UNIVERSITY OF HELSINKI

https://helda.helsinki.fi

Joint Use of Node Attributes and Proximity for Node Classification

Merchant, Arpit; Mathioudakis, Michael

Benito, R.M.; Cherifi, C.; Cherifi, H.; Moro, E.; Rocha, L.M. ...

2022-01-01

http://hdl.handle.net/10138/341566

Merchant, A & Mathioudakis, M 2022, Joint Use of Node Attributes and Proximity for Node Classification. in R M Benito, C Cherifi, H Cherifi, E Moro, L M Rocha & M Sales-Pardo (eds), COMPLEX NETWORKS & THEIR APPLICATIONS X, VOL 2 : Complex Networks & Their Applications X. Studies in Computational Intelligence, vol. 1016, Springer, Cham, pp. 511-522, The International Conference on Complex Networks and their Applications (2021), Madrid, Spain, 30/11/2021. https://doi.org/10.1007/978-3-030-93413-2_43

Downloaded from Helda, University of Helsinki institutional repository. https://helda.helsinki.fi
This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.
Please cite the original version.

Joint Use of Node Attributes and Proximity for Node Classification

Arpit Merchant and Michael Mathioudakis

University of Helsinki, Helsinki, Finland
firstname.lastname@helsinki.fi

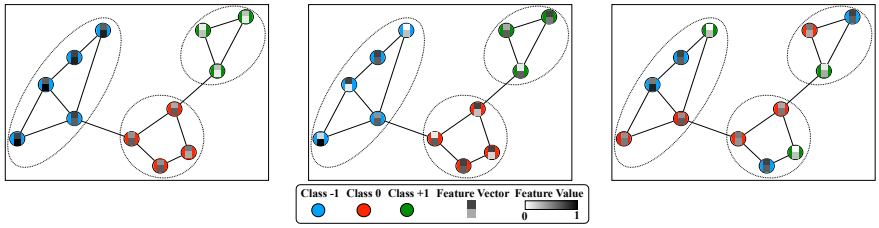
Abstract. The task of node classification is to infer unknown node labels, given the labels for some of the nodes along with the network structure and other node attributes. Typically, approaches for this task assume homophily, whereby neighboring nodes have similar attributes and a node's label can be predicted from the labels of its neighbors or other proximate (i.e., nearby) nodes in the network. However, such an assumption may not always hold – in fact, there are cases where labels are better predicted from the individual attributes of each node rather than the labels of its proximate nodes. Ideally, node classification methods should flexibly adapt to a range of settings wherein unknown labels are predicted either from labels of proximate nodes, or individual node attributes, or partly both. In this paper, we propose a principled approach, JANE, based on a generative probabilistic model that jointly weighs the role of attributes and node proximity via embeddings in predicting labels. Our experiments on a variety of network datasets demonstrate that JANE exhibits the desired combination of versatility and competitive performance compared to standard baselines.

Keywords: node classification, graph embeddings, spectral graph analysis

1 Introduction

Semi-supervised node classification over attributed graphs involves a graph with known structure, where each node is associated with a label (or 'class', a categorical variable), as well as other attributes. However, labels are known only for some of the nodes and are unknown for others. Given the graph structure, the attributes of all nodes, and the labels of some of the nodes, the goal is to predict the labels for the remaining nodes. This task finds application in many domains such as information networks [22], complex systems [31], and protein function identification [6].

Previous Work. Motivated by theories of homophily [16] and social influence [15], a common assumption is that adjacent nodes tend to have similar labels. For instance, in a social network, friends are likely to vote for the same political party. Approaches that rely on this assumption typically enforce homophily by considering node proximity and assigning the same label to nearby nodes. In label



(a) Nodes with similar labels are adjacent to each other and have similar features.

(b) Nodes with similar labels are adjacent to each other but have different features.

(c) Nodes with similar labels have similar features but are not adjacent to each other.

Fig. 1: Node labels may correlate with either graph structure (1b), or node attributes(1c), or both (1a). GCNs would perform well in the case of Fig. 1a, 1b, but not in Fig. 1c because they predict the label of a node by aggregating features of adjacent nodes. Conversely, attribute-based models would perform well in the case of Fig 1a, 1c, but not in Fig. 1b because they do not exploit the correlation between adjacent nodes and their labels.

propagation, for example, labels diffuse from labeled nodes to their unlabeled neighbors in an iterative manner until convergence [30]. Other methods induce label uniformity within cuts or clusters of the graph [3,10,2]; or consider node proximity in a latent space that preserves graph distances, as in DeepWalk [18] and similar matrix factorization approaches [19].

However, the aforementioned methods ignore other node attributes, which can be detrimental. For example, Hamilton et al. [6] show that, for certain predictive tasks on citation and social graphs, a linear classifier that is built only on node attributes outperforms approaches such as DeepWalk that are based on node proximity but ignore node attributes. Moreover, while homophily is often observed in some classification tasks, it is not uncommon to find that adjacent nodes do not share a particular label and that, in such cases, other node attributes can serve as better label predictors than graph structure [14,21]. For example, two individuals may be friends (i.e., be connected on a social network) but vote for different political parties ('label') – something that could be better predicted by rich, node-level attribute data (e.g., geographic location, income, or profession). Therefore, it is important to appropriately leverage both the graph structure and node attributes, for label prediction.

Partially addressing this limitation, AANE [8] and DANE [5] combine low-dimensional encodings of node attributes with graph-distance-preserving node embeddings, and use them as input features for label prediction. However, they do not account for known labels during training, thus potentially ignoring information that would be useful in predicting the unknown labels. LANE [9] overcomes this limitation, by learning a the joint latent representations of node attributes, proximity, and labels. However, LANE does not directly address the node classification task, i.e., it does not optimize the conditional probability

distribution of node labels given the node attributes and graph structure, but rather targets their joint distribution of all quantities.

