Linearized and Single-Pass Belief Propagation

Wolfgang Gatterbauer
Carnegie Mellon University
gatt@cmu.edu

Stephan Günnemann
Carnegie Mellon University
sguennem@cs.cmu.edu

Danai Koutra
Carnegie Mellon University
dkoutra@cs.cmu.edu

Christos Faloutsos
Carnegie Mellon University
christos@cs.cmu.edu

ABSTRACT

How can we tell when accounts are fake or real in a social network? And how can we tell which accounts belong to liberal, conservative or centrist users? Often, we can answer such questions and label nodes in a network based on the labels of their neighbors and appropriate assumptions of homophily ("birds of a feather flock together") or heterophily ("opposites attract"). One of the most widely used methods for this kind of inference is *Belief Propagation (BP)* which iteratively propagates the information from a few nodes with explicit labels throughout a network until convergence. A well-known problem with BP, however, is that there are no known exact guarantees of convergence in graphs with loops.

This paper introduces *Linearized Belief Propagation* (LinBP), a linearization of BP that allows a closed-form solution via intuitive matrix equations and, thus, comes with exact convergence guarantees. It handles homophily, heterophily, and more general cases that arise in multi-class settings. Plus, it allows a compact implementation in SQL. The paper also introduces *Single-pass Belief Propagation* (SBP), a localized (or "myopic") version of LinBP that propagates information across every edge at most once and for which the final class assignments depend only on the nearest labeled neighbors. In addition, SBP allows fast incremental updates in dynamic networks. Our runtime experiments show that LinBP and SBP are orders of magnitude faster than standard BP, while leading to almost identical node labels.

1. INTRODUCTION

Network effects are powerful and often appear in terms of *homophily* ("birds of a feather flock together"). For example, if we know the political leanings of most of Alice's friends on Facebook, then we have a good estimate of her leaning as well. Occasionally, the reverse is true, also called *heterophily* ("opposites attract"). For example, in an online dating site, we may observe that talkative people prefer to date silent ones, and vice versa. Thus, knowing the labels of a few nodes in a network, plus knowing whether homophily or heterophily applies in a given scenario, we can usually give good predictions about the labels of the remaining nodes.

Proceedings of the VLDB Endowment, Vol. 8, No. 5

	D	R
D	0.8	0.2
R	0.2	0.8

(a) homophily

	T	S
T	0.3	0.7
S	0.7	0.3

(b) heterophily

	H	A	F
H	0.6	0.3	0.1
A	0.3	0.0	0.7
F	0.1	0.7	0.2

(c) general case

Figure 1: Three types of *network effects* with example coupling matrices. Shading intensity corresponds to the affinities or coupling strengths between classes of neighboring nodes. (a): D: Democrats, R: Republicans. (b): T: Talkative, S: Silent. (c): H: Honest, A: Accomplice, F: Fraudster.

In this work, we not only cover these two popular cases with k=2 classes, but also capture more general settings that mix homophily and heterophily. We illustrate with an example taken from online auction settings like e-bay [38]: Here, we observe k=3 classes of people: fraudsters (F), accomplices (A) and honest people (H). Honest people buy and sell from other honest people, as well as accomplices; accomplices establish a good reputation (thanks to multiple interactions with honest people), they never interact with other accomplices (waste of effort and money), but they do interact with fraudsters, forming near-bipartite cores between the two classes. Fraudsters primarily interact with accomplices (to build reputation); the interaction with honest people (to defraud them) happens in the last few days before the fraudster's account is shut down.

Thus, in general, we can have k different classes, and $k \times k$ affinities or coupling strengths between them. These affinities can be organized in a coupling matrix (which we call *heterophily matrix*[1]), as shown in Fig. 1 for our three examples. Figure 1a shows the matrix for homophily: It captures that a connection between people with similar political orientations is more likely than between people with different orientations.[2] Figure 1b captures our example for heterophily: Class T is more likely to date members of class S, and vice versa. Finally, Fig. 1c shows our more general example: We see homophily between members of class H and heterophily between members of classes A and F.

In all of the above scenarios, we are interested in the most likely "beliefs" (or labels) for all nodes in the graph. The

[1]In this paper, we assume the heterophily matrix to be given; e.g., by domain experts. Learning heterophily from existing partially labeled data is interesting future work (see [11] for initial results).
[2]An example of homophily with k=4 classes would be co-authorship: Researchers in computer science, physics, chemistry and biology, tend to publish with co-authors of similar training. Efficient labeling in case of homophily is possible; e.g., by simple relational learners [31].

underlying problem is then: *How can we assign class labels when we know who-contacts-whom and the apriori ("explicit") labels for some of the nodes in the network*? This learning scenario, where we reason from observed training cases directly to test cases, is also called *transductive inference*, or semi-supervised learning (SSL).[3]

One of the most widely used methods for this kind of transductive inference in networked data is *Belief Propagation* (BP) [44], which has been successfully applied in scenarios, such as fraud detection [32, 38] and malware detection [5]. BP propagates the information from a few explicitly labeled nodes throughout the network by iteratively propagating information between neighboring nodes. However, BP has well-known convergence problems in graphs with loops (see [44] for a detailed discussion from a practitioner's point of view). While there is a lot of work on convergence of BP (e.g., [8, 34]), *exact* criteria for convergence are not known [35, Sec. 22]. In addition, whenever we get additional explicit labels (e.g., we identify more fraudsters in the online auction setting), we need to re-run BP from scratch. These issues raise fundamental theoretical questions of practical importance: *How can we find sufficient and necessary conditions for convergence of the algorithm*? *And how can we support fast incremental updates for dynamic networks*?

Contributions. This paper introduces two new formulations of BP. Unlike standard BP, these (*i*) come with exact convergence guarantees, (*ii*) allow closed-form solutions, (*iii*) give a clear intuition about the algorithms, (*iv*) can be implemented on top of standard SQL, and (*v*) one can even be updated incrementally. In more detail, we introduce:

(1) LinBP: Section 3 gives a new matrix formulation for multi-class BP called *Linearized Belief Propagation* (LinBP). Section 4 proves LinBP to be the result of applying a certain linearization process to the update equations of BP. Section 4.2 goes one step further and shows that the solution to LinBP can be obtained in *closed-form* by the inversion of an appropriate Kronecker product. Section 5.1 shows that this new closed-form provides us with *exact convergence guarantees* (even on graphs with loops) and a clear intuition about the reasons for convergence/non-convergence. Section 5.3 shows that our linearized matrix formulation of LinBP allows a compact implementation in SQL with standard joins and aggregates, plus iteration. Finally, experiments in Sect. 7 show that a main-memory implementation of LinBP takes 4 sec for a graph for which standard BP takes 40 min, while giving almost identical classifications ($> 99.9\%$ overlap).

(2) SBP: Section 6 gives a novel semantics for "local" (or "myopic") transductive reasoning called *Single-pass Belief Propagation* (SBP). SBP propagates information across every edge *at most once* (i.e. it ignores some edges) and is a generalization of relational learners [31] from homophily to heterophily and even more general couplings between classes in a sound and intuitive way. In particular, the final labels depend only on the nearest neighbors with explicit labels. The intuition is simple: If we do not know the political leanings of Alice's friends, than knowing the political leaning of *friends of Alice's friends* (i.e. nodes that are 2 hops away in the underlying network) will help us make some predictions about her. However, if we do know about most of her friends, then information that is more distant in the network can often be safely ignored. We formally show the connection between LinBP and SBP by proving that the labeling assignments for both are identical in the case of decreasing affinities between nodes in a graph. Importantly, SBP (in contrast to standard BP and LinBP) allows fast *incremental maintenance* for the predicated labels if the underlying network is dynamic: Our SQL implementation of SBP allows incremental updates with an intuitive index based on shortest paths to explicitly labeled nodes. Finally, experiments in Sect. 7 show that a disk-bound implementation of SBP is even faster than LinBP by one order of magnitude while giving similar classifications ($> 98.6\%$ overlap).

Outline. Sect. 2 provides necessary background on BP. Sect. 3 introduces the LinBP matrix formulation. Sect. 4 sketches its derivation. Sect. 5 provides convergence guarantees, extends LinBP to weighted graphs, and gives a SQL implementation of LinBP. Sect. 6 introduces the SBP semantics, including a SQL implementation for incremental maintenance. Sect. 7 gives experiments. Sect. 8 contrasts related work, and Sect. 9 concludes. All proofs plus an additional algorithm for incrementally updating SBP when adding edges to the graph are available in our technical report on ArXiv [13]. The actual SQL implementations of LinBP and SBP are available on the authors' home pages.

2. BELIEF PROPAGATION

Belief Propagation (BP), also called the sum-product algorithm, is an exact inference method for graphical models with tree structure [40]. The idea behind BP is that all nodes receive messages from their neighbors in parallel, then they update their belief states, and finally they send new messages back out to their neighbors. In other words, at iteration i of the algorithm, the posterior belief of a node s is conditioned on the evidence that is i steps away from s in the underlying network. This process repeats until convergence and is well-understood on trees.

When applied to loopy graphs, however, BP is not guaranteed to converge to the marginal probability distribution. Indeed, Judea Pearl, who invented BP, cautioned about the indiscriminate use of BP in loopy networks, but advocated to use it as an approximation scheme [40]. More important, loopy BP is not even guaranteed to converge at all. Despite this lack of exact criteria for convergence, many papers have since shown that "loopy BP" gives very accurate results *in practice* [49], and it is thus widely used today in various applications, such as error-correcting codes [29] or stereo imaging in computer vision [9]. Our practical interest in BP comes from the fact that it is not just an efficient inference algorithm on probabilistic graphical models, but it has also been successfully used for *transductive inference*.

The transductive inference problem appears, in its generality, in a number of scenarios in both the database and machine learning communities and can be defined as follows: Consider a set of keys $X = \{x_1, \ldots, x_n\}$, a domain of values $Y = \{y_1, \ldots, y_k\}$, a partial labeling function $\ell : X_L \to Y$ with $X_L \subseteq X$ that maps a subset of the keys to values, a weighted mapping $w : (X_1, X_2) \to \mathbb{R}$ with $(X_1, X_2) \subseteq X \times X$, and a local condition $f_i(X, w, x_i, \ell_i)$ that needs to hold for a solution to be accepted.[4] The three problems are then to find: (*i*) an appropriate semantics that determines

[3]Contrast this with *inductive learning*, where we first infer *general rules* from training cases, and only then apply them to new test cases.

[4]Notice that update equations define a local condition implicitly by giving conditions that a solution needs to fulfill after convergence.

labels for all keys, (*ii*) an efficient algorithm that implements this semantics, and (*iii*) efficient ways to update labels in case the labeling function ℓ, or the mapping w change.

