Supervised Belief Propagation: Scalable Supervised Inference on Attributed Networks

Jaemin Yoo
Seoul National University
Seoul, Republic of Korea
jaeminyoo@snu.ac.kr

Saehan Jo
Seoul National University
Seoul, Republic of Korea
naheas@snu.ac.kr

U Kang
Seoul National University
Seoul, Republic of Korea
ukang@snu.ac.kr

***Abstract*—Given an undirected network where some of the nodes are labeled, how can we classify the unlabeled nodes with high accuracy? Loopy Belief Propagation (LBP) is an inference algorithm widely used for this purpose with various applications including fraud detection, malware detection, web classification, and recommendation. However, previous methods based on LBP have problems in modeling complex structures of attributed networks because they manually and heuristically select the most important parameter, the propagation strength.**

In this paper, we propose Supervised Belief Propagation (SBP), a scalable and novel inference algorithm which automatically learns the optimal propagation strength by supervised learning. SBP is generally applicable to attributed networks including weighted and signed networks. Through extensive experiments, we demonstrate that SBP generalizes previous LBP-based methods and outperforms previous LBP and RWR based methods in real-world networks.

I. Introduction

Given an attributed network whose edges have weights or signs, how can we classify its nodes into different categories? Node classification is a crucial task with many applications including anomaly and fraud detection [1], [2], [3], [4], link prediction [5], [6], [7], sign prediction [8], and recommendation [9]. Loopy Belief Propagation (LBP) [10], an inference algorithm for probabilistic graphical models, has been widely used for solving node classification problems. Intuitively, LBP is based on the notion of *guilt-by-association*: if a user is a drug-abuser, then it is likely that its neighbors are drug-abusers as well. LBP propagates prior knowledge on observed nodes to infer labels or states of unobserved nodes. Due to its simplicity and generality, LBP has been widely used for solving real-world problems including malware detection [2], [11], fraud detection [1], [4], image processing [12], etc.

However, previous works on LBP have two main limitations. First, they do not provide an algorithmic way of determining the *propagation strength,* which models the degrees of association between adjacent nodes. Instead, heuristic methods are applied to choose the value with no theoretical justification. The difficulty of manually choosing the propagation strength leads to using only a small number of parameters in LBP-based methods: e.g., many existing works [2], [3], [4] use a single heuristically determined propagation strength to uniformly model all the edges in a network. Second, previous works on LBP do not utilize rich information available in attributed networks since choosing the propagation strength is difficult and thus they need to simplify the networks. Although some LBP-based methods were proposed to use the information in attributed networks, they solely focus on specific types of networks, not general ones [13], [9]. Furthermore, they do not provide a way to choose the propagation strength as well. As a result, this rich information is left unused, although it could give useful insights on the identity of nodes in networks.

In this paper, we propose Supervised Belief Propagation (SBP), a scalable inference algorithm for attributed networks designed to overcome the limitations of previous LBP-based methods. SBP learns optimal propagation strengths which had to be chosen manually and heuristically in previous methods. Moreover, SBP extends the model capacity compared to previous methods: SBP allows the use of multiple parameters to consider rich features in attributed networks in determining the propagation strengths. SBP outperforms state-of-the-art node classification methods based on LBP and RWR (Random Walk with Restart). Our main contributions are the following:

- **Algorithm.** We propose SBP, a scalable inference algorithm for attributed networks. SBP automatically learns optimal propagation strengths and reveals relative importance of each attribute. SBP is general enough to take any attributed network as an input and model its attributes.
- **Accuracy.** SBP shows the highest accuracy in node classification, outperforming previous LBP and RWR based methods. SBP provides up to 15.6% higher AUC in real-world datasets as shown in Figure 1.
- **Scalability.** SBP is scalable to large networks, providing linear running time and memory requirement with regard to the number of edges.

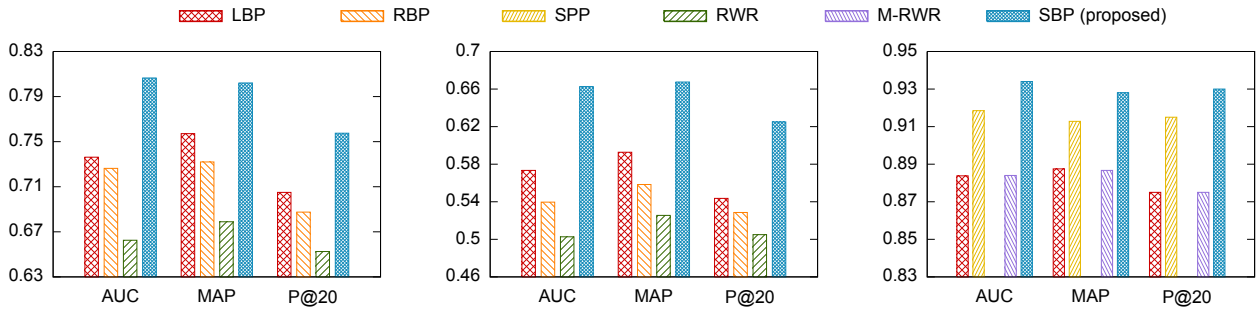
The codes and datasets for this paper are publicly available.[1] The rest of this paper is organized as follows: Section II presents preliminaries. Section III proposes SBP and shows its scalability. Section IV presents experimental results. Section V introduces related works. We conclude the paper and give ideas of future work in Section VI.

II. Preliminaries

In this section, we describe preliminaries on Supervised Belief Propagation (SBP). Symbols we use throughout this paper are summarized in Table I. In Section II-A, we introduce

[1] http://datalab.snu.ac.kr/sbp

2374-8486/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDM.2017.69



(a) MovieLens (b) Epinions-R (c) Epinions-S

Fig. 1: Classification accuracies for all the datasets: (a) MovieLens, (b) Epinions-R, and (c) Epinions-S. SBP shows the highest accuracies for all the datasets: up to 15.6% higher AUC, 12.6% higher MAP, and 15.0% higher P@20 compared to the best existing methods. RWR shows a low AUC of 71.6% for Epinions-S, which is not shown in (c).

TABLE I: Table of symbols.

Symbol	Definition
θ_{ij}	feature vector for edge (i,j)
w	weight vector to be learned
s_p, s_n	positive and negative states
P, N	sets of positive and negative nodes
$(\cdot)_{\text{obs}}$	observed nodes for the propagation step
$(\cdot)_{\text{trn}}$	training nodes for the weight update step
ϵ_{ij}	propagation strength for edge (i,j)
m_{ij}	normalized message for edge (i,j)
b_i	normalized belief for node i
ϕ_i	normalized node potential for node i
$m^*_{ij}(\cdot)$	unnormalized message for edge (i,j)
$b^*_i(\cdot)$	unnormalized belief for node i
$E(\cdot)$	cost function to be minimized
α, β, λ	parameters for gradient update of w
η	number of recursive updates for ∂w

Loopy Belief Propagation (LBP). In Section II-B, we introduce the *edge potential table*, which is widely used to make LBP-based methods applicable to large real-world networks.