Separately, graph convolutional networks use graph topology for low-pass filtering on node features [27]. GAT [24] introduce an attention mechanism to learn weights and aggregate features. GraphSAGE [6] use mean/max pooling to sample and aggregate features from nodes' local neighbourhoods. However, these convolutions are equivalent to repeated smoothing over the node attributes and performance quickly degrades [13]. Subsequent approaches such as DiffPool [29] have sought to address this limitation, but these too aggressively enforce homophily and require that nodes with the same labels have similar graph and attribute representations. Recent work by AM-GCN [26] attempts to weaken this assumption by analyzing the fusing capabilities of convolutional models. They define two modules, one each for the topology space and feature space, and adaptively combine them using an attention mechanism.

Our contribution. In this work, we develop an approach to node classification that is able to flexibly adapt to and perform consistently well in a range of settings, from cases where node labels exhibit strong homophily (i.e., a node's label can be determined by the labels of its neighbors) to cases where labels are independent of graph structure and solely determined by other node attributes, as well as cases that lie between these two extremes. We propose a novel and principled approach to node classification based on a generative probabilistic model that jointly captures the role of graph structure and node similarity in predicting labels. Our analysis leads to JANE, an algorithm that identifies a maximum-likelihood estimate of the model that can be employed to perform node classification. Unlike aforementioned approaches that are heavily based on label homophily (e.g., label propagation), we account not only for the graph structure but also for node attributes, and flexibly weigh each of them appropriately depending on the case. Moreover, unlike AANE [8] and DANE [5], our approach learns a low-dimensional node representation informed by labels, that is then used for prediction. And unlike LANE [9], we directly optimize the conditional probability of labels given the graph structure and node attributes without necessarily enforcing homophily.

We summarize our main contributions below:

- Define a formal generative framework tailored to node classification, that captures the relationship between graph structure, node attributes, and node labels.
- Develop JANE, an algorithm for semi-supervised node classification, based on maximum likelihood estimate (MLE) of the model parameters.
- Demonstrate shortcomings of existing approaches and versatility of JANE on synthetic datasets.
- Empirically validate JANE's performance on benchmark, real datasets compared to standard baselines.

2 Problem Setting

Let us consider an undirected and connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of node size $|\mathcal{V}| = n$. Let its structure be represented by the adjacency matrix $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{n \times n}$. Denote $\mathbf{D} = \text{diag}\,(d_1, d_2, \ldots, d_n)$ to be the degree matrix where $d_i = \sum_j a_{ij}$; and $\mathbf{L} = \mathbf{D} - \mathbf{A}$ as its unnormalized Laplacian matrix. Let $\lambda_i\,(\mathbf{L})$ be the i-th smallest eigenvalue of $\mathbf{L}$ and $\mathbf{e}_i\,(\mathbf{L})$ its corresponding eigenvector.

Each node in the graph is associated with the following: d observed attributes $\mathbf{x} \in \mathbb{R}^d$, k latent attributes $\mathbf{u} \in \mathbb{R}^k$, and one (1) possibly unobserved categorical variable $\mathbf{y} \in \mathbf{M}$ as label from label-set $\mathbf{M} = \{1, 2, \ldots, M\}$. For example, in citation graphs, with nodes corresponding to articles and edges to citations between articles, $\mathbf{x}$ capture observed quantities such as the bag-of-words representation of the article text; and label $\mathbf{y}$ denotes the research area of the article (e.g., 'data mining' or 'machine learning'). The latent attributes $\mathbf{u}$ correspond to properties of the articles that are not captured directly by attributes $\mathbf{x}$ or label $\mathbf{y}$, but that could play a role in determining which articles are connected with a citation (as captured by adjacency matrix $\mathbf{A}$) and what research area $\mathbf{y}$ an article is deemed to belong to. In terms of notation, to refer to the attributes of all nodes, we write $\mathbf{X} = \left\{\mathbf{x}_i \in \mathbb{R}^d, i \in \{1, \ldots, n\}\right\}$ to denote the observed node attributes, $\mathbf{U} = \left\{\mathbf{u}_i \in \mathbb{R}^k, i \in \{1, \ldots, n\}\right\}$ for the latent node attributes, and $\mathbf{Y} = \{\mathbf{y}_i \in \mathbf{M}, i \in \{1, \ldots, n\}\}$ for the node labels.

Having defined all elements in our setting, we now define the task we address as Problem 1.

Problem 1 (Node-Classification). Given adjacency matrix $\mathbf{A}$, node features $\mathbf{X}$, and labels $\mathbf{Y}_L$ for a subset $L \subseteq \mathcal{V}$ of nodes, predict labels $\mathbf{Y}_{\mathcal{V}-L}$ for the remaining nodes $\mathcal{V} - L$ in the graph.

3 Our Approach

Our approach for Problem 1 is based on a probabilistic generative model (described in Section 3.1) and its analysis (Section 3.2).

3.1 Model.

Figure 2 pictorially illustrates our generative model. First, the adjacency matrix $\mathbf{A}$ of the graph is generated from the latent attributes $\mathbf{U}$. Specifically, the probability that there exists an edge between two nodes i and j is given by the inverse exponent of the squares $l2$-distance between their latent attributes $\mathbf{u}$ scaled by a factor s^2 that is set as a hyperparameter.

$$\Pr\left[\ (i,j) \in \mathcal{E} \mid \mathbf{u}_i, \mathbf{u}_j\ \right] = p_{ij} = e^{\frac{-\|u_i - u_j\|^2}{s^2}} \tag{1}$$

This equips our model with the desirable property, common in many types of graph embeddings, that the closer the two nodes are in the Euclidean space of $\mathbf{U}$, the higher the likelihood that they are connected in the graph – and vice versa.

