Belief Propagation Network *for Hard Inductive Semi-Supervised Learning*

Data Mining Laboratory
Seoul National University

Jaemin Yoo
Seoul National Univ.
jaeminyoo@snu.ac.kr

Hyunsik Jeon
Seoul National Univ.
jeon185@snu.ac.kr

U Kang
Seoul National Univ.
ukang@snu.ac.kr

IJCAI 2019 MACAO

Summary

- ***Given:*** Graph-structured data with partially observed labels
- ***Problem:*** Train a robust classifier in a semi-supervised setting that works independently for each node without neighbors
- ***Main idea:*** Propose a ***belief propagation network (BPN)***, which uses a classifier to compute the priors of nodes and then diffuse them through the graph, independently from the priors
- ***Homepage (+ source codes):*** *https://datalab.snu.ac.kr/bpn*

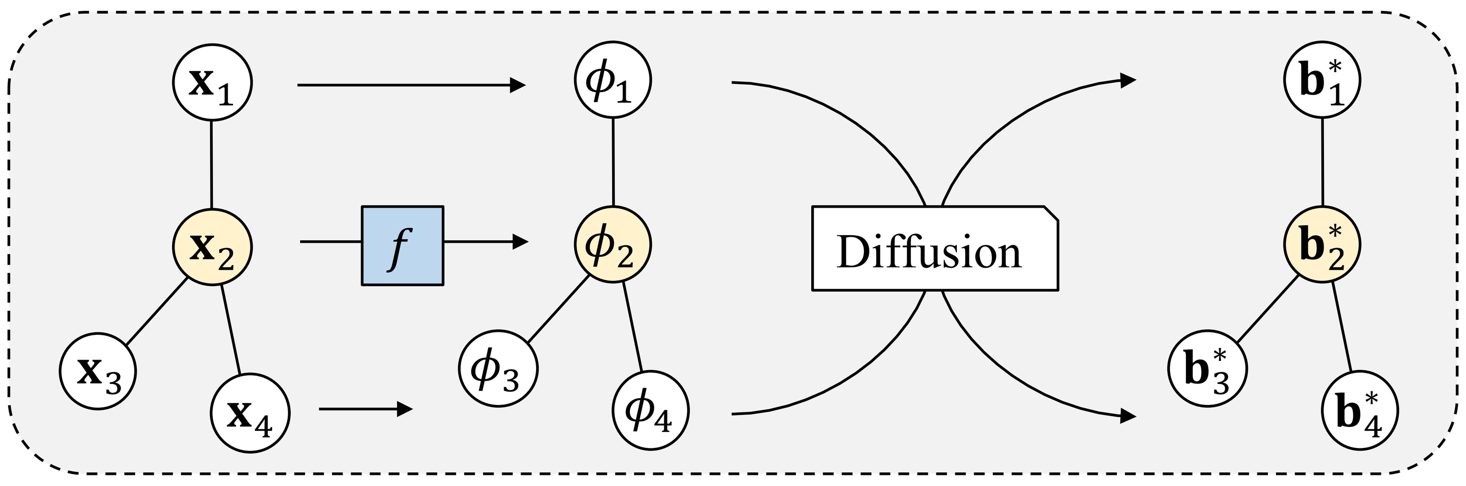
Problem Definition

- ***Given:*** a graph $G = (\mathcal{V}, \mathcal{E})$
- ***Given:*** feature vector $\mathbf{x}_i$ for every node $i \in V$
- ***Given:*** label $y_i \in \mathcal{S}$ of node $i \in V_o$ where $V_o \subset V$
- ***Train:*** a classifier $f: \mathbf{x} \to y$
- ***Semi-supervised learning:*** $|V_o| \ll |V|$
- ***Inductive learning:*** *f should work with data unseen at training*
- ***Hard inductive learning:*** *f should work well without a graph*

Belief Propagation Network (BPN)

- ***Idea 1:*** Separate the classification of each node and diffusion of the predictions (or priors) over the graph
- ***Idea 2:*** Model the diffusion as a parameter-free operation
- ***Idea 3:*** Use an *induction loss* to make the classifier mimic the diffusion by its own structure, without an actual graph
- **Overview:** BPN is an algorithm to train a given classifier f by diffusing its predictions over the given graph

Forward Propagation



1. *Prior Computation*

Use a classifier f to compute the prior ϕ_i of node i

$$\phi_i = f(\mathbf{x}_i; \theta)$$

- $\mathbf{x}_i$ is the feature vector of node i
- θ is the set of parameters of the classifier f
- ϕ_i is a probability vector that sums to one