In our scenario, we are interested in the most likely beliefs (or classes) for all nodes in a network. BP helps to iteratively propagate the information from a few nodes with explicit beliefs throughout the network. More formally, consider a graph of n nodes (or keys) and k possible classes (or values). Each node maintains a k-dimensional *belief vector* where each element i represents a weight proportional to the belief that this node belongs to class i. We denote by $\mathbf{e}_s$ the vector of prior (or *explicit*) beliefs and $\mathbf{b}_s$ the vector of posterior (or *implicit* or *final*) beliefs at node s, and require that $\mathbf{e}_s$ and $\mathbf{b}_s$ are normalized to 1; i.e. $\sum_i e_s(i) = \sum_i b_s(i) = 1$.[5] Using $\mathbf{m}_{st}$ for the k-dimensional *message* that node s sends to node t, we can write the BP update formulas [35, 48] for the belief vector of each node and the messages it sends w.r.t. class i as:

$$b_s(i) \leftarrow \frac{1}{Z_s} e_s(i) \prod_{u \in N(s)} m_{us}(i) \tag{1}$$

$$m_{st}(i) \leftarrow \sum_j H_{st}(j,i)\; e_s(j) \prod_{u \in N(s)\setminus t} m_{us}(j) \tag{2}$$

Here, we write Z_s for a normalizer that makes the elements of $\mathbf{b}_s$ sum up to 1, and $H_{st}(j,i)$ for a proportional "coupling weight" that indicates the relative influence of class j of node s on class i of node t (cf. Fig. 1).[6] We assume that the relative coupling between classes is the same in the whole graph; i.e. $H(j,i)$ is identical for all edges in the graph. We further require this coupling matrix $\mathbf{H}$ to be *doubly stochastic and symmetric*: (*i*) Double stochasticity is a necessary requirement for our mathematical derivation.[7] (*ii*) Symmetry is not required but follows from our assumption of undirected edges. For BP, the above update formulas are then repeatedly computed for each node until the values (hopefully) converge to the final beliefs.

The goal in our paper is to find the *top beliefs* for each node in the network, and to assign these beliefs to the respective nodes. That is, for each node s, we are interested in determining the classes with the highest values in $\mathbf{b}_s$.

Problem 1 (Top belief assignment)**.** *Given: (1) an undirected graph with n nodes and adjacency matrix $\mathbf{A}$, where $A(s,t) \neq 0$ if the edge $s-t$ exists, (2) a symmetric, doubly stochastic coupling matrix $\mathbf{H}$ representing k classes, where $H(j,i)$ indicates the relative influence of class j of a node on class i of its neighbor, and (3) a matrix of explicit beliefs $\mathbf{E}$, where $E(s,i) \neq 0$ is the strength of belief in class i by node s. The goal of* top belief assignment *is to infer for* each *node a set of classes with highest final belief.*

In other words, our problem is to find a mapping from nodes to *sets of classes* (in order to allow for ties).

3. LINEARIZED BELIEF PROPAGATION

[5]Notice that here and in the rest of this paper, we write $\sum_i$ as short form for $\sum_{i\in[k]}$ whenever k is clear from the context.

[6]We chose the symbol H for the coupling weights as reminder of our motivating concepts of homophily and heterophily. Concretely, if $H(i,i) > H(j,i)$ for $j \neq i$, we say homophily is present, otherwise heterophily or a mix between the two.

[7]Notice that single-stochasticity could easily be constructed by taking any set of vectors of relative coupling strengths between neighboring classes, normalizing them to 1, and arranging them in a matrix.

	Formula	Maclaurin series	Approx.
Logarithm	$\ln(1+\epsilon)$	$= \epsilon - \frac{\epsilon^2}{2} + \frac{\epsilon^3}{3} - \ldots$	$\approx \epsilon$
Division	$\frac{\frac{1}{k}+\epsilon_1}{1+\epsilon_2}$	$= (\frac{1}{k}+\epsilon_1)(1-\epsilon_2+\epsilon_2^2-\ldots)$	$\approx \frac{1}{k}+\epsilon_1-\frac{\epsilon_2}{k}$

Figure 2: Two linearizing approximations used in our derivation.

In this section, we introduce *Linearized Belief Propagation (LinBP)*, which is a closed-form description for the final beliefs after convergence of BP under mild restrictions of our parameters. The main idea is to *center* values around default values (using Maclaurin series expansions) and to then restrict our 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 later give exact convergence criteria based on problem parameters.

Definition 2 (Centering)**.** *We call a vector or matrix $\mathbf{x}$ "*centered around *c" if all its entries are close to c and their average is exactly c.*

Definition 3 (Residual vector/matrix)**.** *If a vector $\mathbf{x}$ is centered around c, then the residual vector around c is defined as $\hat{\mathbf{x}} = [x_1 - c, x_2 - c, \ldots]$. Accordingly, we denote a matrix $\hat{\mathbf{X}}$ as a residual matrix if each column and row vector corresponds to a residual vector.*

For example, we call the vector $\mathbf{x} = [1.01, 1.02, 0.97]$ centered around $c = 1$.[8] The residuals from c will form the *residual vector* $\hat{\mathbf{x}} = [0.01, 0.02, -0.03]$. Notice that the entries in a residual vector always sum up to 0, by construction.

The main ideas in our proofs are as follows: (1) the k-dimensional message vectors $\mathbf{m}$ are centered around 1; (2) all the other k-dimensional vectors are probability vectors, they have to sum up to 1, and thus they are centered around $1/k$. This holds for the belief vectors $\mathbf{b}$, $\mathbf{e}$, and for the all entries of matrix $\mathbf{H}$; and (3) we make use of each of the two linearizing approximations shown in Fig. 2 exactly once.

According to aspect (1) of the previous paragraph, we require that the messages sent are normalized so that the average value of the elements of a message vector is 1 or, equivalently, that the elements sum up to k:

$$m_{st}(i) \leftarrow \frac{1}{Z_{st}} \sum_j H(j,i)\, e_s(j) \prod_{u \in N(s)\setminus t} m_{us}(j) \tag{3}$$

Here, we write Z_{st} as a normalizer that makes the elements of $\mathbf{m}_{st}$ sum up to k. Scaling all elements of a message vector by the same constant does *not* affect the resulting beliefs since the normalizer in Eq. 1 makes sure that the beliefs are always normalized to 1, independent of the scaling of the messages. Thus, scaling messages still preserves the exact solution, yet it will be essential for our derivation.

Theorem 4 (Linearized BP (LinBP))**.** *Let $\hat{\mathbf{B}}$ and $\hat{\mathbf{E}}$ be the residual matrices of final and explicit beliefs centered around $1/k$, $\hat{\mathbf{H}}$ the residual coupling matrix centered around $1/k$, $\mathbf{A}$ the adjacency matrix, and $\mathbf{D} = diag(\mathbf{d})$ the diagonal degree matrix. Then, the final belief assignment from belief propagation is approximated by the equation system:*

$$\boxed{\hat{\mathbf{B}} = \hat{\mathbf{E}} + \mathbf{A}\hat{\mathbf{B}}\hat{\mathbf{H}} - \mathbf{D}\hat{\mathbf{B}}\hat{\mathbf{H}}^2} \qquad (LinBP) \tag{4}$$

[8]All vectors $\mathbf{x}$ in this paper are assumed to be *column vectors* $[x_1, x_2, \ldots]^\intercal$ even if written as row vectors $[x_1, x_2, \ldots]$.

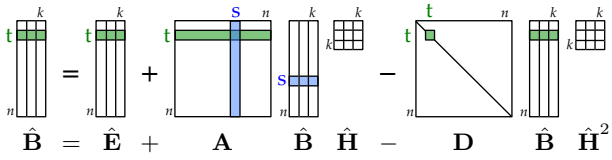


Figure 3: LinBP equation (Eq. 4): Notice our matrix conventions: $\hat{H}(j,i)$ indicates the relative influence of class j of a node on class i of its neighbor, $A(s,t) = A(t,s) \neq 0$ if the edge $s-t$ exists, and $\hat{B}(s,i)$ is the belief in class i by node s.

Figure 3 illustrates Eq. 4 and shows our matrix conventions. We refer to the term $\mathbf{D}\hat{\mathbf{B}}\hat{\mathbf{H}}^2$ as "echo cancellation".[9] For increasingly small residuals, the echo cancellation becomes increasingly negligible, and by further ignoring it, Eq. 4 can be further simplified to

$$\boxed{\hat{\mathbf{B}} = \hat{\mathbf{E}} + \mathbf{A}\hat{\mathbf{B}}\hat{\mathbf{H}}} \qquad (LinBP^*) \quad (5)$$

We will refer to Eq. 4 (with echo cancellation) as LinBP and Eq. 5 (without echo cancellation) as LinBP*.

Iterative updates. Notice that while these equations give an implicit definition of the final beliefs after convergence, they can also be used as iterative update equations, allowing an iterative calculation of the final beliefs. Starting with an arbitrary initialization of $\hat{\mathbf{B}}$ (e.g., all values zero), we repeatedly compute the right hand side of the equations and update the values of $\hat{\mathbf{B}}$ until the process converges:

$$\hat{\mathbf{B}}^{(\ell+1)} \leftarrow \hat{\mathbf{E}} + \mathbf{A}\hat{\mathbf{B}}^{(\ell)}\hat{\mathbf{H}} - \mathbf{D}\hat{\mathbf{B}}^{(\ell)}\hat{\mathbf{H}}^2 \qquad (LinBP) \quad (6)$$

$$\hat{\mathbf{B}}^{(\ell+1)} \leftarrow \hat{\mathbf{E}} + \mathbf{A}\hat{\mathbf{B}}^{(\ell)}\hat{\mathbf{H}} \qquad (LinBP^*) \quad (7)$$

Thus, 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, as explained in Sect. 5.1. Also notice that our update equations calculate beliefs directly (i.e. without having to calculate messages first); this will give us significant performance improvements as our experiments will later show.

4. DERIVATION OF LINBP

This section sketches the proofs of our first technical contribution: Section 4.1 linearizes the update equations of BP by centering around appropriate defaults and using the approximations from Fig. 2 (Lemma 5), and then expressesing the steady state messages in terms of beliefs (Lemma 6). Sect. 4.2 gives an additional closed-form expression for the steady-state beliefs (Proposition 7).