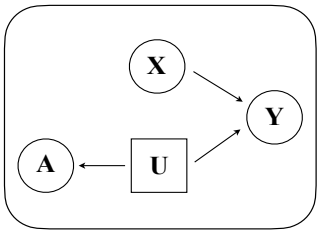


Fig. 2: Visual illustration of the generative framework. Observed node attributes $\mathbf{X}$ (represented by circular box) and latent attributes $\mathbf{U}$ (represented by square box) jointly generate node labels $\mathbf{Y}$. The (observed) adjacency matrix $\mathbf{A}$ is generated from $\mathbf{U}$ and indirectly correlates with $\mathbf{Y}$ (via $\mathbf{U}$).

Therefore, $\mathbf{U}$ represents a low-dimensional Euclidean embedding of the graph that preserves connectivity in the form of Eq. 1. Moreover, since the existence of an edge is independent across pairs of nodes in this model, we have

$$\Pr\left[\mathbf{A}|\mathbf{U}\right] = \prod_{(i,j)\in\mathcal{E}} p_{ij} \times \prod_{(i,j)\notin\mathcal{E}} \left(1 - p_{ij}\right). \tag{2}$$

Second, node labels $\mathbf{Y}$ are generated from $\mathbf{X}$ and $\mathbf{U}$. This assumption provides two benefits: (i) it allows labels to be determined by node attributes $\mathbf{X}$ (directly) as well as graph structure $\mathbf{A}$ (indirectly, via $\mathbf{U}$); and (ii) it allows us to express and train directly the function of the conditional distribution $\Pr\left[\mathbf{Y}|\mathbf{X},\mathbf{U}\right]$, which we then employ for *node classification*, i.e., to predict unobserved node labels.

In this work, we assume that this conditional probability is given by a simple two-layer neural network,

$$\Pr\left[\mathbf{Y}|\mathbf{X},\mathbf{U},\mathbf{W}\right] = \sigma(\mathrm{ReLU}([\mathbf{X}\mathbf{U}]\,W^{(0)})W^{(1)}) \tag{3}$$

where σ denotes the softmax function and weight matrices $\mathbf{W} = \{W^{(0)}, W^{(1)}\}$ are parameters that control the effect of $\mathbf{X}$ and $\mathbf{U}$ on labels $\mathbf{Y}$.

The reason for this choice is that we found this model to be sufficiently expressive for our empirical evaluation – and its analysis could easily extend to more complex models (e.g., neural networks with more hidden layers).

3.2 Algorithms

Problem 1 asks for predictions for $\mathbf{Y}_{\mathcal{V}-L}$ given the data $\mathcal{D} = (\mathbf{X}, \mathbf{A}, \mathbf{Y}_L)$ that are provided as input. Towards this end, we treat the remaining quantities in the model, i.e., the latent variables $\mathbf{U}$ and the weights of the neural network $\mathbf{W}$, as unobserved parameters $\theta = (\mathbf{U}, \mathbf{W})$ – and use their maximum lilekihood values $\hat{\mathbf{U}}$ and $\hat{\mathbf{W}}$ in making the predictions. To summarize, our approach JANE proceeds in two steps: first, a training step from which we learn the maximum likelihood estimates $\hat{\mathbf{U}}$ and $\hat{\mathbf{W}}$; second, a prediction step, in which we use the learned estimates to predict the missing labels.

Training From the product rule of probability, we have that the likelihood of the data $\mathcal{D}$ for given parameters θ is proportional to

$$\begin{aligned}\mathcal{L} &= \mathrm{P}\left[\mathcal{D}|\theta\right] \\ &= \mathrm{P}\left[\mathbf{X}, \mathbf{A}, \mathbf{Y}_L|\mathbf{U}, \mathbf{W}\right] \propto \mathrm{P}\left[\mathbf{Y}_L|\mathbf{X}; \mathbf{U}, \mathbf{W}\right] \mathrm{P}\left[\mathbf{A}|\mathbf{U}\right]\end{aligned} \tag{4}$$

Algorithm 1 describes the procedure to identify parameters θ that lead to maximum log-likelihood.

$$\begin{aligned}\hat{\theta} &= \arg\max_{\theta} \log \mathcal{L} \\ &= \arg\max_{\mathbf{U},\mathbf{W}} \left(\log \mathrm{P}\left[\mathbf{Y}_L|\mathbf{X}; \mathbf{U}, \mathbf{W}\right] + \log \mathrm{P}\left[\mathbf{A}|\mathbf{U}\right] \right)\end{aligned} \tag{5}$$

In summary, Algorithm 1 uses gradient descent on negative log-likelihood ($-\log \mathcal{L}$) to alternatingly update $\hat{\mathbf{U}}$ and $\hat{\mathbf{W}}$ towards their maximum-likelihood estimates, starting from an initial estimate of $\hat{\mathbf{U}}$ given by the spectral embedding of $\mathcal{G}$.

We provide the details below.

Choosing an initial estimate of U. Algorithm 1 begins its iterative updates from $\hat{\mathbf{U}}$. To obtain an initial maximum-likelihood estimate for it, it considers only the second term of log-likelihood (Eq. 5), as it involves only $\mathbf{U}$ and not $\mathbf{W}$. That is, the initial estimate $\hat{\mathbf{U}}$ is unsupervised and (temporarily) ignores labels.

$$\text{initial } \hat{\mathbf{U}} = \arg\min_{\mathrm{U}} \left[-\log \Pr\left(\mathbf{A} \mid \mathrm{U} \right) \right] \tag{6}$$

Using Eq. 2, we get

$$\begin{aligned}\hat{\mathbf{U}} &= \arg\min_{\mathrm{U}} \sum_{(i,j)\in\mathcal{E}} \frac{\left\|u_i - u_j\right\|^2}{s^2} - \log\left(\prod_{(i,j)\notin\mathcal{E}} \left(1 - e^{\frac{-\left\|u_i - u_j\right\|^2}{s^2}} \right) \right) \\ &\leq \arg\min_{\mathrm{U}} \sum_{(i,j)\in\mathcal{E}} \frac{\left\|u_i - u_j\right\|^2}{s^2} + \sum_{(i,j)\notin\mathcal{E}} e^{\frac{-\left\|u_i - u_j\right\|^2}{s^2}}.\end{aligned} \tag{7}$$

