ for small } \epsilon_1, \epsilon_2, \text{ we get:}
\]
\[
1 + m_{s,t}(i) \leftarrow k_t \sum_j \left( \frac{1}{k_t} + P'_s(j,i) \right) \frac{1}{k} + \hat{b}_s(j)
\]
\[
\leftarrow k_t \sum_j \left( \frac{1}{k_t} + P'_s(j,i) \right) \left( \frac{1}{k_s} + \hat{b}_s(j) - \frac{1}{k_s} \hat{m}_s(j) \right)
\]
\[
\leftarrow k_t \left( \frac{1}{k_t} + \frac{1}{k_s} \sum_j P'_s(j,i) \right) + \frac{1}{k_t} \sum_j \hat{b}_s(j) + \sum_j \hat{P}'_{s,t}(j,i) \hat{b}_s(j)
\]
\[
- \frac{1}{k_s k_t} \sum_j \hat{m}_s(j) - \frac{1}{k_s} \sum_j \hat{P}'_{s,t}(j,i) \hat{m}_s(j)
\]
(26)

A.3. Lemma 14: Steady state messages

Proof Lemma 14. To increase the readability of this proof, we ignore the subscripts in \( P_{s,t}, c_{s,t}, r_{s,t} \), and write instead \( P, c, r \), respectively. We start by writing Eq.13 for \( m_{s,t}(j) \) for both directions across the same edge:
\[
m_{s,t}(i) \leftarrow k_t \hat{c}(i)' + k_t \sum_{j=1}^{k_t} \hat{P}'(j,i) \left( \hat{b}_s(j) - \frac{1}{k_s} \hat{m}_s(j) \right)
\]
\[
m_{t,s}(j) \leftarrow k_t \hat{r}(j)' + k_t \sum_{g=1}^{k_t} \hat{P}''(g,j) \hat{b}_s(g) - \frac{1}{k_t} \hat{m}_{s,t}(j)
\]
We then simply combine both equations into one:
\[
m_{s,t}(i) \leftarrow k_t \hat{c}(i)' + k_t \sum_{j=1}^{k_t} \hat{P}'(j,i) \hat{b}_s(j) - \frac{k_t}{k_s} \sum_{j=1}^{k_t} \hat{P}'(j,i) \hat{m}_{s,t}(j)
\]
\[
\left( k_t \hat{r}(j)' + k_t \sum_{g=1}^{k_t} \hat{P}''(g,j) \hat{b}_s(g) - \frac{k_t}{k_t} \sum_{j=1}^{k_t} \hat{P}'(j,i) \hat{m}_{s,t}(j) \right)
\]
Now, if the equations converge, then \( \hat{m}_{s,t}(g) \) on both the left and right side of the equation need to be equivalent. We can, therefore, replace the update symbol with equality and group related terms together:
\[
m_{s,t}(i) - k_t k_s \hat{c}(i)' - k_t k_s \sum_{j=1}^{k_s} \hat{P}'(j,i) \hat{r}(j)'' + k_t \sum_{j=1}^{k_t} \hat{P}'(j,i) \hat{b}_s(j)
\]
\[
- k_t k_s \sum_{j=1}^{k_t} \hat{P}'(j,i) \hat{r}(j)'' + k_t \sum_{j=1}^{k_t} \hat{P}'(j,i) \hat{b}_s(j)
\]
(26)
This equation can then be written in matrix notation as:
\[
(I_{k_t} - \hat{P}'^\top \hat{P}''') \hat{m}_{s,t}
\]
\[
= k_t \hat{c}' - \hat{P}'^\top \hat{r}' + k_t \hat{P}'^\top \hat{b}_s - k_t P'^\top P'' \hat{b}_t
\]
Recall that \( \hat{c}' = \hat{P}^T 1_{k_s} \) and \( \hat{r}'' = \hat{P}'' 1_{k_t} \).

\[
(I_{k_t} - \hat{P}^T \hat{P}'') \hat{m}_{st} = (k_{t} \hat{P}^T 1_{k_s} - \hat{P}^T \hat{P}'' 1_{k_t}) + k_t \hat{P}^T \hat{b}_s - k_t \hat{P}^T \hat{P}'' \hat{b}_t
\]

\[
= \hat{P}^T (k_{t} 1_{k_s} + k_t \hat{b}_s) - \hat{P}'' (1_{k_t} + k_t \hat{b}_t)
\]

If all entries of \( \hat{P} \) are appropriately small (\(|\hat{P}| \ll 1\)), then the inverse of \((I_{k_t} - \hat{P}^T \hat{P}'')\) always exists. Thus, by further substituting \(\hat{P}_s^T := (I_{k_t} - \hat{P}^T \hat{P}'')^{-1} \hat{P}^T\), we can write:

\[
\hat{m}_{st} = \hat{P}_s^T (k_{t} 1_{k_s} + k_t \hat{b}_s) - \hat{P}_s'' (1_{k_t} + k_t \hat{b}_t)
\]