4.1 Centering Belief Propagation

We derive our formalism by centering the elements of the coupling matrix and all message and belief vectors around their natural default values; i.e. the elements of $\mathbf{m}$ around 1, and the elements of $\mathbf{H}$, $\mathbf{e}$, and $\mathbf{b}$ around $\frac{1}{k}$. We are interested in the residual values given by: $m(i) = 1 + \hat{m}(i)$, $H(j,i) = \frac{1}{k} + \hat{H}(j,i)$, $e(i) = \frac{1}{k} + \hat{e}(i)$, and $b(i) = \frac{1}{k} + \hat{b}(i)$.[10] As a consequence, $\hat{\mathbf{H}} \in \mathbb{R}^{k\times k}$ is the *residual coupling matrix* that makes explicit the relative attraction and repulsion: The sign of $\hat{H}(j,i)$ tells us if the class j attracts or repels class i in a neighbor, and the magnitude of $\hat{H}(j,i)$ indicates the strength. Subsequently, this centering allows us to rewrite belief propagation in terms of the residuals.

Lemma 5 (Centered BP)**.** *By centering the coupling matrix, beliefs and messages, the equations for belief propagation can be approximated by:*

$$\hat{b}_s(i) \leftarrow \hat{e}_s(i) + \frac{1}{k}\sum_{u\in N(s)} \hat{m}_{us}(i) \quad (8)$$

$$\hat{m}_{st}(i) \leftarrow k\sum_j \hat{H}(j,i)\hat{b}_s(j) - \sum_j \hat{H}(j,i)\hat{m}_{ts}(j) \quad (9)$$

Using Lemma 5, we can derive a closed-form description of the steady-state of belief propagation.

Lemma 6 (Steady state messages)**.** *For small deltas of all values from their expected values, and after convergence of belief propagation, message propagation can be expressed in terms of the steady beliefs as:*

$$\hat{\mathbf{m}}_{st} = k(\mathbf{I}_k - \hat{\mathbf{H}}^2)^{-1}\hat{\mathbf{H}}(\hat{\mathbf{b}}_s - \hat{\mathbf{H}}\hat{\mathbf{b}}_t) \quad (10)$$

where $\mathbf{I}_k$ is the identity matrix of size k.

From Lemma 6, we can finally prove Theorem 4.

4.2 Closed-form solution for LinBP

In practice, we will solve Eq. 4 and Eq. 5 via an iterative computation (see end of Sect. 3). However, we next give a *closed-form* solution, which allows us later to study the convergence of the iterative updates. We need to introduce two new notions: Let $\mathbf{X}$ and $\mathbf{Y}$ be matrices of order $m \times n$ and $p \times q$, respectively, and let $\mathbf{x}_j$ denote the j-th column of matrix $\mathbf{X}$; i.e. $\mathbf{X} = \{x_{ij}\} = [\mathbf{x}_1 \dots \mathbf{x}_n]$. First, the *vectorization* of matrix $\mathbf{X}$ stacks the columns of a matrix one underneath the other to form a single column vector; i.e.

$$\mathtt{vec}(\mathbf{X}) = \begin{bmatrix}\mathbf{x}_1\\ \vdots \\ \mathbf{x}_n\end{bmatrix}$$

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

$$\mathbf{X}\otimes\mathbf{Y} = \begin{bmatrix} x_{11}\mathbf{Y} & x_{12}\mathbf{Y} & \dots & x_{1n}\mathbf{Y}\\ x_{21}\mathbf{Y} & x_{22}\mathbf{Y} & \dots & x_{2n}\mathbf{Y}\\ \vdots & \vdots & \ddots & \vdots \\ x_{m1}\mathbf{Y} & x_{m2}\mathbf{Y} & \dots & x_{mn}\mathbf{Y}\end{bmatrix}$$

Proposition 7 (Closed-form LinBP)**.** *The closed-form solution for LinBP (Eq. 4) is given by:*

$$\boxed{\mathtt{vec}(\hat{\mathbf{B}}) = (\mathbf{I}_{nk} - \hat{\mathbf{H}}\otimes\mathbf{A} + \hat{\mathbf{H}}^2\otimes\mathbf{D})^{-1}\mathtt{vec}(\hat{\mathbf{E}})} \qquad (LinBP) \quad (11)$$

[9] 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 ("$u \in N(s)\backslash t$" in Eq. 2). In a probabilistic scenario on tree-based graphs, this term is required for correctness. In loopy graphs (without well-justified semantics), this term still compensates for two neighboring nodes building up each other's scores.

[10] Notice that we call these default values "natural" as our results imply that if we start with centered messages around 1 and set $\frac{1}{Z_{st}} = k$, then the derived messages with Eq. 3 remain centered around 1 *for any iteration*. Also notice that multiplying with a message vector with all entries 1 does not change anything. Similarly, a prior belief vector with all entries $\frac{1}{k}$ gives equal weight to each class. Finally, notice that we call "*nodes with explicit beliefs*", those nodes for which the residuals have non-zero elements ($\hat{\mathbf{e}} \neq \mathbf{0}_k$); i.e. the explicit beliefs deviate from the center $\frac{1}{k}$.

By further ignoring the echo cancellation $\hat{\mathbf{H}}^2 \otimes \mathbf{D}$, we get the closed-form for LinBP* (Eq. 5) as:

$$\boxed{\mathtt{vec}(\hat{\mathbf{B}}) = (\mathbf{I}_{nk} - \hat{\mathbf{H}} \otimes \mathbf{A})^{-1}\mathtt{vec}(\hat{\mathbf{E}})} \qquad (LinBP^*) \quad (12)$$

Thus, by using Eq. 11 or Eq. 12, we are able to compute the final beliefs in a closed-form, as long as the inverse of the matrix exists. In the next section, we show the relation of the closed-form to our original update equation Eq. 6 and give exact convergence criteria.

5. ADDITIONAL BENEFITS OF LINBP

In this section, we give *sufficient and necessary* convergence criteria for LinBP and LinBP*, we show how our formalism generalizes to weighted graphs, and we show how our update equations can be implemented in standard SQL.

5.1 Update equations and Convergence

Equation 11 and Eq. 12 give us a closed-form for the final beliefs after convergence. From the Jacobi method for solving linear systems [43], we know that the solution for $\mathbf{y} = (\mathbf{I} - \mathbf{M})^{-1}\mathbf{x}$ can be calculated, under certain conditions, via the iterative update equation

$$\mathbf{y}^{(\ell+1)} \leftarrow \mathbf{x} + \mathbf{M}\,\mathbf{y}^{(\ell)} \qquad (13)$$

These updates are known to converge for any choice of initial values for $\mathbf{y}_{(0)}$, as long as $\mathbf{M}$ has a spectral radius $\rho(\mathbf{M}) < 1$.[11] Thus, the same convergence guarantees carry over when Eq. 11 and Eq. 12 are written, respectively, as

$$\mathtt{vec}(\hat{\mathbf{B}}^{(\ell+1)}) \leftarrow \mathtt{vec}(\hat{\mathbf{E}}) + (\hat{\mathbf{H}} \otimes \mathbf{A} - \hat{\mathbf{H}}^2 \otimes \mathbf{D})\mathtt{vec}(\hat{\mathbf{B}}^{(\ell)}) \qquad (14)$$

$$\mathtt{vec}(\hat{\mathbf{B}}^{(\ell+1)}) \leftarrow \mathtt{vec}(\hat{\mathbf{E}}) + (\hat{\mathbf{H}} \otimes \mathbf{A})\mathtt{vec}(\hat{\mathbf{B}}^{(\ell)}) \qquad (15)$$

Furthermore, it follows from Proposition 7, that update Eq. 14 is equivalent to our original update Eq. 6, and thus both have the same convergence guarantees.

We are now ready to give a sufficient *and* necessary criteria for convergence of the iterative LinBP and LinBP* update equations.

Lemma 8 (Exact convergence)**.** *Necessary and sufficient criteria for convergence of LinBP and LinBP* are:*

$$\textit{LinBP converges} \;\Leftrightarrow \rho\big(\hat{\mathbf{H}} \otimes \mathbf{A} - \hat{\mathbf{H}}^2 \otimes \mathbf{D}\big) < 1 \qquad (16)$$

$$\textit{LinBP}^* \textit{ converges} \Leftrightarrow \rho(\hat{\mathbf{H}}) < \tfrac{1}{\rho(\mathbf{A})} \qquad (17)$$

In practice, computation of the largest eigenvalues can be expensive. Instead, we can exploit the fact that any norm $||\mathbf{X}||$ gives an upper bounds to the spectral radius of a matrix $\mathbf{X}$ to establish sufficient (but not necessary) and easier-to-compute conditions for convergence.

Lemma 9 (Sufficient convergence)**.** *Let $||\cdot||$ stand for any sub-multiplicative norm of the enclosed matrix. Then, the following are sufficient criteria for convergence:*

$$\textit{LinBP converges} \;\Leftarrow ||\hat{\mathbf{H}}|| < \tfrac{\sqrt{||\mathbf{A}||^2 + 4||\mathbf{D}||} - ||\mathbf{A}||}{2||\mathbf{D}||} \qquad (18)$$

$$\textit{LinBP}^* \textit{ converges} \Leftarrow ||\hat{\mathbf{H}}|| < \tfrac{1}{||\mathbf{A}||} \qquad (19)$$

Further, let M be a set of such norms and let $||\mathbf{X}||_M := \min_{||\cdot||_i \in M} ||\mathbf{X}||_i$. Then, by replacing each $||\cdot||$ with $||\cdot||_M$, we get better bounds.

[11] The spectral radius $\rho(\cdot)$ is the supremum among the absolute values of the eigenvalues of the enclosed matrix.

Vector/Elementwise p-norms for $p \in [1, 2]$ (e.g., the Frobenius norm) and all induced p-norms are sub-multiplicative.[12] Furthermore, vector p-norms are monotonically decreasing for increasing p, and thus: $\rho(\mathbf{X}) \leq ||\mathbf{X}||_2 \leq ||\mathbf{X}||_1$. We thus suggest using the following set M of three norms which are all fast to calculate: (i) Frobenius norm, (ii) induced-1 norm, and (iii) induced-∞ norm.

5.2 Weighted graphs

Notice that Theorem 4 can be generalized to allow weighted graphs by simply using a weighted adjacency matrix $\mathbf{A}$ with elements $A(i, j) = w > 0$ if the edge $j - i$ exists with weight w, and $A(i, j) = 0$ otherwise. Our derivation remains the same, we only need to make sure that the degree d_s of a node s is the sum of the squared weights to its neighbors (recall that the echo cancellation goes back and forth). The weight on an edge simply scales the coupling strengths between two neighbors, and we have to add up parallel paths. Thus, Theorem 4 can be applied for *weighted* graphs as well.