The last inequality holds because $-\left(\log\left(1-x\right)\right) \leq x, x \in [0,1)$. For an appropriate choice of scale, i.e. for smaller values of s, the second term in Equation 7 tends to 0 and the value of the first term dominates expression 7. Given that $\mathbf{L}$ is symmetric and $d_i = \sum_j a_{ij}, \forall i \in [n]$, based on the results of Belkin, et al [1],

$$\min_{\substack{\mathrm{U}:\forall l\in[k],\\ \left\|u_l\right\|=1,\\ \sum_p u_{l_p}=0}} \sum_{(i,j)\in\mathcal{E}} \left\|u_i - u_j\right\|^2 = \mathrm{tr}\left(\mathrm{U}^{\mathrm{T}}\mathbf{L}\mathrm{U}\right) = \sum_{l=1}^{k} \lambda_l\left(\mathbf{L}\right).$$

The condition $\mathrm{U} : \forall l \in [k], \quad \left\|u_l\right\| = 1, \sum_p u_{l_p} = 0$ normalizes the columns of U, removes translational invariance, and centers the solution around $\mathbb{0}$. This result

Algorithm 1: JANE

Input: $\mathbf{A}$; $\mathbf{X}$; $\mathbf{Y}_L$
Output: $\hat{\mathbf{U}}$ $\hat{\mathbf{W}}$ (max-likelihood-estimates)
Parameters: k (dimensionality of $\mathbf{U}$); T (training epochs); η_1, η_2 (learning rates)
`/* Initialize Û with first spectral eigenvectors */`
$\mathbf{L} \leftarrow \mathbf{D} - \mathbf{A}$
$\hat{\mathbf{U}} \leftarrow [\mathbf{e}_1(\mathbf{L}), \mathbf{e}_2(\mathbf{L}), \ldots, \mathbf{e}_k(\mathbf{L})]$
for $t \leftarrow 1$ ***to*** T **do**
  `/* Update Ŵ */`
  $\hat{\mathbf{W}} \leftarrow$ backpropagation$(\left[\mathbf{X}\ \hat{\mathbf{U}}\right], \hat{\mathbf{W}}, \mathbf{Y}_L)$ `/* Update Û */`
  $\hat{\mathbf{U}} \leftarrow \hat{\mathbf{U}} - \eta_1 \cdot \frac{\partial \mathcal{C}}{\partial \hat{\mathbf{U}}} - \eta_2 \cdot \frac{\partial(-\log \Pr[\ \mathbf{A}\ |\ \mathbf{U}\ ])}{\partial \hat{\mathbf{U}}}$
end

implies that the minimum value of the first term in Equation 7 is the sum of the k smallest eigenvalues of the graph Laplacian. Moreover, this minimum value is achieved when columns of U are the corresponding eigenvectors. Therefore, we set as initial estimate:

$$\hat{\mathbf{U}} = [\mathbf{e}_1(\mathbf{L}), \mathbf{e}_2(\mathbf{L}), \ldots, \mathbf{e}_k(\mathbf{L})]. \tag{8}$$

where $\mathbf{e}_i(\mathbf{L})$ is the i-th smallest spectral eigenvector.

Maximum Likelihood Estimation. During each training epoch, JANE first concatenates the current estimate $\hat{\mathbf{U}}$ with $\mathbf{X}$ and uses it as input to the neural network (Eq. 3). Since $\hat{\mathbf{W}}$ is present only in the first term of Equation 5, the loss function is

$$\mathcal{C} = \log \mathrm{P}\left[\mathbf{Y}_L | \mathbf{X}; \mathbf{U}, \mathbf{W}\right] \tag{9}$$

and the estimate $\hat{\mathbf{W}}$ is updated using stochastic gradient descent and standard backpropagation over it.

Subsequently, Algorithm 1 treats $\hat{\mathbf{W}}$ as fixed and updates its estimate $\hat{\mathbf{U}}$ using the gradient of both terms of Equation 5. The gradient of the first term $\mathcal{C}$ w.r.t. the current $\hat{\mathbf{U}}_l$ (l-th latent attribute) is obtained using backpropagation as follows,

$$\frac{\partial \mathcal{C}}{\partial \hat{\mathbf{U}}_l} = \sum_{i \in \mathbf{Y}_L} \left(\sum_{r=1}^{M} \left(a_{ir}^{(1)} - \mathbf{Y}_{ir} \right) W_{rl}^{(1)} \right) \times \left(\frac{a_{il}^{(0)}}{1 + \exp\left(-z_{il}^{(0)}\right)} \right) \times \left(\sum_{p=1}^{d+k} \left[W_{li}^{(0)} \right] \right) \tag{10}$$

where $a^{(0)}$, $a^{(1)}$ are activations from the hidden and output layers, and $z^{(0)}$ is the weighted sum from the input layer. The index i ranges over all nodes with

known labels, and r indexes over the M different classes available for prediction. The gradient of the second term is given by the following equation.

$$\frac{\partial\left(-\ \log \mathrm{P}\left(\ \mathbf{A}\mid \hat{\mathbf{U}}\ \right)\right)}{\partial \hat{\mathbf{u}}_i} = \left(\sum_{j:(i,j)\in\mathcal{E}} 2\times\frac{\hat{\mathbf{u}}_i-\hat{\mathbf{u}}_j}{s^2}\right)+ \left(\sum_{j:(i,j)\notin\mathcal{E}} -2\times\frac{\hat{\mathbf{u}}_i-\hat{\mathbf{u}}_j}{s^2}\times \mathrm{e}^{-\frac{\|\hat{\mathbf{u}}_i-\hat{\mathbf{u}}_j\|^2}{s^2}}\right) \quad (11)$$

where indices i and j range over all nodes in $\mathcal{V}$. Algorithm 1 uses the two gradients above with corresponding learning rates η_1, η_2 to update $\hat{\mathbf{U}}$.