(27)

By further substituting \(\hat{h}' := k_{t} \hat{P}_s^T 1_{k_s} - \hat{P}_s'' 1_{k_t}\), we get the following exact equation for the message updates after convergence of belief propagation:

\[
\hat{m}_{st} = \hat{h}' + k_t \hat{P}_s^T (\hat{b}_s - \hat{P}_s'' \hat{b}_t)
\]

(28)

Next notice that \(\hat{P}^T \approx \hat{P}'\), since \(I_{k_t} \gg |\hat{P}^T \hat{P}'|\), and therefore \((I_{k_t} - \hat{P}^T \hat{P}'')^{-1} \approx I_{k_t}\). From this, we can now also approximate \(\hat{h} \approx \frac{k_{t}}{k_s} \hat{P}_s^T (\hat{1}_{k_t} - \hat{P}'') (1_{k_s} + k_t \hat{b}_t)\). Further ignoring the second term, we get \(\hat{h} \approx \frac{k_{t}}{k_s} \hat{c}'\). Plugging back into Eq. 28 finally gives us Eq. 17.

A.4. Theorem 5: Linearized Belief Propagation

Proof Proposition 5. For steady-state, we can write Eq. 12 in vector form as:

\[
\hat{b}_s = \hat{x}_s + \frac{1}{k_s} \sum_{u \in N(s)} \hat{m}_{us}
\]

By permuting subscripts, we can also write Eq. 17 as

\[
\hat{m}_{us} = k_{u} \hat{c}'_{us} + k_s \hat{P}_{us} (\hat{b}_u - \hat{P}_{us} \hat{b}_s)
\]

Combining the last two equations, we get

\[
\hat{b}_s = \hat{x}_s + \sum_{u \in N(s)} \frac{\hat{c}'_{us}}{k_u} + \sum_{u \in N(s)} \hat{P}_{us} (\hat{b}_u - \hat{P}_{us} \hat{b}_s)
\]

(29)

By using our combined new vectors and matrices \(\hat{b}, \hat{x}, \hat{k},\) and \(\hat{P}\) (and analogously for the column-centered residual matrix \(\hat{P}''\)), we can write Eq. 29 in matrix form as:

\[
\hat{b} = \hat{X} + \hat{P}^T \hat{k} + \hat{P}^T \hat{b} - \hat{P}^T \hat{P}'' \hat{b}
\]

(30)

From Corollary 9, we know \(\hat{P}_{ij}^T = \hat{P}''_{ij}\). Therefore, from our construction we also have \(\hat{P}^T = \hat{P}''\). We thus get

\[
\hat{b} = \hat{x} + \hat{P}^T \hat{k} + \hat{P}^T \hat{b} - \hat{P}^T \hat{P}'' \hat{b}
\]

(31)

Equation 31 is now a straight-forward linear equation system that can be solved to derive Theorem 5.

A.5. Proposition 8: LinBP convergence

Proof Proposition 8. From the Jacobi method for solving linear systems (Saad, 2003), we know that the solution for \(y = (I - M)^{-1} x\) can be calculated (under certain conditions) via the iterative update equation

\[
y^{(i+1)} \leftarrow x + My^{(i)}
\]

(32)

These updates are known to converge for any choice of initial values for \(y^{(0)}\), as long as \(M\) has a spectral radius \(\rho(M) < 1\). The same convergence guarantees carry over to Eq. 8. We thus know that the update equation Eq. 8 converges if and only if the spectral radius of the matrix \(\hat{P}^T - \hat{P}''\) is smaller than 1.

A.6. Proposition 15: LinBP with edge types

Proof Proposition 15. We are going to derive Eq. 18 from Eq. 29. For convenience, we repeat here both equations:

\[
\hat{B}_k = \hat{X}_k + \sum_{P_{ij} \in \mathcal{P}^t \times k} (A_{ji}^T \hat{P}_j B_{ik} \hat{P}_i - D_{ij} \hat{P}_j \hat{B}_k \hat{P}_{ik})
\]

\[
\hat{b}_s = \hat{x}_s + \sum_{u \in N(s)} \frac{\hat{c}'_{us}}{k_u} + \sum_{u \in N(s)} \hat{P}_{us} (\hat{b}_u - \hat{P}_{us} \hat{b}_s)
\]

(33)

We need to show that any vector \(\hat{b}_s^T\) for a node \(s\) with \(k\) classes is equivalent to the \(o_s\)-th row of \(\hat{B}_k\), for which we are going to write \(\hat{B}_k(o_s, :)\) from now on (recall that \(o_s\) is the order of node \(s\) within \(N_k\)). We show the equivalence for each of the 4 terms separately:

1. \(\hat{x}_s^T = \hat{X}_k(o_s, :)\) by construction.
The Linearization of Pairwise Markov Networks

(2) For the following, recall that $\hat{c}'_{us} = \hat{P}'_{us} 1_{k_u}$:

$$
\left( \sum_{u \in N(s)} \frac{1}{k_u} \hat{c}'_{us} \right)^{T} = \sum_{u \in N(s)} \frac{1}{k_u} \hat{c}'_{us} = \sum_{u \in N(s)} \frac{1}{k_u} A_{P,us}^{T} (o_s, o_u) [1] 1_{k_u} \hat{P}'_{us} = \sum_{u \in N(s)} \frac{1}{k_u} A_{P,us}^{T} (o_s, o_u) [1] 1_{n_c} \hat{P}' = \sum_{P' \in P'^{T \times k}} \left( \sum_{u \in N(s)} \frac{1}{k_u} A_{P,us}^{T} (o_s, o_u) [1] 1_{n_c} \hat{P}' \right)
$$

with $\hat{C}'_{k_s}(o_s, :)$. We used the transpose in order for the later vectorization $\text{vec}$ to create vectors where the different beliefs of a node are adjacent (otherwise $\text{vec}(B)$ results in a vector where all beliefs in the same class from different nodes are adjacent). We next use Roth’s column lemma (Henderson & Searle, 1981; Roth, 1934) that states that

$$
\text{vec}(XYZ) = (Z^{T} \otimes X) \text{vec}(Y)
$$

to rewrite this equation as

$$
\hat{b} = \hat{x} + \hat{c}' + \left( \sum_{P' \in P'} (A_{P'}^{T} \otimes \hat{P}' - D_{P'} \otimes \hat{P}'^{T}) \right) \hat{b}
$$

for $\hat{b} = \text{vec}(\hat{B}^{T})$, $\hat{x} = \text{vec}(\hat{X}^{T})$, and $\hat{c}' = \text{vec}(\hat{C}'_{k_s})$. Using the substitution

$$
M = \sum_{P' \in P'} \left( A_{P'}^{T} \otimes \hat{P}' - D_{P'} \otimes \hat{P}'^{T} \right)
$$

and reformulating the equation leads to the closed-form solution:

$$
\hat{b} = (I_{nk} - M)^{-1} (\hat{x} + \hat{c}')
$$

which is defined if the spectral radius $\rho$ of $M$ is smaller than 1. \hfill \Box

A.8. LinBP for one symmetric, doubly stochastic potential

Proof Proposition 19. First, notice that for any symmetric potential $P \in \mathbb{R}^{k \times k}$, $\hat{P}' = \hat{P}'' = \hat{P}/k$, and hence $A_{P'}^{T} BP' + A_{P'}^{T} BP'' = (A_{P'}^{T} + A_{P'}^{T}) BP'$. Thus, its adjacency matrix becomes symmetric. Since we only have one potential, we also have only one adjacency matrix $A$. Furthermore, $\hat{P}'^{T}$ and hence, $P_s = \hat{P}'^{T}$. Second, the constant term $\hat{C}'_{k_s}$ disappears for doubly stochastic potentials. This follows from the proof of Lemma 13 and the fact that in any doubly stochastic matrix $P \in \mathbb{R}^{k \times k}$, each column is centered around $\frac{1}{k}$.

This allows to simplify Eq.19 to

$$
\hat{B} = \hat{X} + AB\hat{P}' - DB\hat{P}'^{T}
$$

Similarly, applying above assumptions to our closed-form solution Eq.20 leads to:

$$
\hat{b} = (I_{nk} - A \otimes \hat{P}' + D \otimes \hat{P}'^{T})^{-1} \hat{x}
$$

Notice that Eq.34 and Eq.22 are exactly the ones given by our original derivation, except for our slightly different notation. In particular, we originally already centered the
potential \( P \) around \( 1/k \) (or row-recentering and column-recentering in this paper is more general form of our original centering that is necessary to deal with the more general case of a non-doubly stochastic potentials). We also chose here to formulate Eq.22 as \( b = \text{vec}(\hat{B}^T) \) instead of \( \text{vec}(B) \) to keep the beliefs of same nodes adjacent in the resulting stacked vectors. Vectorizing instead the transpose, we get the exact original formulation:

\[
\text{vec}(\hat{B}) = (I_{nk} - \hat{P}' \otimes A + \hat{P}'^2 \otimes D)^{-1} \text{vec}(\hat{X}) \quad (36)
\]

\[ \Box \]

### B. Weighted edges with repeated potentials

Here we derive what it means for a Markov network to have “weights” on edges, and how we can modify our LinBP to handle such weighted edges.