5.3 LinBP in SQL

Much of today's data is stored in relational DBMSs. We next give a compact translation of our linearized matrix formulation into a simple implementation in SQL with standard joins and aggregates, plus iteration. As a consequence, any standard DBMS is able to perform LinBP on networked data stored in relations. An implementation of the original BP would require either a non-standard product aggregate function (with the practical side effect of often producing underflows) or the use of an additional logarithmic function. Issues with convergence would still apply [44].

In the following, we use Datalog notation extended with aggregates in the tradition of [7]. Such an aggregate query has the form $Q(\bar{x}, \alpha(\bar{y})) :- C(\bar{z})$ with C being a conjunction of non-negated relational atoms and comparisons, and $\alpha(\bar{y})$ being the aggregate term.[13] When translating into SQL, the head of the query $(\bar{x}, \alpha(\bar{y}))$ defines the SELECT clause, and the variables $\bar{x}$ appear in the GROUP BY clause of the query.

We use table $A(\underline{s, t}, w)$ to represent the adjacency matrix $\mathbf{A}$ with s and t standing for source and target node, respectively, and w for weight; $E(\underline{v, c}, b)$ and $B(\underline{v, c}, b)$ to represent the explicit beliefs $\hat{\mathbf{E}}$ and final beliefs $\hat{\mathbf{B}}$, respectively, with v standing for node, c for class and b for belief; and $H(\underline{c_1, c_2}, h)$ to represent the coupling matrix $\hat{\mathbf{H}}$ with coupling strength h from a class c_1 on it's neighbor's class c_2. From these data, we calculate an additional table $D(\underline{v}, d)$ representing the degree matrix $\mathbf{D}$, defined to allow weighted edges:[14]

$$D(s, sum(w * w)) :- A(s, t, w)$$

and an additional table $H_2(\underline{c_1, c_2}, h)$ representing $\hat{\mathbf{H}}^2$:

$$H_2(c_1, c_2, sum(h_1 \cdot h_2)) :- H(c_1, c_3, h_1), H(c_3, c_2, h_2) \qquad (20)$$

Using these tables, Algorithm 1 shows the translation of the update equations for LinBP into the relational model:

[12] Vector p-norms are defined as $||\mathbf{X}||_p = \big(\sum_i \sum_j |X(i,j)|^p\big)^{1/p}$. Induced p-norms, for $p = 1$ and $p = \infty$, are defined $||\mathbf{X}||_1 = \max_j \sum_i |X(i,j)|$ and $||\mathbf{X}||_\infty = \max_i \sum_j |X(i,j)|$, i.e. as maximum absolute column sum or maximum absolute row sum, respectively.

[13] Notice that in a slight abuse of notation (and for the sake of conciseness), we use variables to express both attribute names and join variables in Datalog notation.

[14] Remember from Sect. 5.2 that the degree of a node in a weighted graph is the sum of the squares of the weights to all neighbors.

Algorithm 1: (LinBP) Returns the final beliefs B with LinBP for a weighted network A with explicit beliefs E, coupling strengths H, and calculated tables D and H_2.

Input: $A(s,t,w), E(v,c,b), H(c_1,c_2,h),\ D(v,d), H_2(c_1,c_2,h)$
Output: $B(v,c,b)$

1 Initialize final beliefs for nodes with explicit beliefs:
$B(s,c,b) :- E(s,c,b)$
2 **for** $i \leftarrow 1$ **to** l **do**
3 Create two temporary views:
$V_1(t,c_2,sum(w \cdot b \cdot h)) :- A(s,t,w), B(s,c_1,b), H(c_1,c_2,h)$
$V_2(s,c_2,sum(d \cdot b \cdot h)) :- D(s,d), B(s,c_1,b), H_2(c_1,c_2,h)$
4 Update final beliefs:
$B(v,c,b_1+b_2-b_3) :- E(v,c,b_1), V_1(v,c,b_2), V_2(v,c,b_3)$
return $B(v,c,b)$

We initialize the final beliefs with the explicit beliefs (line 1). We then create two temporary tables, $V_1(\underline{v,c},b)$ representing the result of $\mathbf{A\hat{B}\hat{H}}$ and $V_2(\underline{v,c},b)$ for $\mathbf{D\hat{B}\hat{H}}^2$ (line 3). These views are then combined with the explicit beliefs to update the final beliefs (line 4).[15] This is repeated a fixed number l of times or until the maximum change of a belief between two iterations is smaller than a threshold. Finally, we return the top beliefs for each node.

Corollary 10 (LinBP in SQL)**.** *The iterative updates for LinBP can be expressed in standard SQL with iteration.*

6. SINGLE-PASS BELIEF PROPAGATION

Our ultimate goal with belief propagation is to assign the most likely class(es) to each unlabeled node (i.e. each node without explicit beliefs). Here, we define a semantics for top belief assignment that is closely related to BP and LinBP (it gives the same classification for increasingly small coupling weights), but that has two algorithmic advantages: (*i*) calculating the final beliefs requires to visit every node only once (and to propagate values across an edge *at most* once); and (*ii*) the beliefs can be maintained incrementally when new explicit beliefs or edges are added to the graph.

6.1 Scaling Beliefs

We start with a simple definition that helps us separate the relative strength of beliefs from their absolute values.

Definition 11 (Standardization)**.** *Given a vector* $\mathbf{x} = [x_1, x_2, \ldots, x_k]$ *with* $\mu(\mathbf{x})$ *and* $\sigma(\mathbf{x})$ *being the mean and the standard deviation of the elements of* $\mathbf{x}$*, respectively. The* standardization *of* $\mathbf{x}$ *is the new vector* $\mathbf{x}' = \zeta(\mathbf{x})$ *with* $x'_i = \frac{x_i - \mu(\mathbf{x})}{\sigma(\mathbf{x})}$ *if* $\sigma \neq 0$*, and with* $x'_i = 0$ *if* $\sigma = 0$.[16]

For example, $\zeta\big([1,0]\big) = [1,-1]$, $\zeta\big([1,1,1]\big) = [0,0,0]$, and $\zeta\big([1,0,0,0,0]\big) = [2,-0.5,-0.5,-0.5,-0.5]$. The *standardized belief assignment* $\hat{\mathbf{b}}'_s$ for a node s is then the standardization of the final belief assignment: $\hat{\mathbf{b}}'_s = \zeta(\hat{\mathbf{b}}_s)$. For example, assume two nodes s and t with final beliefs $\hat{\mathbf{b}}_s = [4,-1,-1,-1,-1]$ and $\hat{\mathbf{b}}_t = [40,-10,-10,-10,-10]$, respectively. The standardized belief assignment is then the same for both nodes: $\hat{\mathbf{b}}'_s = \hat{\mathbf{b}}'_t = [2,-0.5,-0.5,-0.5,-0.5]$ whereas the standard deviations indicate the magnitude of differences: $\sigma(\hat{\mathbf{b}}'_s) = 2$, $\sigma(\hat{\mathbf{b}}'_s) = 20$.

[15] In practice, we use union all, followed by a grouping on v, c.
[16] We use the symbol ζ since standardized vector elements are also varyingly called *standard scores*, z-scores, or z-values.

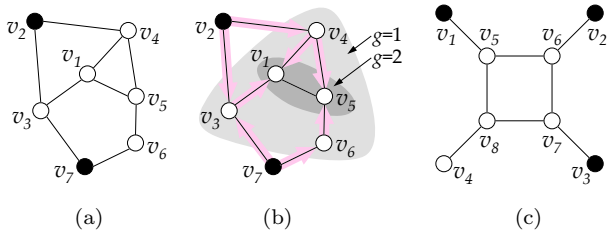


Figure 5: (a),(b): Example 16: Node v_1 has geodesic number 2 and three shortest paths to nodes with explicit beliefs v_2 and v_7. (c): Example 20: Example torus graph taken from [48].

Lemma 12 (Scaling $\hat{\mathbf{E}}$)**.** *Scaling the explicit beliefs with a constant factor* λ *leads to scaled final beliefs by* λ*. In other words,* $\forall\lambda \in \mathbb{R} : \big(\hat{\mathbf{E}} \leftarrow \lambda \cdot \hat{\mathbf{E}}\big) \Rightarrow \big(\hat{\mathbf{B}} \leftarrow \lambda \cdot \hat{\mathbf{B}}\big)$.

Proof. This follows immediately from Eq. 11. □

Corollary 13 (Scaling $\hat{\mathbf{E}}$)**.** *Scaling* $\hat{\mathbf{E}}$ *with a constant factor does not change the standardized belief assignment* $\hat{\mathbf{B}}'$.

The last corollary implies that scaling the explicit beliefs has *no effect* on the top belief assignment, and thus the ultimate classification by LinBP.

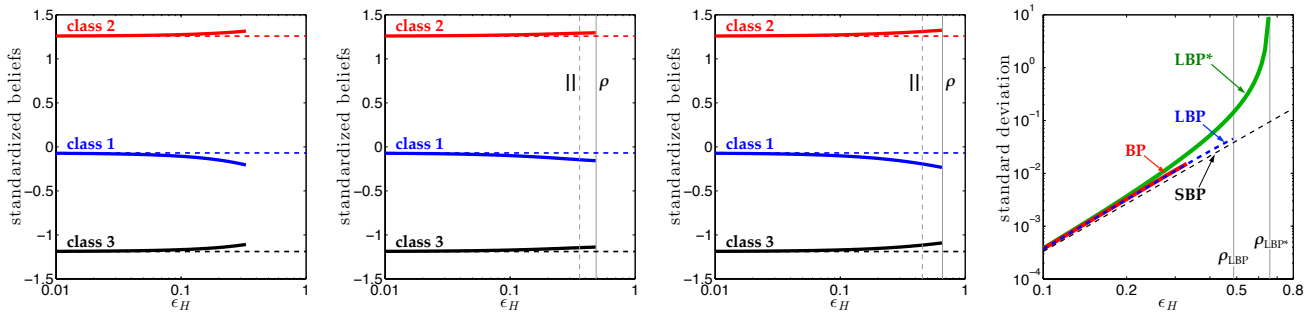
6.2 Scaling Coupling Strengths

While scaling $\hat{\mathbf{E}}$ has no effect on the standardized beliefs, the scale of the residual coupling matrix $\hat{\mathbf{H}}$ is important. To separate (*i*) the *relative difference* among beliefs from (*ii*) their *absolute scale*, we introduce a positive parameter ϵ_H and define with $\hat{\mathbf{H}}_o$ the unscaled ("original") residual coupling matrix implicitly by: $\hat{\mathbf{H}} = \epsilon_H \hat{\mathbf{H}}_o$. This separation allows us to keep the relative scaling fixed as $\hat{\mathbf{H}}_o$ and to thus analyze the influence of the absolute scaling on the standardized belief assignment (and thereby the top belief assignment) by varying ϵ_H only.