Prediction Given maximum-likelihood estimates $\hat{\mathbf{U}}$ and $\hat{\mathbf{W}}$, we predict labels $\hat{\mathbf{Y}}_{\mathcal{V}-L}$ for all nodes in $\mathcal{V}-L$ using the softmax function applied row-wise.

$$\hat{\mathbf{Y}}_{\mathcal{V}-L} = \arg\max_{r\in M}\sigma\left(\mathrm{ReLU}\left(\left[\mathbf{X}\hat{\mathbf{U}}\right]\hat{W}^{(0)}\right)\hat{W}^{(1)}\right)_r \quad (12)$$

4 Experiments

In this section, we empirically evaluate the performance of JANE on synthetic and real-world datasets.

Baselines. We evaluate JANE against standard baselines from a variety of different design philosophies.

- JANE *and variants:* JANE-NU wherein we do not update the initial estimate of $\hat{\mathbf{U}}$ w.r.t. labels or graph structure during training, and JANE-R where the initial estimate of U is a random matrix.
- *Graph-structure based approaches:* Label Propagation (LP) [30], and DeepWalk (DW) [18] that encodes neighbourhood information via truncated random walks. These do not incorporate node attributes.
- *Deep attributed embeddings:* We evaluate LANE [9] which constructs node embeddings that encode graph structure and node attribute information, in addition to node labels.
- *Graph-convolution based approaches:* GCN [11] and GraphSAGE (mean aggregator) as representatives of graph convolutional networks. We acknowledge that this is extremely active area of research today and there are several approaches that demonstrate improved performance along axes such as training efficiency [4], explainability [27], etc. on various real-world datasets [7]. For economy of space, we empirically compare with two benchmark, representative methods to demonstrate our central point: these approaches sometimes fail because they strictly and inherently enforce homophily and social influence.

We were unable to reproduce the node classification results for AANE [8] based on the available implementation[1]. Further, we could not locate the authors' implementations for DANE [5]. Therefore, we do not report these results.

Experimental Setup. We implement JANE, JANE-NU, and LP in Pytorch. We use out-of-box Pytorch implementations of DeepWalk[2], GCN[3] and GraphSAGE[4]. And, we use a MATLAB implementation[5] of LANE [5]. In all of our experiments, as is standard, all approaches receive only the adjacency matrix of the graph $\mathbf{A}$ and the node attributes $\mathbf{X}$ as input, along with the same 10% and 20% of the node labels for training and validation, respectively. Wherever available, we use the default hyperparamater configurations as suggested in the original papers. Otherwise, we grid search over the hyperparameter space to find the best setting for all of our baselines. We perform all experiments on a Linux machine with 4 cores and 32GB RAM.

Reproducibility. To aid further research, we make our code publicly available[6]. This also includes an implementation for constructing synthetic datasets as described below. The real-world datasets (cf. Section 4.2) are publicly available.

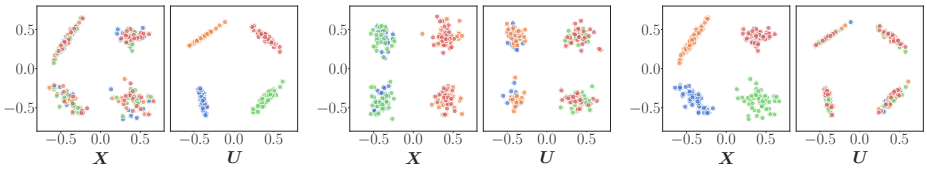
4.1 Node Classification on Synthetic Data

The goal of these experiments is two-fold – (1) demonstrate fundamental differences between and limitations of existing classification approaches using synthetic datasets wherein labels derive from only $\mathbf{X}$, only $\mathbf{U}$, or partly from both, and (2) show the strengths and general-purpose nature of JANE vis-à-vis the source of node labels.

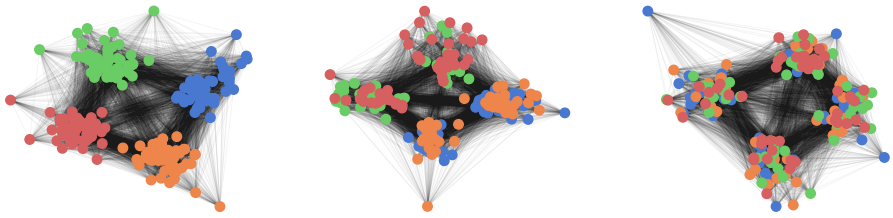
Synthetic Datasets. Figure 3 describes representative synthetic datasets generated according to the framework described previously.

We set the number of individual node features $|\mathbf{X}| = d = 2$ and number of latent features $|\mathbf{U}| = k = 2$. We generate these features (gaussian-distributed) for $n = 200$ points, each of which belongs to one of $M = 4$ classes and set the scale $s^2 = 1$. An influence parameter, $\alpha \in [0, 1]$, controls the degree to which node labels derive from $\mathbf{X}$ or $\mathbf{U}$: $\alpha = 0.0$ signifies that they derive only from $\mathbf{U}$ and are independent of $\mathbf{X}$; $\alpha = 1.0$ that they derive only from $\mathbf{X}$ and are independent of $\mathbf{U}$; and $\alpha = 0.5$ that they derive equally from $\mathbf{X}$ and $\mathbf{U}$ (specifically, without loss of generality, only the first feature from $\mathbf{X}$ and $\mathbf{U}$ contributes to label assignment). Figure 3a depict an instance of $\mathbf{X}$ and $\mathbf{U}$ each for $\alpha = \{0.0, 0.5, 1.0\}$, respectively. The colors of points represent classes. We construct the adjacency matrix from $\mathbf{U}$ as per Equation 2. Figure 3b depicts an instance of the corresponding graphs.