A. Loopy Belief Propagation

A pairwise Markov Random Field (MRF) is a set of discrete random variables whose joint relationships are modeled as an undirected graph. Given the variables $X = (X_i)_{i \in V}$ which are modeled as a graph (V, E) where V and E represent the sets of nodes and edges, respectively, the joint probability $p(X = x)$ is computed by multiplying all the *potentials* ϕ and ψ as

$$p(X = x) = \frac{1}{Z} \prod_{i \in V} \phi_i(x_i) \prod_{(i,j) \in E} \psi_{ij}(x_i, x_j),$$

where Z is a normalization constant. A node potential $\phi_i(x_i)$ represents an unnormalized probability of node i being in state x_i without considering the influences from other nodes. An edge potential $\psi_{ij}(x_i, x_j)$ represents a joint unnormalized probability of nodes i and j being in states x_i and x_j.

LBP is an approximate algorithm to compute the marginal distribution of a pairwise MRF by passing *messages* between the variables [10], [14]. A message $m^*_{ij}(x_j)$ is an opinion of node i about the probability of node j being in state x_j. LBP uniformly initializes all the messages and updates them through iterations until they converge. Equation (1) shows how to update $m^*_{ij}(x_j)$ at each iteration where $N(i)$ denotes the set of neighbors of node i. All the incoming messages from $N(i)$ except node j are multiplied to compute $m^*_{ij}(x_j)$.

$$m^*_{ij}(x_j) \leftarrow \sum_{x_i} \phi_i(x_i) \psi_{ij}(x_i, x_j) \frac{\prod_{k \in N(i)} m^*_{ki}(x_i)}{m^*_{ji}(x_i)} \quad (1)$$

LBP returns *beliefs* as a result. A belief $b_j(x_j)$ is an approximate marginal probability of node j being in state x_j, which is computed from the converged messages and then normalized. Equation (2) shows how to compute the unnormalized beliefs and then how to normalize them accordingly.

$$\begin{aligned} b_j(x_j) &= \frac{b^*_j(x_j)}{\sum_{x'_j} b^*_j(x'_j)} \text{ where} \\ b^*_j(x_j) &= \phi_j(x_j) \prod_{i \in N(j)} m^*_{ij}(x_j) \end{aligned} \quad (2)$$

A popular application of LBP is node classification whose goal is to infer the states of unobserved nodes based on the known states of observed nodes. LBP solves this by assigning different node potentials to the nodes and then propagating the messages. For instance, assuming random variables having two states, LBP gives a node potential $(0.9, 0.1)$ or $(0.1, 0.9)$ to a observed node based on its observed state. On the other hand, LBP gives a node potential $(0.5, 0.5)$ to the unobserved nodes. LBP propagates the messages from the biased potentials of the observed nodes and classifies the unobserved nodes. Although it is not guaranteed, the messages often converge in a small number of iterations for most real-world networks.

B. Edge Potential Table

LBP has been widely used in solving real-world problems due to its simplicity and scalability [1], [4], [3]; running time of LBP scales linearly with the number of edges. Nevertheless, previous methods have difficulty in assigning edge potentials because it requires too many parameters. Thus, a majority of existing methods tend to simplify the models and assume that all edges in a network have the same edge potentials, which are represented by *edge potential tables*. The use of edge potential tables makes such methods simple enough to be applicable to large networks by reducing the number of parameters.

Given a real-world network modeled as a pairwise MRF, an edge potential table is defined as a $k \times k$ table where k is the

TABLE II: Edge potential table with propagation strength ϵ for a binary network whose nodes are either positive or negative.

State	positive	negative
positive	ϵ	$1-\epsilon$
negative	$1-\epsilon$	ϵ

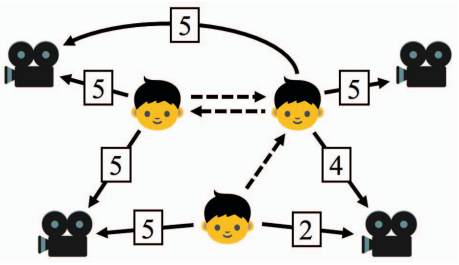


Fig. 2: Attributed network which contains two types of nodes (users and items) and edges (trusts and reviews). The dashed arrows represent trusts, and the solid arrows represent reviews with ratings between 1 and 5.

number of states of its variables. Table II shows an example of an edge potential table of a network whose nodes have two states: positive and negative. Given a *propagation strength* ϵ between 0 and 1, the table denotes that the joint probability of adjacent nodes is determined by ϵ. Formally, the edge potential $\psi_{ij}(x_i, x_j)$ for each edge (i, j) is given as follows:

$$\psi_{ij}(x_i, x_j) = \begin{cases} \epsilon, & \text{if } x_i = x_j \\ 1-\epsilon, & \text{otherwise} \end{cases} \tag{3}$$

It has been a challenging problem to choose the right value of the propagation strength [3]. A common workaround is a grid search which finds the best parameter from a finite set of reasonable values by running the algorithm for each possible case. For example, we choose ϵ in $\{0.6, 0.51, 0.501, 0.5001\}$. However, such an approach restricts introducing more parameters since it takes exponential time in the number of parameters. Previous methods [9] use only one or two propagation strengths for rich attributed networks although their attributes contain meaningful information about the inference.

We introduce Figure 2 to further elaborate on the limitation of the current model used in existing LBP-based methods. The figure shows an attributed network which contains two types of nodes (users and items) and edges (trusts and reviews). The review edges contain ratings between 1 and 5, while the trust edges have no additional information. Assuming that we want to classify the items into *recommended* and *not recommended* for a given user, it is impractical to find the best propagation strengths for each type and rating using the grid search. The typical approach is to simplify the edges by eliminating the attributes. However, such simplification suffers from the loss of rich information which leads to poor accuracy.

III. Proposed Method

In this section, we propose Supervised Belief Propagation (SBP), a scalable inference algorithm for attributed networks with the following improvements:

Algorithm 1: Supervised Belief Propagation

Input: attributed network G, sets P and N of positive and negative nodes, and node potential ϕ
Output: beliefs b for all nodes

$P_{\text{obs}}, P_{\text{trn}} \leftarrow$ randomly split P into two sets
$N_{\text{obs}}, N_{\text{trn}} \leftarrow$ randomly split N into two sets
$w \leftarrow$ an initial weight vector
while convergence criterion of w is not met **do**
  $b, m \leftarrow propagate(w, P_{\text{obs}}, N_{\text{obs}}, \phi)$
  $w \leftarrow weight\_update(w, b, m, P_{\text{trn}}, N_{\text{trn}})$
end while
$b, m \leftarrow propagate(w, P, N, \phi)$
return b

- SBP automatically learns optimal propagation strengths which previous methods have manually chosen.
- SBP utilizes rich information of attributed networks by considering these attributes in the message propagation.

After giving an overview of the algorithm in Section III-A, we describe the details in the following sections. In Section III-B, we introduce how to encode the attributes in attributed networks. In Sections III-C, III-D, and III-E, we describe main steps of SBP in detail. In Section III-F, we show space and time complexities of the algorithm.

A. Overview

SBP is an inference algorithm which overcomes the limitations of previous LBP-based methods and thus is generally applicable to attributed networks. Given an attributed network whose attributes are encoded as feature vectors, SBP learns *weights* of the features to determine the propagation strengths; edges with different feature vectors have different propagation strengths depending on the learned weights. This enables SBP to utilize rich information in attributed networks, resulting in more accurate inference than existing methods.