It was previously observed in experiments [26] that the top belief assignment is the same for a large range of ϵ_H in belief propagation with binary classes, but that it deviates for very small ϵ_H. Here we show that the standardized belief assignment for LinBP converges for $\epsilon_H \to 0^+$, and that any deviations are due to limited computational precision. We also give a new closed-form for the predictions of LinBP in the limit of $\epsilon_H \to 0^+$ and name this semantics *Single-Pass Belief Propagation (SBP)*. SBP has several advantages: (*i*) it is faster to calculate (we chose its name since information is propagated across each edge at most once), (*ii*) it can be maintained incrementally, and (*iii*) it provides a simple intuition about its behavior and an interesting connection to relational learners [31]. For that, we need one more notion:

Definition 14 (Geodesic number g)**.** *The* geodesic number g_t *of a node* t *is the length of the shortest path to any node with explicit beliefs.*

Notice that any node with explicit beliefs has geodesic number 0. For the following definition, let the weight w of a path p be the product of the weights of its edges (if the graph is unweighted, than the weights are 1).



(a) $\hat{\mathbf{b}}'_{v_4}$ for BP (b) $\hat{\mathbf{b}}'_{v_4}$ for LinBP (c) $\hat{\mathbf{b}}'_{v_4}$ for LinBP* (d) $\sigma(\hat{\mathbf{b}}_{v_4})$

Figure 4: Example 20: (a-c): For decreasing ϵ_H, the standardized beliefs of BP, LinBP, and LinBP* converge towards the ones from SBP: $[-0.069, 1.258, -1.189]$ (horizontal dashed lines). While there are no known *exact* convergence criteria for BP, we gave *necessary and sufficient* criteria for both LinBP and LinBP* (vertical full lines named ρ) plus easier-to-calculate sufficient only conditions (vertical dashed lines named ||). Notice that our ρ-criteria predict exactly when they algorithms stop converging (end of lines). (d): For decreasing ϵ_H, the standard deviations of final beliefs for BP, LinBP, and LinBP* also converge towards the one of SBP.

Definition 15 (Single-Pass BP (SBP))**.** *Given a node t with geodesic number k, let P_t^k be the set of all paths with length k from a node with explicit beliefs to t. For any such path $p \in P_t^k$, let w_p be its weight, and $\hat{\mathbf{e}}_p$ the explicit beliefs of the node at the start of path p. The final belief assignment $\hat{\mathbf{b}}_t$ for Single-pass Belief Propagation (SBP) is defined by*

$$\hat{\mathbf{b}}_s = \hat{\mathbf{H}}^k \sum_{p \in P_s^k} w_p \hat{\mathbf{e}}_p \tag{21}$$

The intuition behind SBP is that nodes with increasing distance have an increasingly negligible influence: For every additional edge in a path, the original influence is scaled by ϵ_H times the modulation by $\hat{\mathbf{H}}$. Thus in the limit of $\epsilon_H \to 0^+$, the nearest neighbors with explicit beliefs will dominate the influence of any other node. Since linear scaling does not change the standardization of a vector, $\zeta(\epsilon\,\mathbf{x}) = \zeta(\mathbf{x})$, *scaling $\hat{\mathbf{H}}$ has no effect on the standardized and thus also top belief assignments* for SBP. In other words, the standardized belief assignment of SBP is independent of ϵ_H (as long as $\epsilon_H > 0$), and w.l.o.g. we can therefore use the unscaled coupling matrix $\hat{\mathbf{H}}_o$ ($\epsilon_H = 1$). This does not hold for LinBP.

Example 16 (SBP illustration)**.** *Consider the undirected and unweighted graph of Fig. 5a. Node v_1 has geodesic number 2 since the closest nodes with explicit beliefs are v_2 and v_7 two hops away. There are three highlighted shortest paths to those beliefs. The SBP standardized belief assignment is then $\hat{\mathbf{b}}'_{v_1} = \zeta\big(\hat{\mathbf{H}}_o^2(2\hat{\mathbf{e}}_{v_2} + \hat{\mathbf{e}}_{v_7})\big)$. Notice that the factor 2 for $\hat{\mathbf{e}}_{v_1}$ arises from the 2 shortest paths from v_2 to v_1.*

Given a graph with adjacency matrix $\mathbf{A}$ and a selection of explicit nodes. Then for any edge, one of two cases is true: (*i*) the edge connects two nodes with the same geodesic number, or (*ii*) the edge connects two nodes that have geodesic numbers of difference one. It follows that SBP has the same semantics as LinBP over a modified graph with some edges removed and the remaining edges becoming directed:

Lemma 17 (Modified adjacency matrix)**.** *Consider a graph with adjacency matrix $\mathbf{A}$ and a selection of explicit nodes. Remove all edges between nodes with same geodesic numbers. For the remaining edges, keep the direction from lower to higher geodesic number. Let $\mathbf{A}_*$ be the resulting modified adjacency matrix. Then: (1) the directed graph $\mathbf{A}_*$ has no directed cycles; and (2) SBP for $\mathbf{A}$ leads to the same final beliefs as LinBP over the transpose $\mathbf{A}_*^\intercal$.*

Example 18 (SBP adjacency matrix)**.** *Let's consider again the undirected graph of Fig. 5b. Among the 4 entries for $v_1 - v_3$ and $v_1 - v_5$ in $\mathbf{A}$, the modified adjacency matrix contains only one entry for $v_3 \to v_1$, because v_3, v_1, v_5 have geodesic numbers 1, 2, 2, respectively. Thus the edge $v_1 - v_3$ only propagates information from v_3 to v_1, and the edge $v_1 - v_5$ propagates no information, as both end points have the same geodesic number.*

$$\mathbf{A} = \begin{bmatrix} 0&0&1&1&0&0&0\\ 0&0&1&1&0&0&0\\ 1&1&0&0&0&0&1\\ 1&1&0&0&1&0&0\\ 0&0&0&1&0&1&0\\ 0&0&0&0&1&0&1\\ 0&0&1&0&0&1&0 \end{bmatrix} \qquad \mathbf{A}_* = \begin{bmatrix} 0&0&0&0&0&0&0\\ 0&0&1&1&0&0&0\\ 0&0&0&0&0&0&0\\ 1&0&0&0&1&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0\\ 0&0&1&0&0&1&0 \end{bmatrix}$$

The following theorem gives the connection between LinBP and SBP and is the main result of this section.

Theorem 19 (Limit of LinBP)**.** *For $\lim_{\epsilon_H \to 0^+}$, the standardized belief assignment for LinBP converges towards the standardized belief assignment for SBP.*

In other words, except for ties (!), the top belief assignment for LinBP and SBP are equal for sufficiently small ϵ_H.

Example 20 (Detailed example)**.** *Consider the unweighted and undirected torus graph shown in Fig. 5c, and assume explicit beliefs $\hat{\mathbf{e}}_{v_1} = [2, -1, -1]$, $\hat{\mathbf{e}}_{v_2} = [-1, 2, -1]$, $\hat{\mathbf{e}}_{v_3} = [-1, -1, 2]$, plus the coupling matrix from Fig. 1c. We get the unscaled residual matrix by centering all entries around $\frac{1}{3}$: $\hat{\mathbf{H}}_o = \left[\begin{smallmatrix} 0.6 & 0.3 & 0.1 \\ 0.3 & 0.0 & 0.7 \\ 0.1 & 0.7 & 0.2 \end{smallmatrix}\right] - \left[\frac{1}{3}\right]_{3\times 3}$. We focus on node v_4 and compare the standardized belief assignment $\hat{\mathbf{b}}'_{v_4}$ and the standard deviation $\sigma(\hat{\mathbf{b}}_{v_4})$ between BP, LinBP, LinBP*, and SBP for $\hat{\mathbf{H}} = \epsilon_H \hat{\mathbf{H}}_o$ and the limit of $\epsilon_H \to 0$. SBP predicts the standardized beliefs to result from the two shortest paths, $v_1 \to v_5 \to v_8 \to v_4$ and $v_3 \to v_7 \to v_8 \to v_4$, and thus $\hat{\mathbf{b}}'_{v_4} = \zeta\big(\hat{\mathbf{H}}_o^3\,(\hat{\mathbf{e}}_{v_1} + \hat{\mathbf{e}}_{v_3})\big) \approx [-0.069, 1.258, -1.189]$. For the standard deviation, we get $\sigma\big(\hat{\mathbf{b}}_{v_4}\big) = \sigma\big(\hat{\mathbf{H}}^3\,(\hat{\mathbf{e}}_{v_1} + \hat{\mathbf{e}}_{v_3})\big) = \epsilon_H^3 \sigma\big(\hat{\mathbf{H}}_o^3\,(\hat{\mathbf{e}}_{v_1} + \hat{\mathbf{e}}_{v_3})\big) \approx \epsilon_H^3 \cdot 0.332$. According to Eq. 16, LinBP converges iff $\rho\big(\epsilon_H \hat{\mathbf{H}}_0 \otimes \mathbf{A} - \epsilon_H^2 \hat{\mathbf{H}}_0^2 \otimes \mathbf{D}\big) < 1$, from which we can calculate numerically $\epsilon_H \lesssim 0.488$. According to Eq. 17, LinBP* converges iff $\epsilon_H < \frac{1}{\rho(\hat{\mathbf{H}}_o)\rho(\mathbf{A})}$ and thus*

Algorithm 2: (SBP) Returns the final beliefs B and geodesic numbers G with SBP for a weighted network A with explicit beliefs E, and coupling scores H.