Intuitively, an edge between nodes \( s \) and \( t \) with potential \( P_{st} \) and weight \( w = 2 \) should behave identically as two unweighted parallel edges and each with the same potential. From the original BP update equations Eq.1 and Eq.2, we see that both edges carry the same message as before, and that these two messages need to be multiplied to calculate the resulting messages and beliefs. We can see that this operation is identical to having one single unweighted edge with a new potential \( P_{st} \odot P_{st} \), i.e. with element-wise exponentiation of the entries of the potential. More generally, an edge with a potential \( P \) and weight \( w \) is the same as an unweighted edge with a new potential \( \hat{P} \otimes \hat{P} \) with entries \( \hat{P}_{w}(j,i) = P(j,i)^w \).

To see how weights affect our linearized formulation in terms of residuals, recall that \( P(j,i) = 1 + \hat{P}(j,i) \). Therefore, \( \hat{P}(j,i)^w = (1 + \hat{P}(j,i))w = 1 + wP(j,i) + O(\hat{P}(j,i)^2) \). Under the assumption of small deviations from the center, we thus get: \( P_{w}(j,i) = wP(j,i) \). Hence, weights on edges simply multiply the residual potentials in our linearized formulation. In other words, weights on an edge simply scale the coupling strengths between two neighbors.

It follows that Proposition 15 can be generalized to weighted networks by using weighted adjacency matrices \( A_{\hat{P}} \) with elements \( A_{\hat{P}}(o_s, o_t) = w > 0 \) if the edge \( s \rightarrow t \) with potential \( \hat{P} \) and weight \( w \) exists, and \( A_{\hat{P}}(o_s, o_t) = 0 \) otherwise. In addition, each entry \( D_{\hat{P}}(o_t, o_s) \) in the block-diagonal matrix \( D_{\hat{P}} \) is now the sum of the squared weights of edges to neighbors that are connected to \( t \) via edge potential \( \hat{P} \), instead of just the number of neighbors (recall that the echo cancellation goes back and forth, and notice again that the potentials work along the direction \( s \rightarrow t \)). After this modification, Proposition 15 can immediately be used for weighted graphs as well.

**Example 20.** We give here a small detailed example that shows the effects of weights for a potential whose entries are not really close to each other (i.e. the average entry is 1, however entries can diverge considerably from 1). We start with the potential \( P = \frac{1}{6} \begin{bmatrix} 1 & 4 & 6 & 5 \end{bmatrix} \). By dividing all entries by 6, we get an equivalent potential that is centered around 1; and from this we get the residual and the row-recentered residual matrices:

\[
P = \frac{1}{6} \begin{bmatrix} 1 & 4 & 6 & 5 \end{bmatrix}, \quad \hat{P} = \frac{1}{6} \begin{bmatrix} -2 & 0 & -1 \end{bmatrix}, \quad \hat{P}' = \frac{1}{18} \begin{bmatrix} 1 & -1 & 0 \end{bmatrix}
\]

The squared potential centered around 1 is then: \( P_2 = \frac{3}{113} \begin{bmatrix} 4 & 2 & 2 & 6 & 6 & 8 & 7 \end{bmatrix} \). And the residual and row-recentered residual matrices:

\[
\hat{P}_2 \approx \begin{bmatrix} 0.575 & 0.044 & -0.699 & -0.336 \end{bmatrix}, \quad \hat{P}_2' \approx \begin{bmatrix} 0.085 & -0.091 & 0.006 \end{bmatrix}
\]

We can now compare the potential we get by multiplying the residual by 2, or by squaring the original potential and then recentering:

\[
2\hat{P} \approx \begin{bmatrix} 0.111 & -0.111 \end{bmatrix}, \quad \hat{P}_2' \approx \begin{bmatrix} 0.085 & -0.091 & 0.006 \end{bmatrix}
\]

We see that the overall direction is correct, but there are considerable differences (e.g., \( \approx 30\% \) relative difference for the first matrix entry: 0.111 vs. 0.085).

We next bring each entry in the potential closer to the center. Concretely, we reduce the deviation by one order of magnitude:

\[
P = \frac{1}{6} \begin{bmatrix} 1 & 5.8 & 6.0 & 5.9 \end{bmatrix}, \quad \hat{P} = \frac{1}{60} \begin{bmatrix} -2 & 0 & -1 \end{bmatrix}, \quad \hat{P}' = \frac{1}{180} \begin{bmatrix} 1 & -1 & 0 \end{bmatrix}
\]

Now both versions are very close (e.g., \( \approx 2\% \) relative difference for the first matrix entry: 0.0111 vs. 0.0109):

\[
2\hat{P} \approx \begin{bmatrix} 0.0111 & -0.0111 \end{bmatrix}, \quad \hat{P}_2' \approx \begin{bmatrix} 0.0109 & -0.0110 & 0.00005 \end{bmatrix}
\]