SBP initializes a weight vector w and alternates two main steps to train it. In the *propagation step*, SBP computes the messages and beliefs based on the current w. This step is similar to the standard LBP algorithm which uses fixed values for the propagation strengths. In the *weight update step*, SBP updates w based on the computed messages and beliefs so that w moves toward a local optimum. These steps are described in detail in Sections III-C and III-D, respectively.

The main idea of weight optimization is to train the weight vector w so that the beliefs of nodes are in accordance with their true states. For this purpose, given a set of nodes whose true states are known, we hide the labels of some nodes to use them as correct answers in training w. We call these nodes as *training nodes* and the others as *observed nodes*. We use only the observed nodes as evidence of the inference and only the training nodes to update the weight vector.

Although the main ideas of SBP are generally applicable to multi-label classification, we focus on a binary case due to its simplicity. In binary classification, we assume that every node has two possible states s_p and s_n, and we call a node positive or negative if its state is known as s_p or s_n, respectively.

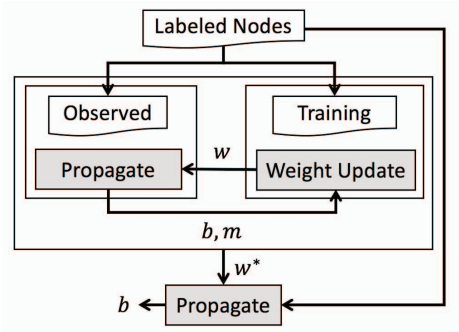


Fig. 3: Flowchart of SBP representing Algorithm 1. SBP splits labeled nodes into observed and training, and then alternates propagation and weight update steps until w converges. After the convergence, SBP computes beliefs of all the nodes using the learned w^* and returns them.

Algorithm 1 shows SBP. In lines 1 and 2, we split each set of nodes into observed and training, as we discussed above, in order to separate the nodes for the propagation and weight update steps. In lines 3 to 7, we initialize w and then iteratively update it until it converges. After the convergence, we compute the beliefs using all the positive and negative nodes and return them. Figure 3 shows a summary of the algorithm.

The stopping criterion in line 4 of Algorithm 1 for updating w is either one of the following: 1) the difference between w in consecutive iterations is within a small threshold, or 2) the maximum number of iterations is reached. This criterion is also used for the message updates in the propagation step.

B. Encoding Attributed Networks

We encode an attributed network as a directed graph where each edge (i, j) contains a feature vector θ_{ij} generated from the original attributes. SBP determines propagation strength ϵ_{ij} customized for edge (i, j) using its feature vector θ_{ij} and the globally trained weight vector w. As there can be multiple possible encodings for a single attributed network, choosing a proper encoding θ_{ij} from the attributes is important for the performance of the algorithm.

For instance, let us consider the attributed network in Figure 2, which is discussed in Section II-B. The network contains two kinds of attributes: edge type and rating. While the edge type attribute is naturally encoded as a binary vector of length 1, the rating can be encoded in various ways: 1) a continuous vector of length 1 with the raw score or 2) a one-hot encoded vector of length 5.

At first glance, the first encoding seems natural. However, it is problematic in that discrete attributes are modeled as a continuous integer. This assumes a strict linear scaling of preferences where rating 5 is exactly 5 times higher than rating 1 although their exact correlations in the network are unknown. On the contrary, the second separates all the ratings so that each represents a distinctive evaluation. In SBP, which learns different optimal weights for each feature element, the second gives better results than the first one.

TABLE III: Two types of feature vectors with different encodings for the network of Figure 2. While the first encoding assumes the linear correlation between the ratings, the second makes SBP learn independent weights for each feature.

Edge Type	Encoding 1	Encoding 2
Trust	$(1, 0)$	$(1, 0, 0, 0, 0, 0)$
Review with rating 1	$(0, 1)$	$(0, 1, 0, 0, 0, 0)$
Review with rating 2	$(0, 2)$	$(0, 0, 1, 0, 0, 0)$
Review with rating 3	$(0, 3)$	$(0, 0, 0, 1, 0, 0)$
Review with rating 4	$(0, 4)$	$(0, 0, 0, 0, 1, 0)$
Review with rating 5	$(0, 5)$	$(0, 0, 0, 0, 0, 1)$

After encoding each attribute as a vector, we concatenate them to generate the feature vectors. Table III shows two types of feature vectors with different encodings for the network in Figure 2. The first element represents the edge type as a binary integer and the rest represent the rating.

C. Propagation Step

In the propagation step (lines 5 and 8 in Algorithm 1), the messages and beliefs are computed based on the current weight vector w. After the messages converge through iterations, the beliefs are computed from the converged messages. The message propagation in SBP have two main differences compared to that in previous LBP-based methods: 1) the messages are propagated with different strengths customized for the feature vector of each edge and 2) computations of the messages and beliefs are optimized for binary networks.

Propagation strength ϵ_{ij} which is customized for each edge (i, j) is modeled as Equation (4). Its value is determined by the inner product between the feature vector θ_{ij} of edge (i, j) and the current weight vector w, and is interpreted as a probability between 0 and 1. As we model ϵ_{ij} in this way, we 1) easily consider both θ_{ij} and w in determining the propagation strength, and 2) improve stability of the algorithm by avoiding extreme values of ϵ_{ij}. It also can be considered as a function of θ_{ij} given w as a parameter. Thus, its value changes during the algorithm as w is updated at each iteration.

$$\epsilon_{ij} = (1 + \exp(-\theta_{ij}^{\mathrm{T}} w))^{-1} \tag{4}$$

Next, we describe efficient computations of SBP for binary networks. We optimize Equations (1) and (2) of LBP assuming a binary network whose nodes have two possible states: s_p and s_n. As a result, we have Equations (5), (6), and (7) which are more efficient. Note that the simplified equations lead to the same results as those from the original equations.

First, we normalize the messages and use only the positive ones. It gives the following advantages: 1) we avoid numerical underflow when multiplying a lot of messages on high-degree nodes, 2) it saves the cost of space to store both messages, and 3) it simplifies the equations. Resulting beliefs computed from the converged messages remain the same because any constant multiplied to the messages cancels out when we normalize the beliefs. Equation (5) shows how to compute the normalized message m_{ij} from the unnormalized ones m^*_{ij}.

$$m_{ij} = m^*_{ij}(s_p)/(m^*_{ij}(s_p) + m^*_{ij}(s_n)) \tag{5}$$

Then, using the normalized messages, unnormalized beliefs $b^*_j(s_p)$ and $b^*_j(s_n)$ for each node j are computed as in Equation

Algorithm 2: $propagate(\cdot)$

Input: weight vector w, sets $P_{\rm obs}$ and $N_{\rm obs}$ of positive and negative observed nodes, and node potential ϕ
Output: computed beliefs b and messages m

1: $\phi_i \leftarrow 0.5$ for each node i not in $P_{\rm obs}$ and $N_{\rm obs}$
2: $\phi_i \leftarrow \phi$ for each node i in $P_{\rm obs}$
3: $\phi_i \leftarrow 1-\phi$ for each node i in $N_{\rm obs}$
4: $m_{ij} \leftarrow 0.5$ for each edge (i,j)
5: $\epsilon_{ij} \leftarrow (1+\exp(-\theta_{ij}^{\rm T} w))^{-1}$ for each edge (i,j)
6: **while** convergence criterion of m is not met **do**
7: $\quad b \leftarrow$ compute beliefs using m and ϕ
8: $\quad m \leftarrow$ compute messages using ϵ, m, and b
9: **end while**
10: **return** b, m