Input: $A(s,t,w)$, $E(v,c,b)$, $H(c_1,c_2,h)$
Output: $B(v,c,b)$, $G(v,g)$

1 Initialize geodesic n. and beliefs for nodes with explicit beliefs:
$G(v,'0') :- E(v,\_,\_)$
$B(v,c,b) :- E(v,c,b)$
2 $i \leftarrow 1$
3 **repeat**
4 Find next nodes to calculate:
$G(t,i) :- G(s,i-1), A(s,t,\_), \neg G(t,\_)$
5 Calculate beliefs for new nodes:
$B(t,c_2,sum(w \cdot b \cdot h)) :- G(t,i), A(s,t,w), B(s,c_1,b), G(s,i-1), H(c_1,c_2,h)$
6 $i \leftarrow i+1$
7 **until** *no more inserts into* G
8 **return** B *and* G

for $\epsilon_H \lesssim 0.658$*, given* $\rho(\mathbf{A}) \approx 2.414$ *and* $\rho(\hat{\mathbf{H}}_o) \approx 0.629$*. Using the norm approximations instead of the spectral radii, we get* $\epsilon_H \lesssim 0.360$ *for LinBP, and* $\epsilon_H \lesssim 0.455$ *for LinBP* as sufficient (but not necessary) conditions for convergence. Figure 4c and Fig. 4d illustrate that our spectral radii criteria capture the convergence of LinBP and LinBP* exactly.*

6.3 SBP in SQL

The SBP semantics may assign beliefs to a node that depend on an exponential number of paths (exponential in the geodesic number of a node). However, SBP actually allows a simple algorithm in SQL that propagates information across every edge at most once, which justifies our choice of name "single-pass". We achieve this in SQL by adding a table $G(\underline{v}, g)$ to the schema that stores the geodesic number g for each node v. This table G, in turn, also supports efficient updates. In the following, we give two algorithms for (1) the initial assignments of beliefs and (2) addition of explicit beliefs. The Appendix also includes an algorithm for (3) addition of edges to the graph.

(1) Initial belief assignment. Algorithm 2 calculates the initial final beliefs: We start with nodes with explicit beliefs; i.e. geodesic number 0 (line 1). At each subsequent iteration (line 3), we then determine nodes with increasing geodesic number by following edges from previously inserted nodes (i.e. those with geodesic number smaller by 1), but ignoring nodes that have already been visited (i.e. those that are already in G) (line 4). Notice that in a slight abuse of Datalog notation (and for the sake of conciseness), we allow negation on relational atoms with anonymous variables implying a nested not exist query.[17] The beliefs of the new nodes are then calculated by following all edges from nodes that have just been assigned their beliefs in the previous step (line 5). This is repeated for nodes with increasing geodesic numbers until the table G remains unchanged (line 7).

Proposition 21 (Algorithm 2)**.** *Algorithm 2 terminates in finite number of iterations and returns a sound and complete enumeration of final beliefs according to SBP.*

(2) Addition of explicit beliefs. We assume the set of changed or additional explicit beliefs to be available in table

[17]The common syntactic safety restriction is that all variables need to appear in a positive relational atom of the body. In practice, we use a left outer join and an "is null" condition.

Algorithm 3: (ΔSBP:newExplicitBeliefs) Updates B and G, given new explicit beliefs E_n and weighted network A.

Input: $E_n(v,c,b)$, $A(s,t,w)$
Output: Updated $B(v,c,b)$ and $G(v,g)$

1 Initialize geodesic numbers for new nodes with explicit beliefs:
$G_n(v,'0') :- E_n(v,\_,\_)$
$!\, G(v,'0') :- G_n(v,\_)$
2 Initialize beliefs for new nodes:
$B_n(v,c,b) :- E_n(v,c,b)$
$!\, B(v,c,b) :- B_n(v,c,b)$
3 $i \leftarrow 1$
4 **repeat**
5 Find next nodes to update:
$G_n(t,i) :- G_n(s,i-1), A(s,t,\_), \neg\big(G(t,g_t), g_t < i\big)$
$!\, G(v,i) :- G_n(v,i)$
6 Calculate new beliefs for these nodes:
$B_n(t,c_2,sum(w \cdot b \cdot h)) :- G_n(t,i), A(s,t,w), B(s,c_1,b), G(s,i-1), H(c_1,c_2,h)$
$!\, B(v,c,b) :- B_n(v,c,b)$
7 $i \leftarrow i+1$
8 **until** *no more inserts into* G_n
9 **return** B *and* G

$E_n(\underline{v,c},b)$ and use tables $G_n(\underline{v},g)$ and $B_n(\underline{v,c},b)$ to store temporary information for nodes that get updated. We will further use an exclamation mark left of a Datalog query to imply that the respective data record is either inserted or an existing one updated. Algorithm 3 shows the SQL translation for batch updates of explicit beliefs: Line 1 and line 2 initialize tables G_n and B_n for all *new* explicit nodes. At each subsequent iteration i (line 4), we then determine all nodes t that need to be updated with *new* geodesic number $g_t = i$ by following edges from previously updated nodes s with geodesic number $g_s = i-1$ and ignoring those that already have a smaller geodesic number $g_t < i$. (line 5).[18] For these nodes t, the updated beliefs are then calculated by only following edges that start at nodes s with geodesic number $g_s = i-1$, independent of whether those were updated or not (line 6). The algorithm terminates when there are no more inserts in table G_n (line 8).

Proposition 22 (Algorithm 3)**.** *Algorithm 3 terminates in finite number of iterations and returns a sound and complete enumeration of updated beliefs.*

7. EXPERIMENTS

In this section, we experimentally verify how well our new methods LinBP and SBP scale, and how close the top belief classification of both methods matches that of standard BP.

Experimental setup. We implemented main memory-based versions of BP and LinBP in JAVA, and disk-bound versions of LinBP and SBP in SQL. The JAVA implementation uses optimized libraries for sparse matrix operations [39]. When timing our memory-based algorithms, we focus on the running times for computations only and ignore the time for loading data and initializing matrices. For the SQL implementation, we report the times from start to finish on PostgreSQL 9.2 [41]. We are mainly interested in relative performance within a platform (LinBP vs. BP in JAVA, and

[18]Notice that edges $s \to t$ with $g_s \geq g_t$ cannot contain a geodesic path in that direction and are thus ignored. Also notice that, again for the sake of conciseness, we write $\neg(G(t,g), g < i)$ to indicate that nodes t with $g_t < i$ are not updated. In SQL, we used an except clause.

#	Graph characteristics Nodes n	Edges e	e/n	Explicit b. 5%	1‰
1	243	1 024	4.2	12	1
2	729	4 096	5.6	36	1
3	2 187	16 384	7.6	110	3
4	6 561	65 536	10.0	328	7
5	19 683	262 144	13.3	984	20
6	59 049	1 048 576	17.8	2 952	60
7	177 147	4 194 304	23.7	8 857	178
8	531 441	16 777 216	31.6	26 572	532
9	1 594 323	67 108 864	42.6	79 716	1 595

(a) Number of nodes, edges, explicit beliefs

	1	2	3
1	10	-4	-6
2	-4	7	-3
3	-6	-3	9

(b) Unscaled residual coupling m. $\hat{\mathbf{H}}_o$

Figure 6: Synthetic data used for our experiments.

SBP vs. LinBP in SQL) and scalability with graph sizes. Both implementations 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 results, we run BP and LinBP for 5 iterations, and SBP until termination.

Synthetic data. We assume a scenario with $k = 3$ classes and the matrix $\hat{\mathbf{H}}_o$ from Fig. 6b as the unscaled coupling matrix. We study the convergence of our algorithms by scaling $\hat{\mathbf{H}}_o$ with a varying parameter ϵ_H. We created 9 "Kronecker graphs" of varying sizes (see Fig. 6a) which are known to share many properties with real world graphs [30].[19] To generate initial class labels (explicit beliefs), we pick 5% of the nodes in each graph and assign to them two random numbers from $\{-0.1, -0.09, \ldots, 0.09, 0.1\}$ as centered beliefs for two classes (the belief in the third class is then their negative sum due to centering). For timing of incremental updates for SBP (denoted as ΔSBP), we created similar updates for 2% of the nodes with explicit beliefs (corresponding to $1‰ = 0.1\%$ of all nodes in a graph).

DBLP data. For this experiment, we use the DBLP data set from [21] which consists of 36 138 nodes representing papers, authors, conferences, and terms. Each paper is connected to its authors, the conference in which it appeared and the terms in its title. Overall, the graph contains 341 564 edges (counting edges twice according to their direction). Only 3 750 nodes (i.e. $\approx 10.4\%$) are labeled explicitly with one of 4 classes: AI (Artificial Intelligence), DB (Databases), DM (Data Mining), and IR (Information Retrieval). We are assuming homophily, which is represented by the 4×4-matrix in Fig. 8a. Our goal is to label the remaining 89.6% of the nodes.

Measuring classification quality. We take the top beliefs returned by BP as "ground truth" (GT) and are interested in how close the classifications returned by LinBP and SBP come for varying scaling of $\hat{\mathbf{H}}_o$.[20] We measure quality of our methods with *precision* and *recall* as follows: Given a set of top beliefs B_{GT} for a GT labeling method and a set of top beliefs B_{O} of another method (O), let $B_\cap$ be the set of shared beliefs: $B_\cap = B_{\text{GT}} \cap B_{\text{O}}$. Then, recall r measures the portion of GT beliefs that are returned by O: $r = |B_\cap|/|B_{\text{GT}}|$, and precision p measures the portion of "correct" beliefs among B_{O}: $p = |B_\cap|/|B_{\text{O}}|$. Notice that this method naturally handles ties. For example, assume that the GT assigns classes c_1, c_2, c_3 as top beliefs to 3 nodes v_1, v_2, v_3, respectively: $\{v_1 \to c_1, v_2 \to c_2, v_3 \to c_3\}$, whereas the comparison method assigns 4 beliefs: $\{v_1 \to \{c_1, c_2\}, v_2 \to c_2, v_3 \to c_2\}$. Then $r = 2/3$ and $p = 2/4$. As alternative, we also use the F1-score, which is the harmonic mean of precision and recall: $h = \frac{2pr}{p+r}$.

Question 1. *Timing: How fast and scalable are LinBP and SBP as compared to BP in both implementations?*

Result 1. The main memory implementation of LinBP is up to 600 times faster than BP, and the SQL implementation of SBP is more than 10 times faster than LinBP.

Figure 7a and Fig. 7b show our timing experiments in both JAVA and SQL, respectively. Figure 7c shows the times for the 5 largest graphs. Notice that all implementations except BP show approximate linear scaling behavior in the number of edges (as reference, both Fig. 7a and Fig. 7b show a dashed grey line that represents an exact linear scalability of 100 000 edges per second). The main-memory implementation of LinBP is 600 times faster than that of BP for the largest graph. We see at least two reasons for these speedups: (*i*) the LinBP update equations calculate beliefs as function of beliefs. In contrast, the BP update equations calculate, for each node, outgoing messages as function of incoming messages; (*ii*) our matrix formulation of LinBP enables us to use well-optimized JAVA libraries for matrix operations. These optimized operations lead to a highly efficient algorithm. SBP is 10 times faster than LinBP in SQL (we look at this closer in the next question). Not surprisingly, the main-memory JAVA implementation of LinBP is much faster than the disk-bound LinBP implementation in SQL. It is worth mentioning that even though our SQL implementation did not exploit special libraries and is the disk-bound, our SBP implementation in SQL is still faster than the BP implementation in JAVA (!).