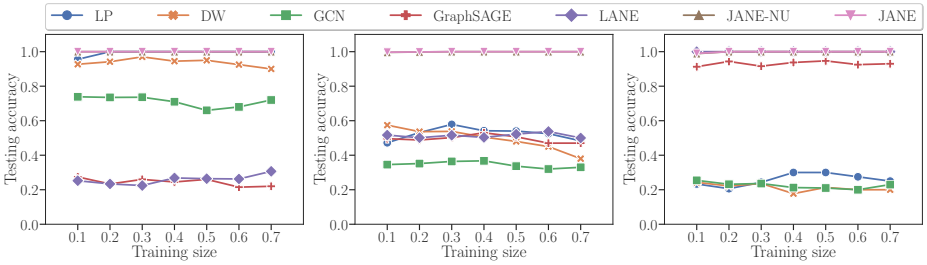
[1] https://github.com/xhuang31/AANE_Python
[2] https://github.com/phanein/deepwalk
[3] https://github.com/tkipf/pygcn
[4] https://github.com/williamleif/graphsage-simple
[5] https://github.com/xhuang31/LANE
[6] https://version.helsinki.fi/ads/jane



(a) Observed and latent attributes $\mathbf{X}$ and $\mathbf{U}$ (respectively) generated for $\alpha = 0.0$ (left), $\alpha = 0.5$ (center), and $\alpha = 1.0$ (right).



(b) Random graph generated from latent attributes $\mathbf{U}$ for $\alpha = 0.0$ (left), $\alpha = 0.5$ (center), and $\alpha = 1.0$ (right).



(c) Node classification accuracy averaged over randomly generated datasets for $\alpha = 0.0$ (left), $\alpha = 0.5$ (center), and $\alpha = 1.0$ (right).

Fig. 3: Figure 3a depicts 3 synthetically generated datasets $\mathbf{X}$ and $\mathbf{U}$ with class labels $\mathbf{Y}$ influenced only by $\mathbf{U}$ ($\alpha = 0.0$), partly by $\mathbf{U}$ and partly by $\mathbf{X}$ ($\alpha = 0.5$), and only by $\mathbf{X}$ ($\alpha = 1.0$). Figure 3b shows an instance of the corresponding graph generated from $\mathbf{U}$ according to Equation 2. Figure 3c compares the node classification accuracy of JANE and JANE-NU with the baselines averaged over 5 random train-test splits.

Implementation Details. We use Scikit-Learn's [17] MAKE_CLASSIFICATION to generate these datasets. Approaches do not have access to α or $\mathbf{U}$. JANE is trained as a two-layer neural network for a maximum of $T = 200$ epochs with dropout of 0.2 for each layer, weight decay of $5e^{-2}$, and learning rate of 0.005 using Adam. We set the number of eigenvectors $k = 2$ and choose a scaling factor $s^2 = 0.01$.

Performance. Figure 3c shows performance as a function of increasing training set sizes.

- $\alpha = 0.0$: LP and DW infer that labels derive from $\mathbf{A}$ (indirectly). GCN converges attribute values of nodes in the same cluster but is not perfectly accurate because $\mathbf{X}$ does not correlate with $\mathbf{Y}$. LANE forces the proximity representation to be similar to the attribute representation and then smoothens it using the labels. It does not perform well since there is no correlation between them.
- $\alpha = 0.5$: LP, DW are able to correctly classify nodes belong to 2 out of 4 classes, i.e. precisely those nodes whose labels are influenced by $\mathbf{U}$. Conversely, LANE is able to classify those nodes belong to two classes of nodes that correlate with $\mathbf{X}$. GCN smoothens attribute values of adjacent nodes and thus can correctly infer labels correlated with $\mathbf{X}$.
- $\alpha = 1.0$ LP and DW reduce to random classifiers since adjacent nodes do not have similar labels. GCN reduces to a nearly random classifier because by forcing adjcent nodes with different attribute values to become similar, it destroys the correlation between $\mathbf{X}$ and the labels.

In each of the three cases, JANE-NU and JANE achieve perfect accuracy because they flexibly learn during training whether labels correlate with $\mathbf{X}$, $\mathbf{A}$ (indirectly), or partially both. While these datasets are simplistic in nature, this demonstrates how the homophily assumption–requiring nodes with similar proximity and attributes to have the same labels–limits the performance of other approaches.

4.2 Node Classification on Real-World Data

We seek to understand – (1) to what extent JANE can capture more complex network structures and their correlations with node labels, and (2) how well JANE compares with our baselines on these datasets.

Datasets. We show our results on four tasks for a total of 9 graphs. If the original graph is diconnected, we extract node attributes and labels belonging to its largest connected component.

- *Citation Networks:* We use Cora, Citeseer, Pubmed [22], and UAI2010 [25]. Here, nodes represent academic papers, edges denote a citation between two nodes, node features are 0/1-valued sparse bag-of-words vectors and class labels denote the subfield of research that the papers belong to.
- *Social Networks:* We focus on BlogCatalog and Flickr where the task is to predict pre-defined categories of blogs and images, respectively. Nodes are users that post content, edges represent follower relationships, and features are specified by a list of tags reflecting the interests of the users [12].
- *Air-traffic Networks:* Based on flight records from Brazil, Europe, and USA, each node is an airport and an edge indicates a commercial airline route exists between them. Labels denote the level of activity in terms of people and flights passing through an airport [20]. Since no features for the nodes exists, we assign the all-ones vector as the sole attribute.

Dataset	Nodes	Edges	Classes	Features
Cora	2708	5429	7	1433
Citeseer	3327	4732	6	3703
Pubmed	19717	44325	3	500
UAI2020	3067	56622	18	4973
BlogCatalog	5196	171743	6	8189
Flickr	7575	239738	9	12047
Brazil	131	1038	4	NA
Europe	399	5995	4	NA
USA	1190	13599	4	NA
PPI	2373	56952	1	121

Table 1: Summary of dataset statistics.

- *Biological Networks:* We use a processed protein-protein interaction (PPI) dataset [6] where the task is to identify protein roles based on gene ontology sets using positional gene sets, motif gene sets, and immunological signatures as features [23].