(6), and then normalized as in Equation (2). Note that the node potentials ϕ_j are normalized in the same way as the messages; ϕ_j and $1-\phi_j$ represent $\phi_j(s_p)$ and $\phi_j(s_n)$, respectively.

$$\begin{aligned} b_j^*(s_p) &= \phi_j \prod_{i \in N(j)} m_{ij} \\ b_j^*(s_n) &= (1-\phi_j) \prod_{i \in N(j)} (1-m_{ij}) \end{aligned} \tag{6}$$

Lastly, we rewrite Equation (1) as follows. First, we replace edge potential $\psi_{ij}(x_i, x_j)$ by propagation strength ϵ_{ij} as discussed in Section II-B; the difference is that we use customized strength for each edge instead of the global strength. Second, we replace the unnormalized messages $m_{ij}^*(x_i, x_j)$ by the normalized ones. Third, we replace the multiplications of incoming messages by the pre-computed beliefs to avoid duplicate calculations [15]. Resulting equations are given as Equation (7).

$$\begin{aligned} m_{ij}^*(s_p) &\leftarrow \epsilon_{ij} \frac{b_i}{m_{ji}} + (1-\epsilon_{ij}) \frac{1-b_i}{1-m_{ji}} \\ m_{ij}^*(s_n) &\leftarrow (1-\epsilon_{ij}) \frac{b_i}{m_{ji}} + \epsilon_{ij} \frac{1-b_i}{1-m_{ji}} \end{aligned} \tag{7}$$

Algorithm 2 shows a summary of the propagation step. In lines 1 to 3, we initialize the node potentials. The normalized potential ϕ is given as a parameter. Node potentials for the unobserved nodes are uniformly set to 0.5. In lines 4 to 5, we initialize all the messages and propagation strengths. In lines 6 to 9, we iteratively update the messages until they converge. In line 10, we return the computed messages and beliefs.

D. Weight Update Step

In the weight update step (line 6 in Algorithm 1), we update the weight vector w using a gradient-based approach. First, we define a cost function of w which we try to minimize. The cost function should be defined in a way that its minimization makes better classification. Second, we differentiate the cost function with respect to w and compute the gradient. Finally, we update w to the negative direction of the derivative.

The cost function $E(w)$ is defined as Equation (8), which is computed from the pairwise differences between the beliefs of positive and negative training nodes. Given an increasing loss function h, $E(w)$ is minimized as the beliefs b_n of negative nodes are minimized and the beliefs b_p of positive nodes are maximized. An L2 regularization parameter λ is introduced to avoid overfitting and decrease the model complexity.

$$E(w) = \lambda ||w||_2^2 + \sum_{p \in P_{\rm trn}} \sum_{n \in N_{\rm trn}} h(b_n - b_p) \tag{8}$$

We use the loss function $h(x) = (1+\exp(-x/d))^{-1}$ which approximates the step function when d is small. It is known that AUC (area under the ROC curve) of binary classification is maximized when h is used as a loss function and d is small enough [6], [16]. We set d to 0.0001.

Then, we differentiate the cost function as follows, where an error term $x = b_n - b_p$ is introduced to simplify the equation. $\partial h(x)/\partial x$ is easily computed since $h(x)$ is a simple sigmoid function of x. The problem is to compute the derivatives of beliefs, which is discussed in Section III-E.

$$\frac{\partial E(w)}{\partial w} = 2\lambda w + \sum_{p \in P_{\rm trn}} \sum_{n \in N_{\rm trn}} \frac{\partial h(x)}{\partial x} \left(\frac{\partial b_n}{\partial w} - \frac{\partial b_p}{\partial w} \right)$$

$$\text{where } \frac{\partial h(x)}{\partial x} = d^{-1} h(x)(1-h(x))$$

Assuming we have the derivatives of beliefs, we differentiate the cost function as $w' = \partial E/\partial w$ and update the weight vector w, as shown in Equation (9). $\min\{\cdot,\cdot\}$ selects the vector whose L2 norm is smaller than the other. New parameters α and β are introduced; α is a step size that determines quality and speed of the convergence, and β limits the L2 norm of the gradient step to avoid a steep change.

$$w \leftarrow w - \min \left\{ \alpha w', \frac{\beta}{||\alpha w'||_2} \alpha w' \right\} \tag{9}$$

Algorithm 3 shows a summary of the weight update step. In line 1, we differentiate the beliefs. In lines 2 to 6, we compute the derivative of the cost function with regularization. In line 7, we update the weight vector and return the result.

Algorithm 3: $weight\_update(\cdot)$

Input: beliefs b, messages m, and sets $P_{\rm trn}$ and $N_{\rm trn}$ of positive and negative training nodes
Output: updated weight vector w

1: $b' \leftarrow differentiate(b, m)$
2: $w' \leftarrow 2\lambda w$
3: **for** p in $P_{\rm trn}$ and n in $N_{\rm trn}$ **do**
4: $\quad h \leftarrow (1+\exp(-d^{-1}(b_n - b_p)))^{-1}$
5: $\quad w' \leftarrow w' + d^{-1} h(1-h)(b'_n - b'_p)$
6: **end for**
7: **return** $w - \max\{\alpha w', \frac{\beta}{|\alpha w'|} \alpha w'\}$

E. Computing Derivatives for Weight Update Step

Here we describe how to approximately compute the derivatives of the beliefs with regard to the weight vector (line 1 of Algorithm 3). The problem is that the beliefs are not expressed as closed-form functions of w since they are computed from the messages which converge through iterations. Thus, we use the chain rule to express $\partial b_j/\partial w$ as a function of $\partial b_j/\partial m_{ij}$

and $\partial m_{ij}/\partial w$ for each neighboring node i, and then Lemma 1 to replace $\partial b_j/\partial m_{ij}$. Equation (10) shows the result.

$$\begin{aligned}\frac{\partial b_j}{\partial w} &= \sum_{i\in N(j)} \frac{\partial b_j}{\partial m_{ij}} \frac{\partial m_{ij}}{\partial w} \\ &= \sum_{i\in N(j)} \frac{b_j(1-b_j)}{m_{ij}(1-m_{ij})} \frac{\partial m_{ij}}{\partial w}\end{aligned} \tag{10}$$

Lemma 1. *Belief b_j of node j is differentiated by an incoming message m_{ij} from a neighboring node i as*

$$\frac{\partial b_j}{\partial m_{ij}} = \frac{b_j(1-b_j)}{m_{ij}(1-m_{ij})}.$$

Proof. We differentiate Equation (6) to get the derivatives of the unnormalized beliefs $b^*_j(s_p)$ and $b^*_j(s_n)$ as

$$\frac{\partial b^*_j(s_p)}{\partial m_{ij}} = \frac{b^*_j(s_p)}{m_{ij}} \qquad \frac{\partial b^*_j(s_n)}{\partial m_{ij}} = -\frac{b^*_j(s_n)}{1-m_{ij}}.$$

Then, we differentiate the first part of Equation (2) to express $\partial b_j/\partial m_{ij}$ as a function of the above derivatives. After substituting the values, we get the equation in the lemma. □