Question 2. *Timing: What can the speed-up of SBP over LinBP be mostly attributed to?*

Result 2. SBP needs fewer iterations to converge and requires fewer calculations for each iteration, on average.

Figure 7d shows the time required by our JAVA implementation for both LinBP and SBP within each iteration on graph #7. SBP visits different edges in each iteration, and thus needs a different amount of time for each iteration, whereas LinBP revisits every edge in every iteration again. The fact that SBP needs more time for the 2nd iteration than LinBP, although fewer edges are visited, is a consequence of the overhead for maintaining the indexing structure required to decide on which edges to visit next.

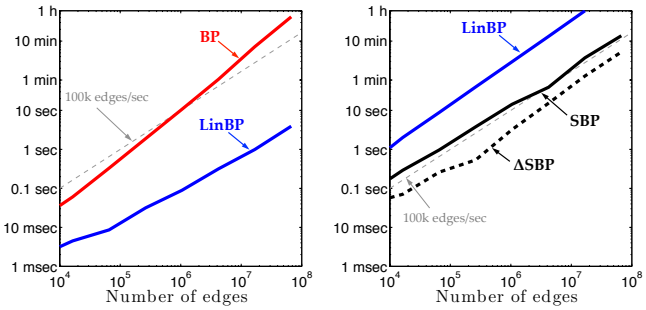
Question 3. *Timing: When is it faster to update a graph incrementally than to recalculate from scratch with SBP?*

Result 3. In our experiments, it was faster to update SBP when less than $\approx 50\%$ of the final explicit beliefs are new.

Figure 7e shows the results for SQL on graph #5. We fix 10% of the nodes with explicit beliefs after the update. Among these nodes, we vary a certain fraction as *new* beliefs. For example, 20% on the horizontal axis implies that

[19] Notice that we count the number of entries in $\mathbf{A}$ as the number of edges; thus, each edge is counted twice ($s-t$ equals $s \to t$ plus $t \to s$).

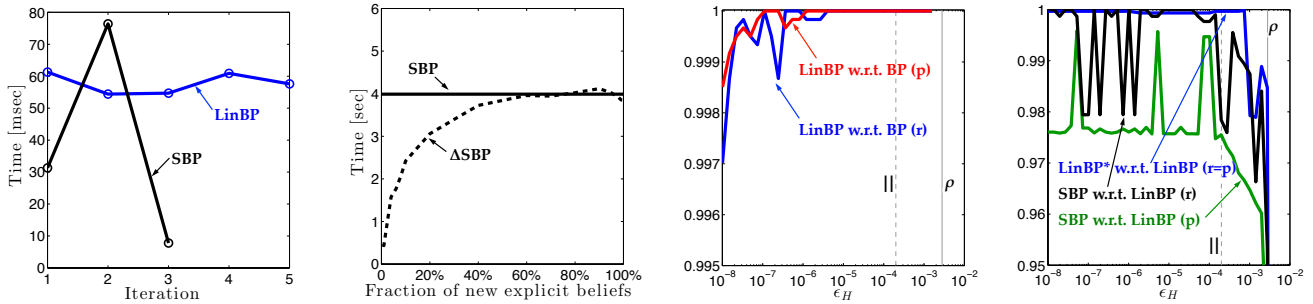
[20] Our experimental approach is justified since BP has previously been shown to work well in real-life classification scenarios. Our goal in this paper is not to justify BP for such inference, but rather to replace BP with a faster and simpler semantics that gives similar classifications.



(a) Scalability JAVA (b) Scalability SQL

#	JAVA [sec] BP	JAVA [sec] LinBP	PostgreSQL [sec] LinBP	PostgreSQL [sec] SBP	PostgreSQL [sec] ΔSBP	Comparisons BP/LinBP	Comparisons LinBP/SBP	Comparisons SBP/ΔSBP
5	2	0.03	40	4.0	0.5	60	10.0	7.5
6	11	0.09	167	14.4	3.2	120	12.3	4.5
7	62	0.32	788	39.1	15.3	198	20.1	2.6
8	430	0.99	3584	222.7	76.0	433	16.1	2.9
9	2 514	3.92	-	820.7	313.5	642	-	2.6

(c) Timing results of all methods in SQL/JAVA on 5 largest graphs



(d) SBP/LinBP in JAVA on #7 (e) ΔSBP/SBP in SQL on #5 (f) Recall & precision on #5 (g) Recall & precision on #5

Figure 7: (a)-(c): Scalability of methods in Java and SQL: dashed gray lines represent linear scalability. (d): ΔSBP vs. SBP for various fractions of updates assuming 10% explicit beliefs. (e): Timing LinBP and SBP per iteration. (f),(g): Quality of LinBP w.r.t. BP, and SBP w.r.t. LinBP: the vertical gray lines mark $\epsilon_H = 0.0002$, i.e. the sufficience convergence criterium from Lemma 9.

we had 80% of the explicit nodes (= 8% of all nodes) known before the update, and are now adding 20% of the explicit nodes (= 2% of all nodes) with the incremental SBP Algorithm 3 ("ΔSBP"). For the alternative Algorithm 2 ("SBP"), we recalculate the final beliefs from scratch (therefore, shown with a constant horizontal line). In addition, Fig. 7c shows timing for updating 1‰ of the nodes in a graph that previously had 5% nodes with explicit beliefs: The relative speed-up is around 2.5 for the larger graphs.

Question 4. *Quality: How do the top belief assignments of LinBP, LinBP* and SBP compare to that of BP?*

Result 4. BP, LinBP, LinBP*, and SBP give almost identical top belief assignments (for ϵ_H given by Lemma 9). However, ties can drop the quality of SBP to <95%.

Figure 7f shows recall (r) and precision (p) of LinBP with BP as GT ("LinBP with regard to BP") on graph #5 (similar results hold for all other graphs). The vertical gray lines show $\epsilon_H = 0.0002$ and $\epsilon_H = 0.0028$, which result from our sufficient (Lemma 9) and exact (Lemma 8) convergence criteria of LinBP, respectively. The graphs stop earlier than $\epsilon_H = 0.0028$ as BP stops converging earlier. We see that LinBP matches the top belief assignment of BP exactly in the upper range of guaranteed convergence; for smaller ϵ_H, errors result from roundoff errors due to limited precision of floating-point computations. We thus recommend choosing ϵ_H according to Lemma 8. Overall accuracy (harmonic mean of precision and recall) is still > 99.9% across all ϵ_H.

Figure 7g shows that the results of LinBP and LinBP* are almost identical as long as ϵ_H is small enough for the algorithms to converge (both LinBP and LinBP* always return unique top belief assignments; thus, r and p are identical and we only need to show one graph for both). The vertical drops in r and p on the right correspond to choices of ϵ_H for which LinBP stops converging.

Figure 7g also validates that SBP closely matches LinBP (and thus BP). The averaged recall of SBP w.r.t. LinBP for $10^{-9} < \epsilon_H < 0.0002$ is 0.995 and the averaged precision 0.978. Thus overall accuracy is > 98.6% across all ϵ_H. The visible oscillations and the observation that SBP's precision values are generally lower than its recall values are mainly due to "*tied top beliefs*:" the final beliefs are almost identical, but SBP returns two top beliefs, while LinBP returns only one. For example, we observed the following final beliefs which lead to a drop in precision (due to SBP's tie):

- LinBP: $[1.0000000014, 1.0000000002, -2.0000000016] \cdot 10^{-2}$
- SBP: $[1, 1, -2] \cdot 10^{-2}$

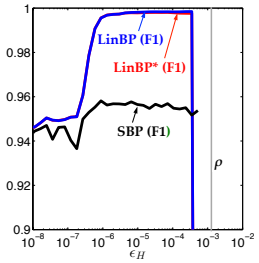
The following more rare scenario is due to numerical rounding errors and led to a drop in both precision and recall (LinBP and SBP return two different top beliefs):

- LinBP: $[7.60009, 7.60047, -15.20056] \cdot 10^{-11}$
- SBP: $[7.6, 7.59999999999999, -15.2] \cdot 10^{-11}$

Minimizing the possibility of ties by choosing initial explicit beliefs with additional digits (e.g., 0.0503 instead of 0.05) removed these oscillations. On the other hand, if there are many tied explicit beliefs (such as in the DBLP data where all explicit nodes have one among 4 different beliefs), the difference between SBP and BP increases. Figure 8b shows that, for the DBLP data set, SBP performs worse than LinBP due to many ties. The absolute accuracy, however, is still above 95%. LinBP and LinBP* approximate BP very well as long as BP converges. LinBP converges for $\epsilon_H < 0.0013$, however BP stops converging earlier: This explains the gap between the convergence bounds for LinBP, and when the accuracy actually drops. For very small ϵ_H,

	1	2	3	4
1	6	-2	-2	-2
2	-2	6	-2	-2
3	-2	-2	6	-2
4	-2	-2	-2	6

(a) Unscaled residual coupling matrix $\hat{\mathbf{H}}_o$



(b) F1 on DBLP data

Figure 8: Coupling matrix and quality results on DBLP data.

we see results from floating-point rounding errors.

In summary, SBP and LinBP match the classification of BP very well. Misclassifications are mostly due to closely tied top beliefs, in which case returning *both* tied beliefs (as done by SBP) would arguably be *the preferable alternative*.

8. RELATED WORK

The two main philosophies for transductive inference are *logical* approaches and *statistical* approaches (see Fig. 9).

Logical approaches determine the solution based on hard rules, and are most common in the database literature. Examples are trust mappings, preference-based updates, stable model semantics, but also tuple-generating dependencies, inconsistency-resolution, database repairs, community databases. Example applications are peer-data management and collaborative data sharing systems that have to deal with conflicting data and lack of consensus about which data is correct during integration, update exchange, and that have adopted some form of conflict handling or trust mappings in order to facilitate data sharing among users [3, 12, 14, 16, 17, 23, 24, 46]. Commonly, those inconsistencies are expressed with key violations [10] and resolved at query time through database repairs [1].

Statistical approaches determine the solution based on soft rules. The related work comprises guilt-by-association approaches, which use limited prior knowledge and network effects in order to derive new knowledge. The main alternatives are semi-supervised learning (SSL), random walks with restarts (RWR), and label or belief propagation (BP). SSL methods can be divided into low-density separation methods, graph-based methods, methods for changing the representation, and co-training methods (see [31, 50] for overviews). A multi-class approach has been introduced in [21]. RWR methods are used to compute mainly node relevance; e.g., original and personalized PageRank [4, 18], lazy random walks [33], and fast approximations [37, 47].