Table 1 summarizes the dataset statistics.

Experimental Setup. for the citation datasets, we use the same train-validation-test splits as in Yang, et al [28] minus the nodes which do not belong to the largest connected component. These comprise of 20 samples for each class and represent 5% of the entire dataset. We use 500 additional samples as a validation set for hyperparameter optimization as per Kipf, et al [11] to enable fair comparison. For all other tasks, we use 10% and 20% of the dataset for training and validation, respectively. We evaluate the performance of all approaches on the remaining nodes of the graph. Values of hyperparameters k, the number of eigenvectors of the graph Laplacian, and the scaling factor s are determined empirically so as to minimize Equation 7.

Dataset	LP	DW	GCN	GraphSAGE	LANE	JANE-R	JANE-NU	JANE
Cora	74.29	32.54	79.20	**79.70**	64.50	59.34	78.87	**79.24**
Citeseer	67.19	61.31	69.11	**69.84**	56.93	52.80	69.71	**69.77**
Pubmed	64.33	77.91	**82.26**	81.86	78.81	72.91	82.70 1	**83.18**
UAI2010	42.18	44.67	49.52	61.22	66.36	45.96	66.91	**69.84**
BlogCatalog	48.81	37.18	70.53	75.13	**82.26**	70.85	74.85	77.53
Flickr	41.24	33.61	49.58	57.17	**64.12**	49.24	59.92	61.68
Brazil	26.83	51.71	27.21	28.26	24.59 5	29.32	68.55	**69.93**
Europe	26.33	47.15	25.46	46.61	27.50	28.81	45.97	**50.21**
USA	24.73	56.67	24.13	48.63	25.74 5	27.67	57.10	**61.91**
PPI	38.83	47.19	51.19	**61.26**	53.73	45.12	62.68	**62.98**

Table 2: Classification accuracy (%) on test data averaged over 10 independent runs of $T = 200$ epochs each. Bold denotes best average accuracy or overlapping with best accuracy range (within ± 0.3 standard deviation). JANE-NU and JANE perform as well as or better than LANE, GCN and GraphSAGE on most datasets and consistently well across datasets.

Performance Analysis. Table 2 provides the average accuracy of each approach over 10 independent runs for $T = 200$ epochs each for GCN, GraphSAGE, LANE, and JANE. Values in bold denote that the approach either performs best or its accuracy range overlaps with that of the best. JANE-NU and JANE consistently outperform LP and DW on all datasets by significant margins. JANE-R on the other hand, has significantly lower performance either because it gets stuck in a local optima or it requires much longer training time. This shows that the choice of the initial estimate is crucial to performance. The embedding obtained from DW is unsupervised and thus its predictive power is limited in comparison to our $\hat{\mathbf{U}}$ which is label-informed. Both variants of JANE are competitive with GCN and GraphSAGE on Cora, Citeseer, and UAI2020. For instance, GraphSAGE achieves 79.70% accuracy on Cora, while JANE gets 79.24% which is within the margin of error. We observe strong performance by JANE-NU and JANE on Pubmed and PPI. This may be explained in part by Li, et al [13]'s observation that Pubmed exhibits strong manifold structure as well as JANE's ability to better utilize graph structural information. Lastly, we find significant gains on the social, and air-traffic datasets. LANE [9] report strong performance on Blogcatalog and Flickr, but its performance is noticeably poorer on citation, flights, and biological datasets.

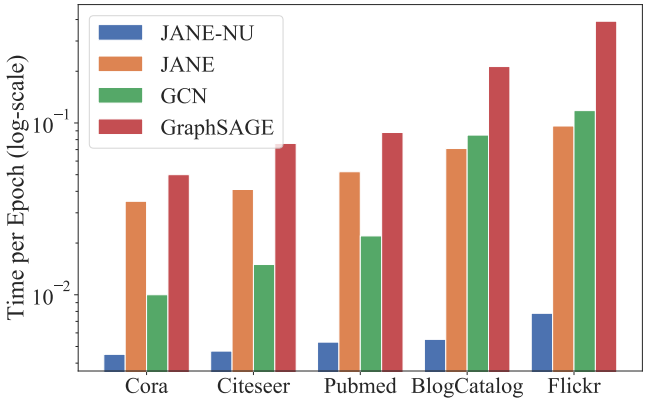


Fig. 4: Average training time per epoch (seconds) of baselines on different datasets.

Runtime. Figure 4 plots the average time for a single training epoch for JANE-NU, JANE, GCN, and GraphSAGE on various datasets. JANE is comparatively the slowest model while JANE-NU is by far the fastest. This is because JANE-NU is a vanilla neural network model that does not update its estimate of $\hat{\mathbf{U}}$. Further, a single laplacian eigenvector can be approximately computed using the Lanczos algorithm in $\tilde{\mathcal{O}}(|\mathcal{E}|)$ (up to log factors). Thus k eigenvectors can be

computed in $\tilde{\mathcal{O}}(k \cdot |\mathcal{E}|)$ and this is a one-time operation. Note, we do not need to perform a full eigendecomposition.

Parameter Sensitivity. A crucial parameter of JANE is the number of eigenvectors used as features during training. Figure 5 demonstrates their impact on performance. In each case, JANE is allowed a maximum of 200 training epochs. We find that test accuracy consistently increases as dimension of $\hat{\mathbf{U}}$ increases up until a certain threshold. However, increasing beyond this threshold introduces noise and reduces performances. Having too many eigenvectors gives marginal and diminishing returns in performance while increasing the runtime.

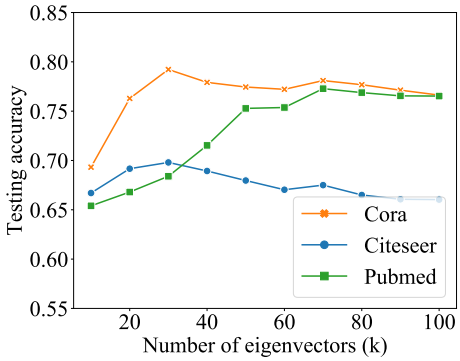


Fig. 5: JANE's classification accuracy (%) on test data averaged over 10 runs w.r.t. number of eigenvectors.