Next, we apply the chain rule again to express the message derivative as Equation (11). This is not a closed-form solution because the derivative terms in the form of $\partial m_{ij}/\partial w$ exist in both sides of the equation. Unlike the messages which converge through iterations in most real-world networks, the derivatives are shown to diverge because of positive loops existing in a cyclic network. Thus, we introduce a recursion parameter η to limit the number of updates.

$$\frac{\partial m_{ij}}{\partial w} = \underbrace{\frac{\partial m_{ij}}{\partial \epsilon_{ij}} \frac{\partial \epsilon_{ij}}{\partial w}}_{\text{base terms}} + \sum_{k\in N(i)\setminus j} \underbrace{\frac{\partial m_{ij}}{\partial m_{ki}} \frac{\partial m_{ki}}{\partial w}}_{\text{recursive terms}} \tag{11}$$

We separate the derivatives in the right hand side of Equation (11) into the base and recursive terms. The first base term $\partial m_{ij}/\partial \epsilon_{ij}$ is computed from Lemma 2 and the second base term $\partial \epsilon_{ij}/\partial w$ is computed as $\epsilon_{ij}(1-\epsilon_{ij})\theta_{ij}$ since ϵ_{ij} is a simple sigmoid function of w. The coefficient $\partial m_{ij}/\partial m_{ki}$ of the recursive terms is computed from Lemma 3. As a result, we get Equation (12):

$$\frac{\partial m_{ij}}{\partial w} = k_1(b_i - m_{ji})\theta_{ij} + \sum_{k\in N(i)\setminus j} \frac{\epsilon_{ij} - k_2}{k_3} \frac{\partial m_{ki}}{\partial w}, \tag{12}$$

where the following nonnegative constants k_1, k_2 and k_3 are introduced to simplify the equation:

$$\begin{aligned}k_1 &= \epsilon_{ij}(1-\epsilon_{ij})(b_i + m_{ji} - 2b_i m_{ji})^{-1} \\ k_2 &= m_{ij} + m_{ji} - 2m_{ij}m_{ji} \\ k_3 &= m_{ki}(1-m_{ki})(b_i + m_{ji} - 2b_i m_{ji}) b_i^{-1}(1-b_i)^{-1}.\end{aligned}$$

Lemma 2. *Message m_{ij} of edge (i,j) is differentiated by the propagation strength ϵ_{ij} of the same edge as*

$$\frac{\partial m_{ij}}{\partial \epsilon_{ij}} = \frac{b_i - m_{ji}}{b_i + m_{ji} - 2b_i m_{ji}}.$$

Algorithm 4: *differentiate*$(\cdot)$

Input: beliefs b and messages m
Output: derivatives $\partial b/\partial w$ of the beliefs

1: **for** each edge (i,j) **do**
2: $\quad (m'_{ij})_{\text{base}} \leftarrow \frac{\epsilon_{ij}(1-\epsilon_{ij})(b_i - m_{ji})}{b_i + m_{ji} - 2b_i m_{ji}}$
3: $\quad z_{ij} \leftarrow \frac{\epsilon_{ij} - m_{ji} - m_{ij} + 2m_{ij}m_{ji}}{b_i + m_{ji} - 2b_i m_{ji}}$
4: **end for**
5: $m'_{ij} \leftarrow (m'_{ij})_{\text{base}}$ for each edge (i,j)
6: **for** η times **do**
7: $\quad m''_{ij} \leftarrow \frac{b_j(1-b_j)}{m_{ij}(1-m_{ij})} m'_{ij}$ for each edge (i,j)
8: $\quad$ **for** each edge (i,j) **do**
9: $\quad\quad (m'_{ij})_{\text{new}} \leftarrow (m'_{ij})_{\text{base}} + z_{ij} \sum_{k\in N(i)\setminus j} m''_{ki}$
10: $\quad$ **end for**
11: $\quad$ substitute m'_{ij} with $(m'_{ij})_{\text{new}}$ for each edge (i,j)
12: **end for**
13: $b'_j \leftarrow \sum_{i\in N(j)} \frac{b_j(1-b_j)}{m_{ij}(1-m_{ij})} m'_{ij}$ for each node j
14: **return** b'

Proof. We differentiate Equation (7) by ϵ_{ij} to get the derivatives of the unnormalized messages $m^*_{ij}(s_p)$ and $m^*_{ij}(s_n)$:

$$\begin{aligned}\frac{\partial m^*_{ij}(s_p)}{\partial \epsilon_{ij}} &= \frac{b_i}{m_{ji}} - \frac{1-b_i}{1-m_{ji}} \\ \frac{\partial m^*_{ij}(s_n)}{\partial \epsilon_{ij}} &= -\frac{b_i}{m_{ji}} + \frac{1-b_i}{1-m_{ji}}.\end{aligned}$$

Then, we differentiate Equation (5) to express $\partial m_{ij}/\partial \epsilon_{ij}$ as a function of the above derivatives. After substituting the values, we get the equation in the lemma. □

Lemma 3. *When nodes k and j are both neighbors of node i, message m_{ij} is differentiated by message m_{ki} as*

$$\frac{\partial m_{ij}}{\partial m_{ki}} = \frac{z_{ij} b_i (1-b_i)}{m_{ki}(1-m_{ki})},$$

where $z_{ij} = (\epsilon_{ij} - m_{ji} - m_{ij} + 2m_{ij}m_{ji})(b_i + m_{ji} - 2b_i m_{ji})^{-1}$.

Proof. We differentiate Equation (7) by m_{ki} to get the derivatives of the unnormalized messages $m^*_{ij}(s_p)$ and $m^*_{ij}(s_n)$:

$$\begin{aligned}\frac{\partial m^*_{ij}(s_p)}{\partial m_{ki}} &= \frac{\epsilon_{ij}}{m_{ji}} \frac{\partial b_i}{\partial m_{ki}} - \frac{1-\epsilon_{ij}}{1-m_{ji}} \frac{\partial b_i}{\partial m_{ki}} \\ &= \frac{\epsilon_{ij} - m_{ji}}{m_{ji}(1-m_{ji})} \frac{b_i(1-b_i)}{m_{ki}(1-m_{ki})} \\ \frac{\partial m^*_{ij}(s_n)}{\partial m_{ki}} &= \frac{1-\epsilon_{ij} - m_{ji}}{m_{ji}(1-m_{ji})} \frac{b_i(1-b_i)}{m_{ki}(1-m_{ki})}\end{aligned}$$

Then, we differentiate Equation (5) to express $\partial m_{ij}/\partial m_{ki}$ as a function of the above derivatives. After substituting the values, we get the equation in the lemma. □

Algorithm 4 shows a summary of the differentiation process. In lines 1 to 4, we compute the base terms in Equation (12) and the coefficient z_{ij} in Lemma 3. In line 5, we initialize the message derivatives using the base terms. In lines 6 to 12, we iteratively update the message derivatives η times. Specifically, in line 7, we compute the recursive terms in Equation (12) to avoid duplicate computations, and in lines 8 to 11, we update

all the message derivatives. In lines 13 to 14, we compute the belief derivatives using the computed message derivatives.