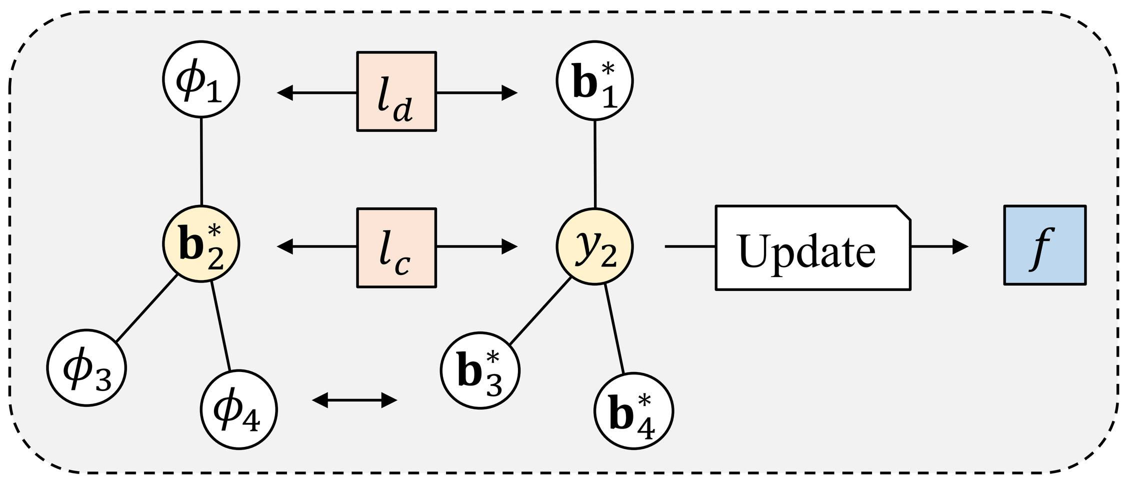
2. *Diffusion by Loopy Belief Propagation*

Diffuse the predicted priors by loopy belief propagation

$$\mathbf{m}_{ij}^t = [\psi(\mathbf{b}_i^{t-1} \oslash \mathbf{m}_{ji}^{t-1})]$$
$$\mathbf{b}_j^t = \mathrm{softmax}\left(\log \phi_j + \textstyle\sum_{i \in \mathcal{N}_j} \log \mathbf{m}_{ij}^t\right)$$

- ψ is an edge potential matrix that imposes adjacent nodes to have the same label (correlation by the graph structure)
- $\mathbf{m}_{ij}^t$ is a message from node i to node j at iteration t
- $\mathbf{b}_j^t$ is a belief (diffused prediction) of node j at iteration t

Backward Propagation



1. *Classification Loss*

How well the observed labels are predicted by the beliefs

$$l_c(\theta) = -\textstyle\sum_{i \in \mathcal{V}_o} \log b_i^*(y_i)$$

2. *Induction Loss*

How well the diffused beliefs are predicted by the priors

$$l_d(\theta) = -\textstyle\sum_{i \notin \mathcal{V}_o} \sum_{s \in \mathcal{S}} b_i^*(s)(\log \phi_i(s) - \log b_i^*(s))$$

3. *Overall Loss Function*

Combine the two losses by a hyperparameter β

$$l(\theta) = (1 - \beta) l_c(\theta) + \beta l_d(\theta) + \lambda \|\theta\|_2^2$$

- β is set to a value between 0 and 1 (0.5 in experiments)
- λ is an L2 regularization parameter for the training

Experiments

- ***Classification accuracy (vs. competitors)***
  - *Planetoid (Yang et al., 2016)*
  - *GCN (Graph convolutional networks, Kipf et al., 2017)*
  - *SEANO (Liang et al., 2018)*
  - *GAT (Graph attention networks, Velickovic et al.,, 2018)*

Method	Pubmed	Cora	Citeseer	Amazon
Planetoid	74.6 ± 0.5	66.2 ± 0.9	66.8 ± 1.0	70.1 ± 1.9
GCN-I	74.1 ± 0.2	67.8 ± 0.6	63.6 ± 0.5	76.5 ± 0.3
SEANO	75.7 ± 0.4	64.5 ± 1.2	66.3 ± 0.8	78.6 ± 0.6
GAT	76.5 ± 0.4	70.1 ± 1.0	66.7 ± 1.0	77.5 ± 0.4
BPN (ours)	**78.3** $\pm$ **0.3**	**72.2** $\pm$ **0.5**	**70.1** $\pm$ **0.9**	**81.5** $\pm$ **1.3**

- ***Training process (loss values)***
  - Classification loss decreases continuously
  - Induction loss increases at first then decreases

This is because f's updates change the beliefs of nodes!