Limitations. The primary limitation of JANE is the time and memory requirement for computing the gradient of $\mathbf{A}$ w.r.t. $\hat{\mathbf{U}}$ (cf. Equation 11). These requirements grow linearly in graph size. Since the gradient is computed in every training epoch, it may not be viable to fit it into GPU memory. Future work can outline procedures for generating mini-batches and schedules for updating the estimate of $\hat{\mathbf{U}}$ using $\mathbf{A}$ at regular intervals (as opposed to every epoch as we do now) whilst still gaining the same benefits. It is crucial to note however, that JANE-NU is orders of magnitude faster than the other methods while also demonstrating strong performance.

5 Conclusion

In this paper, we developed an approach to node classification that flexibly adapts to settings ranging between graphs where labels are predicted by node proximity, on one hand, and graphs where labels are predicted by node attributes, on the other. We propose a generative framework to demonstrate how graph structural information and node attributes both, can influence the labels of nodes. Even simple instances of such situations, as shown in Figure 1 and empirically

evaluated in Figure 3b, severely affect the performance of various embedding methods and standard graph convolutional networks. Our principled approach, JANE, jointly and effectively utilizes Laplacian eigenmaps and individual features for classification leading to strong performance on a variety of datasets. Given its simplicity, interpretability and performance, JANE can serve as a useful starting point in designing models that holistically account for different sources of node labels. As a future direction, we aim to the evaluate the design and performance of advanced graph neural networks that go beyond requiring or enforcing homophily.

References

1. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural computation 15(6), 1373–1396 (2003)
2. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. Journal of machine learning research 7(Nov), 2399–2434 (2006)
3. Blum, A., Lafferty, J., Rwebangira, M.R., Reddy, R.: Semi-supervised learning using randomized mincuts. In: Proceedings of the twenty-first international conference on Machine learning. p. 13. ACM (2004)
4. Cong, W., Forsati, R., Kandemir, M., Mahdavi, M.: Minimal variance sampling with provable guarantees for fast training of graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1393–1403 (2020)
5. Gao, H., Huang, H.: Deep attributed network embedding. In: IJCAI (2018)
6. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. pp. 1025–1035 (2017)
7. Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., Leskovec, J.: Open graph benchmark: Datasets for machine learning on graphs. arXiv preprint arXiv:2005.00687 (2020)
8. Huang, X., Li, J., Hu, X.: Accelerated attributed network embedding. In: Proceedings of the 2017 SIAM international conference on data mining. pp. 633–641. SIAM (2017)
9. Huang, X., Li, J., Hu, X.: Label informed attributed network embedding. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining. pp. 731–739 (2017)
10. Joachims, T.: Transductive learning via spectral graph partitioning. In: Proceedings of the 20th International Conference on Machine Learning (ICML-03). pp. 290–297 (2003)
11. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (ICLR) (2017)
12. Li, J., Hu, X., Tang, J., Liu, H.: Unsupervised streaming feature selection in social media. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (2015)
13. Li, Q., Han, Z., Wu, X.M.: Deeper insights into graph convolutional networks for semi-supervised learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)

14. Lorrain, F., White, H.C.: Structural equivalence of individuals in social networks. The Journal of mathematical sociology 1(1), 49–80 (1971)
15. Marsden, P.V., Friedkin, N.E.: Network studies of social influence. Sociological Methods & Research 22(1), 127–151 (1993)
16. McPherson, M., Smith-Lovin, L., Cook, J.M.: Birds of a feather: Homophily in social networks. Annual review of sociology 27(1), 415–444 (2001)
17. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830 (2011)
18. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 701–710 (2014)
19. Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., Tang, J.: Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In: Proceedings of the eleventh ACM international conference on web search and data mining. pp. 459–467 (2018)
20. Ribeiro, L.F., Saverese, P.H., Figueiredo, D.R.: struc2vec: Learning node representations from structural identity. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 385–394. ACM (2017)
21. Sailer, L.D.: Structural equivalence: Meaning and definition, computation and application. Social Networks 1(1), 73–90 (1978)
22. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI magazine 29(3), 93–93 (2008)
23. Subramanian, A., Tamayo, P., Mootha, V.K., Mukherjee, S., Ebert, B.L., Gillette, M.A., Paulovich, A., Pomeroy, S.L., Golub, T.R., Lander, E.S., et al.: Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. Proceedings of the National Academy of Sciences 102(43), 15545–15550 (2005)
24. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
25. Wang, W., Liu, X., Jiao, P., Chen, X., Jin, D.: A unified weakly supervised framework for community detection and semantic matching. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 218–230. Springer (2018)
26. Wang, X., Zhu, M., Bo, D., Cui, P., Shi, C., Pei, J.: Am-gcn: Adaptive multi-channel graph convolutional networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1243–1253 (2020)
27. Wu, J., He, J., Xu, J.: Net: Degree-specific graph neural networks for node and graph classification. arXiv preprint arXiv:1906.02319 (2019)
28. Yang, Z., Cohen, W., Salakhudinov, R.: Revisiting semi-supervised learning with graph embeddings. In: International conference on machine learning. pp. 40–48. PMLR (2016)
29. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. In: NeurIPS. pp. 4800–4810 (2018)
30. Zhou, D., Huang, J., Schölkopf, B.: Learning from labeled and unlabeled data on a directed graph. In: Proceedings of the 22nd international conference on Machine learning - ICML '05. pp. 1036–1043. ACM Press, Bonn, Germany (2005), http://portal.acm.org/citation.cfm?doid=1102351.1102482

31. Zhu, X., Goldberg, A.B.: Introduction to semi-supervised learning. Synthesis lectures on artificial intelligence and machine learning 3(1), 1–130 (2009)