F. Scalability

We show that time and space complexities of SBP are linear with the number of edges in a given network. The complexities are given by Lemmas 4 and 5. We have two variables T_1 and T_2 representing the numbers of iterations. T_1 is the number of iterations for the propagation step and is relatively small; it does not exceed 10 for all the networks we use in the experiments. T_2 is the number of weight updates and varies from 20 to 100 depending on the network. Assuming the number of training nodes are fixed, the term $|\theta||P_{\text{trn}}||N_{\text{trn}}|T_2$ added in the time complexity can be considered as a constant.

Lemma 4. *Space complexity of SBP is $O(|\theta||E|)$ where $|\theta|$ is the number features and $|E|$ is the number of edges.*

Proof. The largest variables SBP needs to store are the message derivatives. Since every edge has two derivatives of size $|\theta|$ when comparing the old and new, the space complexity is proportional to the number of edges and features. □

Lemma 5. *Time complexity of SBP is given as*

$$O(((T_1 + \eta|\theta|)|E| + |\theta||P_{\text{trn}}||N_{\text{trn}}|)T_2),$$

where $|\theta|$ is the number of features, $|E|$ is the number of edges, T_1 is the number of iterations for the propagation step, η is the number of derivative updates for the update step, $|P_{\text{trn}}|$ and $|N_{\text{trn}}|$ are the number of positive and negative training nodes, respectively, and T_2 is the number of weight updates.

Proof. Time complexity of the propagation step is $O((|\theta| + T_1)|E|)$ since all the messages are updated T_1 times. Time complexity of the update step is $O(|\theta|(\eta|E| + |P_{\text{trn}}||N_{\text{trn}}|))$ since the running time mostly depends on the computations of message derivatives. We prove the lemma by summing them up and multiplying T_2 as SBP alternates them T_2 times. □

IV. Experiments

In this section, we present experimental results of SBP to answer the following questions:

- **Q1. Accuracy (Section IV-B).** How accurately does SBP classify nodes in attributed networks?
- **Q2. Learning process (Section IV-C).** How accurately does SBP find the optimal weights? How do values of the cost function and AUC change during iterations?
- **Q3. Scalability (Section IV-D).** How does running time of SBP scale with regard to the size of a network?

A. Experimental Settings

1) Datasets: We use three publicly available datasets which are summarized in Table IV. Epinions-R [17] is a heterogeneous network which we introduced in Section II-B. It contains two kinds of nodes (users and items) and edges (reviews and trusts). The review edges connect users and items with ratings between 1 and 5, while the trust edges connect only users with no additional information. Epinions-S [8] is a signed social network whose edges are either positive or negative. The sign of an edge (u, v) indicates either a positive or negative feeling of user u towards user v. MovieLens [18] is a bipartite review network for movies from the users of MovieLens, whose edges contain integer ratings between 1 and 5.

TABLE IV: Summary of the datasets.

Dataset	Nodes	Edges	Attributes
Epinions-R[2]	189,028	1,152,005	ratings and trusts
Epinions-S[3]	131,828	841,372	signs (trusts or distrusts)
MovieLens[4]	9,940	1,000,209	ratings (1 to 5)

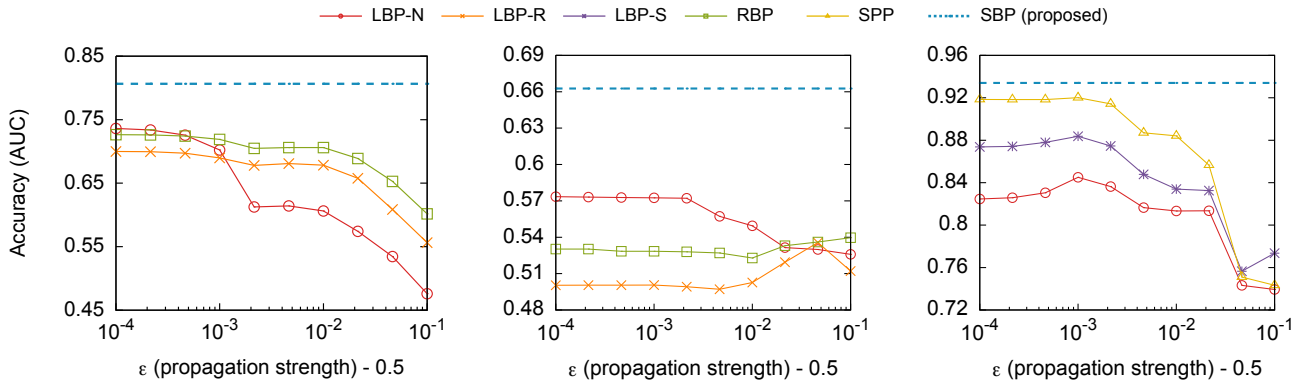
2) Encoding: We model the networks based on one-hot encoded feature vectors to make SBP learn independent weights for the feature elements. The simplest network is Epinions-S whose attributes (signs) are easily modeled as one-hot encoded vectors of length 2. On the other hand, we have two factors to consider in MovieLens: directions and ratings. We ignore the directions of edges since the network is bipartite; the edges represent undirected relationships between users and movies. Then, we encode the ratings as one-hot encoded vectors of length 5. Epinions-R is modeled as described in Section III-B. The ratings are modeled as the same in MovieLens and an additional element representing the edge type is added to the vectors. As a result, the feature vectors are of length 6.

3) Competitors: We compare SBP with node classification methods which are based on Loopy Belief Propagation (LBP) and Random Walk with Restart (RWR). LBP is an inference algorithm introduced in Section II, which is the basis of SBP. RWR [19] is an algorithm to compute node relevance. Starting from a seed node, it walks through other nodes and jumps back to the seed node with a certain probability. As a result, nodes near the seed node are likely to have high visiting probabilities compared to the ones far way from the seed node. Since LBP and RWR are both based on the notion of guilt-by-association, they share common characteristics [20].

RBP [9] and SPP [13] are LBP-based methods for review and signed networks, respectively. RBP uses the rating information to assign different node potentials and neglects edges with low ratings to ensure homophily relationships between the nodes. SPP uses two edge potential tables to model different propagation strengths of positive and negative edges. M-RWR [21] is an RWR-based method for signed networks, which runs RWR for each subgraph consisting of only positive or negative edges, and then subtracts the probabilities.

Furthermore, we use various LBP settings as the baselines of SBP. We use three baseline methods LBP-N, LBP-S, and LBP-R whose strategies of determining the propagation strengths are chosen heuristically. LBP-N is the simplest method which assumes a uniform propagation strength by ignoring the attributes. LBP-S works for Epinions-S; it assumes propagation strengths of ϵ and $\epsilon/2$ for the positive and negative edges, respectively. LBP-R works for the review networks; it models a linear scaling of propagation strengths of the review edges.

[2]http://www.trustlet.org/downloaded_epinions.html
[3]http://www.trustlet.org/extended_epinions.html
[4]http://grouplens.org/datasets/movielens/1m



(a) MovieLens (b) Epinions-R (c) Epinions-S

Fig. 4: AUCs of the LBP-based methods for varying values of the propagation strength. Each plot illustrates the result for each dataset: (a) MovieLens, (b): Epinions-R, and (c) Epinions-S. The existing methods are shown to be sensitive to the propagation strength. SBP, our proposed method denoted by the dashed blue lines, provides the highest accuracies for all the datasets.

In other words, it assumes the propagation strength of $r(\epsilon/5)$ for the edges with rating r.