Belief Propagation (or min-sum or product-sum algorithm) is an iterative message-passing algorithm that is a very expressive formalism for assigning classes to unlabeled nodes and has been used successfully in multiple settings for solving inference problems, such as error-correcting codes [29] or stereo imaging in computer vision [9], fraud detection [32, 38], malware detection[5], graph similarity [2, 27], structure identification [25], and pattern mining and anomaly detection [22]. BP solves the inference problem approximately; it is known that when the factor graph has a tree structure, it reaches a stationary point (convergence to the true marginals) after a finite number of iterations. Although in loopy factor graphs, convergence to the correct marginals

Databases	Machine Learning
Inconsistency resolution	Semi-supervised learning
Logic-based approaches	Statistical approaches
extensional database	prior beliefs
intensional database	posterior beliefs

Figure 9: Comparing common formulations of transductive inference in the database and machine learning communities.

is not guaranteed, the true marginals may still be achieved in *locally* tree-like structures. As a consequence, approaches in the database community that rely on BP-type of inference also commonly lack convergence guarantees [45].

Convergence of BP in loopy graphs has been studied before [8, 20, 34]. To the best of our knowledge, all existing bounds for BP give only sufficient convergence criteria. In contrast, our work presents a stronger result by providing sufficient *and necessary* conditions for the convergence of LinBP, which is itself an approximation of BP. Other recent work [28] studies a form of linearization for unsupervised classification in the stochastic block model without an obvious way to include supervision in this setting.

There exist various works that speed up BP by: (*i*) exploiting the graph structure [6, 38], (*ii*) changing the order of message propagation [8, 15, 34], or (*iii*) using the MapReduce framework [22]. Here, we derive a linearized formulation of standard BP. This is a multivariate ("polytomous") generalization of the linearized belief propagation algorithm FABP [26] from binary to multiple labels for classification. In addition, we provide translations into SQL and a new, faster semantics that captures the underlying intuition and provides efficient incremental updates.

Incremental maintenance. While our nearest-labeled-neighbor-semantics SBP allows efficient incremental updates (cf. Lemma 17), incrementally updating LinBP is more challenging since it involves general matrix computations. For such scenarios, combining our work with approaches like the one from [36] is left for future work.

9. CONCLUSIONS

This paper showed that the widely used multi-class belief propagation algorithm can be approximated by a linear system that replaces multiplication with addition. This allows us to give a fast and compact matrix formulation and a compact implementation in standard SQL. The linear system also allows a closed-form solution with the help of the inverse of an appropriate matrix. We can thus explain *exactly* when the system will converge, and what the limit value is as the neighbor-to-neighbor influence tends to zero. For the latter case, we show that the scores depend only on the "nearest labeled neighbor," which leads to an even faster algorithm that also supports incremental updates.

Acknowledgements. This work was supported in part by NSF grants IIS-1217559 and IIS-1408924. Stephan Günnemann has been supported by a fellowship within the postdoc-program of the German Academic Exchange Service (DAAD). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties. We would also like to thank Garry Miller for pointing us to Roth's column lemma and the anonymous reviewers for their careful feedback.

10. REFERENCES

[1] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *PODS*, pp. 68–79, 1999.

[2] M. Bayati, M. Gerritsen, D. Gleich, A. Saberi, and Y. Wang. Algorithms for large, sparse network alignment problems. In *ICDM*, pp. 705–710, 2009.

[3] P. A. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data management for peer-to-peer computing: A vision. In *WebDB*, pp. 89–94, 2002.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

[5] D. H. Chau, C. Nachenberg, J. Wilhelm, A. Wright, and C. Faloutsos. Polonium: Tera-scale graph mining and inference for malware detection. In *SDM*, pp. 131–142, 2011.

[6] A. Chechetka and C. Guestrin. Focused belief propagation for query-specific inference. In *AISTATS*, pp. 89–96, 2010.

[7] S. Cohen, W. Nutt, and Y. Sagiv. Containment of aggregate queries. In *ICDT*, pp. 111–125, 2003.

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

[9] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *Int. J. Comput. Vision*, 70(1):41–54, Oct. 2006.

[10] A. Fuxman, E. Fazli, and R. J. Miller. Conquer: Efficient management of inconsistent databases. In *SIGMOD*, pp. 155–166, 2005.

[11] W. Gatterbauer. Semi-supervised learning with heterophily, Dec 2014. (CoRR abs/1412.3100).

[12] W. Gatterbauer, M. Balazinska, N. Khoussainova, and D. Suciu. Believe it or not: Adding belief annotations to databases. *PVLDB*, 2(1):1–12, 2009.

[13] W. Gatterbauer, S. Günnemann, D. Koutra, and C. Faloutsos. Linearized and single-pass belief propagation, June 2014. (CoRR abs/1406.7288).

[14] W. Gatterbauer and D. Suciu. Data conflict resolution using trust mappings. In *SIGMOD*, pp. 219–230, 2010.

[15] J. Gonzalez, Y. Low, and C. Guestrin. Residual splash for optimally parallelizing belief propagation. *Journal of Machine Learning Research - Proceedings Track*, 5:177–184, 2009.

[16] T. J. Green, G. Karvounarakis, N. E. Taylor, O. Biton, Z. G. Ives, and V. Tannen. ORCHESTRA: facilitating collaborative data sharing. In *SIGMOD*, pp. 1131–1133, 2007.

[17] A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *ICDE*, pp. 505–516, 2003.

[18] T. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Trans. Knowl. Data Eng.*, pp. 784–796, 2003.

[19] H. V. Henderson and S. R. Searle. The vec-permutation matrix, the vec operator and Kronecker products: a review. *Linear and Multilinear Algebra*, 9(4):271–288, 1981.

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

[21] M. Ji, Y. Sun, M. Danilevsky, J. Han, and J. Gao. Graph regularized transductive classification on heterogeneous information networks. In *ECML/PKDD (1)*, pp. 570–586, 2010.

[22] U. Kang, D. H. Chau, and C. Faloutsos. Mining large graphs: Algorithms, inference, and discoveries. In *ICDE*, pp. 243–254, 2011.

[23] A. Kementsietsidis, M. Arenas, and R. J. Miller. Mapping data in peer-to-peer systems: Semantics and algorithmic issues. In *SIGMOD*, pp. 325–336, 2003.

[24] L. Kot and C. Koch. Cooperative update exchange in the Youtopia system. *PVLDB*, 2(1):193–204, 2009.

[25] D. Koutra, U. Kang, J. Vreeken, and C. Faloutsos. VoG: Summarizing and understanding large graphs. In *SDM*, pp. 91–99, 2014.

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

[27] D. Koutra, J. Vogelstein, and C. Faloutsos. Deltacon: A principled massive-graph similarity function. In *SDM*, pp. 162–170, 2013.

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

[29] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[30] J. Leskovec, D. Chakrabarti, J. M. Kleinberg, and C. Faloutsos. Realistic, mathematically tractable graph generation and evolution, using Kronecker multiplication. In *PKDD*, pp. 133–145, 2005.

[31] S. A. Macskassy and F. J. Provost. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, 8:935–983, 2007.

[32] M. McGlohon, S. Bay, M. G. Anderle, D. M. Steier, and C. Faloutsos. SNARE: a link analytic system for graph labeling and risk detection. In *KDD*, pp. 1265–1274, 2009.

[33] E. Minkov and W. Cohen. Learning to rank typed graph walks: Local and global approaches. In *WebKDD workshop on Web mining and social network analysis*, pp. 1–8, 2007.

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

[35] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT Press, 2012.

[36] M. Nikolic, M. Elseidy, and C. Koch. LINVIEW: incremental view maintenance for complex analytical queries. In *SIGMOD*, pp. 253–264, 2014.

[37] J. Pan, H. Yang, C. Faloutsos, and P. Duygulu. GCap: Graph-based automatic image captioning. In *MDDE*, p. 146, 2004.

[38] S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *WWW*, pp. 201–210, 2007.

[39] Parallel Colt: `http://sourceforge.net/projects/parallelcolt/`.

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

[41] PostgreSQL 9.2: `http://www.postgresql.org/download/`.

[42] W. E. Roth. On direct product matrices. *Bull. Amer. Math. Soc.*, 40:461–468, 1934.

[43] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, 2nd ed edition, 2003.

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

[45] F. M. Suchanek, S. Abiteboul, and P. Senellart. Paris: Probabilistic alignment of relations, instances, and schema. *PVLDB*, 5(3):157–168, 2011.

[46] N. E. Taylor and Z. G. Ives. Reconciling while tolerating disagreement in collaborative data sharing. In *SIGMOD*, pp. 13–24, 2006.

[47] H. Tong, C. Faloutsos, and J. Pan. Fast random walk with restart and its applications. In *ICDM*, pp. 613–622, 2006.

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

[49] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In *Exploring artificial intelligence in the new millennium*, pp. 239–269. 2003.

[50] X. Zhu. Semi-supervised learning literature survey, 2006.

NOMENCLATURE

n	number of nodes
s, t, u	indices used for nodes
$N(s)$	list of neighbours for node s
k	number of classes
i, j, g	indices used for classes
$\mathbf{e}_s$	k-dimensional prior (explicit) belief vector at node s
$\mathbf{b}_s$	k-dim. posterior (implicit, final) belief vector at node s
$\mathbf{m}_{st}$	k-dim. message vector from node s to node t
$\mathbf{A}$	$n \times n$ weighted symmetric adjacency matrix
$\mathbf{D}$	$n \times n$ diagonal degree matrix
$\mathbf{E}, \mathbf{B}$	$n \times k$ explicit or implicit belief matrix with $E(s,i)$ indicating the strength of belief in class i by node s
$\mathbf{H}$	$k \times k$ coupling matrix with $H(j,i)$ indicating the influence of class j of a sender on class i of the recipient
$\hat{\mathbf{H}}, \hat{\mathbf{E}}, \hat{\mathbf{B}}$	residual matrices centered around $\frac{1}{k}$
$\hat{\mathbf{H}}_o$	unscaled, original coupling matrices $\hat{\mathbf{H}} = \epsilon_H \hat{\mathbf{H}}_o$
ϵ_H	scaling factor
$\mathbf{I}_k$	k-dimensional identity matrix
$\mathtt{vec}(\mathbf{X})$	vectorization of matrix $\mathbf{X}$
$\mathbf{X} \otimes \mathbf{Y}$	Kronecker product between matrices $\mathbf{X}$ and $\mathbf{Y}$
$\rho(\mathbf{X})$	spectral radius of a matrix $\mathbf{X}$