4) Experimental Process: Since the nodes in the networks are not initially labeled, we pick seed nodes for each network and then label other nodes based on each seed node. Given a seed node u, we label the others as follows: for MovieLens and Epinions-R, items that received ratings 5 from node u are positive, and items that received ratings below 5 are negative [9]. For Epinions-S, users that received the trust edges from node u are positive, and users that received the distrust edges are negative. Labels are not defined for the nodes which are not connected to the seed node. We uniformly set node potential ϕ to 0.55 in all the LBP-based methods, except RBP which uses its own node potential values, since they generally give the best accuracies for the datasets. In other words, we set $\phi_i(s_p)$ and $\phi_i(s_n)$ of node i to 0.55 and 0.45, respectively. In SBP, we set the number η of recursive updates to 1 because the results are shown to be insensitive to η.

5) Evaluation: We use three kinds of evaluation metrics: the area under the ROC curve (AUC), mean average precision (MAP), and precision at k (P@k). They are computed from the beliefs and visiting probabilities of the test nodes whose labels are hidden. AUC is generated from a result of binary classification by plotting the true positive rates against the false positive rates at various threshold settings. MAP is the mean of the average precision for each query, where a query represents a seed user in our experiments. P@k is the ratio of true positive nodes from the top-k ranked nodes. We set k to 20.

B. Accuracy (Q1)

We randomly pick k_1 seed nodes for each network, run all the methods for each seed node, and average the accuracies. Each seed node is picked from the nodes connected to at least k_2 positive and k_2 negative nodes since the propagation does not work well when the number of observed nodes is too small. Then, we do the following for each seed node: we 1) randomly sample k_2 nodes from each set of positive and negative nodes to balance the number of nodes in both sets, 2) divide each set of sampled nodes into training and test, 3) run the algorithms using only the training sets, and 4) compute the classification accuracies for the test sets. We set k_1 to 20 and k_2 to 80.

Figure 4 shows accuracies of the LBP-based methods for varying values of the propagation strength. Values of ϵ equal to or less than 0.5 are not included in the experiment since they violate the assumption of guilt-by-association. As seen in the figure, accuracies of the existing methods significantly change depending on its value. On the other hand, SBP, our proposed method which automatically learns the propagation strengths, shows higher accuracies than the others even when compared to their optimal performances. This is because SBP learns different propagation strengths of the edges while the others neglect the attributes or depend on the heuristically determined strengths.

Figure 1 shows overall results comparing SBP to the other methods in three evaluation metrics. LBP represents the best baseline method for each dataset; LBP-N for MovieLens and Epinions-R, and LBP-S for Epinions-S. SBP shows the highest accuracies for all the datasets. Since performances of the LBP-based methods depend on the values of the propagation strength, we use the best ones found in the experiments of Figure 4 for a fair comparison. Note that different groups of competing methods are used for each network; SPP and M-RWR are used only for Epinions-S, and RBP is used only for Epinions-R and MovieLens. This is because they are designed for specific types of networks, not general ones.

C. Learning Process (Q2)

Figures 5 and 6 show the iterative process of SBP learning the optimal weight vector w in the experiment of MovieLens. Figure 5 shows that the elements of w converge through the iterations starting from the initial values of 0.001, although it is not theoretically guaranteed. The kth element of w, denoted by w_k, represents the propagation strength of the edges with rating k since the feature vectors in the network are one-hot encoded. Figure 6 shows the values of the cost function $E(w)$ and AUC during the same iterations. The costs are minimized

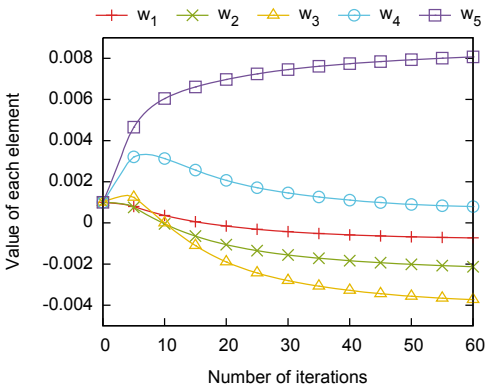


Fig. 5: Changing values of the weight elements in w during the iterations for MovieLens. The weights w_4 and w_5 for high ratings increase above zero, while the others decrease below zero. All the weights are initialized to 0.0001.

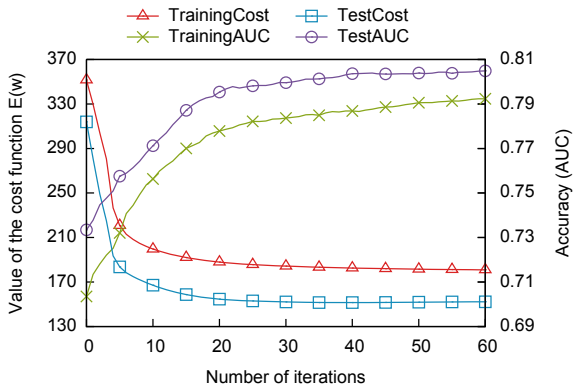


Fig. 6: Changing values of the cost function $E(w)$ and AUC during the iterations for MovieLens. The cost keeps decreasing as the weights are updated, resulting in increased AUCs for both training and test sets.

for both training and test sets as the weights are updated, resulting in increased AUCs.

We note the following observations in Figure 5. First, the elements show different patterns as the iteration proceeds; w_3 and w_4 rapidly increase at first, and then slowly decrease until the convergence. It shows that the weights are updated toward a local optimum. Second, the optimal weights for low ratings (from 1 to 3) are sorted in the reverse order; w_1 converges to a higher value than w_3. This is because the ratings in the dataset are biased toward high ratings; the average of all the ratings is 3.58, and the number of rating 3 is greater than the sum of the numbers of ratings 1 and 2. Thus, rating 3 possibly represents a lower evaluation than ratings 1 and 2 in this dataset, and the result shows that SBP is capable of learning dataset-specific relationships without any given prior knowledge.

D. Scalability (Q3)

We measure running time of SBP for various networks to verify the linear scalability proved in Lemma 5. To generate smaller networks with similar characteristics, we sample principal submatrices from the adjacency matrix of each dataset. For consistency, we randomly choose one seed node for each dataset and use it for all the subgraphs, and fix the number of weight updates (T_2 in Lemma 5) to 40. A laptop with 2.2GHz Intel Core i7 processors is used to measure the running time. Figure 7 shows the result; running time of SBP scales linearly with regard to the number of edges for all the datasets. SBP takes the longest in MovieLens since it is 15$\times$ more dense than the other networks; it takes more iterations for the messages to converge (T_1 in Lemma 5).

V. Related Works

In this section, we review related works which are categorized into three parts: Loopy Belief Propagation (LBP), real-world applications of LBP, and node classification methods based on Random Walk with Restart (RWR).

1) Loopy Belief Propagation: Yedidia et al. showed that LBP can be applied to various probabilistic graphical models without loosing generality [10]. Gonzalez et al. and Elidan et al. increased speed of the convergence in LBP by updating the messages in an asynchronous way [22], [23]. Chechetka et al. concentrated the computation of messages on more important areas of a real-world graph for faster convergence [24]. Kang et al. introduced a distributed LBP algorithm on MapReduce [25], [15]. Koutra et al. showed that three guilt-by-association methods including LBP, RWR, and Semi-supervised learning lead to a similar matrix inversion problem [20].

2) Real-world applications of LBP: Pandit et al. and Chau et al. applied LBP in detecting fraudulent users in online auction networks [1], [4]. Akoglu et al. proposed an algorithm that extends LBP to a signed network in order to spot fraudsters and fraudulent reviews from online review networks [26]. Jang et al. and Akoglu also applied LBP in signed networks with different propagation strengths [13], [27]. Chau et al. and Tamersoy et al. applied LBP to large scale malware detection problems [2], [11]. McGLohon et al. applied LBP to graph labeling and risk detection problems [3]. Ayday et al. and Ha et al. applied LBP to recommendation problems [9], [28]. Felzenszwalb et al. and Yang et al. applied max-product LBP to computer vision problems [12], [29]. Koutra et al. used a variant of LBP to compute the node affinities of two graphs required to measure the graph similarity [30].

3) RWR-based methods: Haveliwala proposed RWR, an algorithm to measure relevance between nodes based on random walks [19]. Backstrom and Leskovec proposed a supervised algorithm based on RWR which determines transition probabilities as a function of node and edge features [6]. Shahriari and Jalili proposed M-RWR to solve sign prediction problems [21]. Jung et al. proposed an RWR-based algorithm on signed social networks [31].

VI. Conclusion

We propose Supervised Belief Propagation (SBP), a novel and scalable graph inference algorithm for general attributed

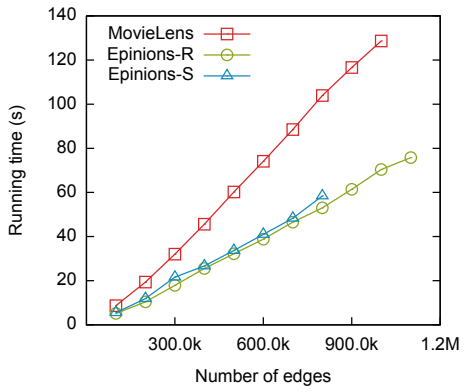


Fig. 7: Running time of SBP for the generated subgraphs. SBP shows linear scalability with respect to the numbers of edges for all three datasets we use in the experiments.

networks. SBP assigns different edge potentials based on the attributes of each edge by learning the optimal weights for the attributes. As a consequence, SBP generalizes existing methods with higher accuracies on various real-world networks. Experimental results show that SBP brings up to 15.6% higher AUC on node classification problem compared to the best existing methods. SBP enjoys linear scalability with the number of edges for both time and space complexities. Future research directions include extending SBP for distributed computing environments and learning the node potentials as well.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT and Future Planning(NRF-2015K1A3A1A14021055). U Kang is the corresponding author.

REFERENCES

[1] D. H. Chau, S. Pandit, and C. Faloutsos, "Detecting fraudulent personalities in networks of online auctioneers," in *Knowledge Discovery in Databases: PKDD 2006*. Springer, 2006, pp. 103–114.

[2] D. H. P. Chau, C. Nachenberg, J. Wilhelm, A. Wright, and C. Faloutsos, "Polonium: Tera-Scale Graph Mining and Inference for Malware Detection," in *Proceedings of the 2011 SIAM International Conference on Data Mining*. Philadelphia, PA: Society for Industrial and Applied Mathematics, Dec. 2013, pp. 131–142.

[3] 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 *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 1265–1274.

[4] S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos, "Netprobe: a fast and scalable system for fraud detection in online auction networks," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 201–210.

[5] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social networks*, vol. 25, no. 3, pp. 211–230, 2003.

[6] L. Backstrom and J. Leskovec, "Supervised random walks: predicting and recommending links in social networks," in *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 2011, pp. 635–644.

[7] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM (JACM)*, vol. 46, no. 5, pp. 604–632, 1999.

[8] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in *Proceedings of the 19th international conference on World wide web*, 2010.

[9] J. Ha, S.-H. Kwon, S.-W. Kim, C. Faloutsos, and S. Park, "Top-N recommendation through belief propagation," in *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012, pp. 2343–2346.

[10] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Understanding belief propagation and its generalizations," *Exploring artificial intelligence in the new millennium*, vol. 8, pp. 236–239, 2003.

[11] Tamersoy, Acar, Roundy, Kevin A, and Chau, Duen Horng, "Guilt by association - large scale malware detection by mining file-relation graphs." *KDD*, pp. 1524–1533, 2014.

[12] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," *International journal of computer vision*, vol. 70, no. 1, pp. 41–54, 2006.

[13] L. Akoglu, "Quantifying Political Polarity Based on Bipartite Opinion Networks." *ICWSM*, 2014.

[14] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[15] U. Kang, D. H. Chau, and C. Faloutsos, "Mining large graphs: Algorithms, inference, and discoveries," in *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*. IEEE, 2011, pp. 243–254.

[16] L. Yan, R. H. Dodier, M. Mozer, and R. H. Wolniewicz, "Optimizing Classifier Performance via an Approximation to the Wilcoxon-Mann-Whitney Statistic." *ICML*, 2003.

[17] P. Massa and P. Avesani, "Trust-aware recommender systems," in *Proceedings of the 2007 ACM conference on Recommender systems*. ACM, 2007, pp. 17–24.

[18] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 5, no. 4, p. 19, 2016.

[19] T. H. Haveliwala, "Topic-sensitive pagerank," in *Proceedings of the 11th international conference on World Wide Web*. ACM, 2002, pp. 517–526.

[20] 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." *ECML/PKDD*, 2011.

[21] M. Shahriari and M. Jalili, "Ranking nodes in signed social networks," *Social Network Analysis and Mining*, vol. 4, no. 1, pp. 1–12, 2014.

[22] J. Gonzalez, Y. Low, and C. Guestrin, "Residual splash for optimally parallelizing belief propagation," in *International Conference on Artificial Intelligence and Statistics*, 2009, pp. 177–184.

[23] G. Elidan, I. McGraw, and D. Koller, "Residual belief propagation: Informed scheduling for asynchronous message passing," *arXiv preprint arXiv:1206.6837*, 2012.

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

[25] U. Kang, D. Chau, and C. Faloutsos, "Inference of beliefs on billion-scale graphs," *The 2nd Workshop on Large-scale Data Mining: Theory and Applications*, 2010.

[26] L. Akoglu, R. Chandy, and C. Faloutsos, "Opinion fraud detection in online reviews by network effects." *ICWSM*, vol. 13, pp. 2–11, 2013.

[27] M.-H. Jang, C. Faloutsos, S.-W. Kim, U. Kang, and J. Ha, "Pin-trust: Fast trust propagation exploiting positive, implicit, and negative information," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2016, pp. 629–638.

[28] E. Ayday and F. Fekri, "A belief propagation based recommender system for online services," in *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010, pp. 217–220.

[29] Q. Yang, L. Wang, and N. Ahuja, "A constant-space belief propagation algorithm for stereo matching," in *Computer vision and pattern recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 1458–1465.

[30] D. Koutra, J. T. Vogelstein, and C. Faloutsos, "Deltacon: A principled massive-graph similarity function," in *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 2013, pp. 162–170.

[31] J. Jung, W. Jin, L. Sael, and U. Kang, "Personalized ranking in signed networks using signed random walk with restart," in *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 2016, pp. 973–978.