Independence Promoted Graph Disentangled Networks

Yanbei Liu,[1] **Xiao Wang,**[2*] **Shu Wu,**[3] **Zhitao Xiao**[1]

[1]School of Life Sciences, Tiangong University
[2]School of Computer Science, Beijing University of Posts and Telecommunications
[3]Center for Research on Intelligent Perception and Computing, National Laboratory
of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences
{liuyanbei, xiaozhitao}@tjpu.edu.cn, xiaowang@bupt.edu.cn, shu.wu@nlpr.ia.ac.cn

Abstract

We address the problem of disentangled representation learning with independent latent factors in graph convolutional networks (GCNs). The current methods usually learn node representation by describing its neighborhood as a perceptual whole in a holistic manner while ignoring the entanglement of the latent factors. However, a real-world graph is formed by the complex interaction of many latent factors (e.g., the same hobby, education or work in social network). While little effort has been made toward exploring the disentangled representation in GCNs. In this paper, we propose a novel Independence Promoted Graph Disentangled Networks (IPGDN) to learn disentangled node representation while enhancing the independence among node representations. In particular, we firstly present disentangled representation learning by neighborhood routing mechanism, and then employ the Hilbert-Schmidt Independence Criterion (HSIC) to enforce independence between the latent representations, which is effectively integrated into a graph convolutional framework as a regularizer at the output layer. Experimental studies on real-world graphs validate our model and demonstrate that our algorithms outperform the state-of-the-arts by a wide margin in different network applications, including semi-supervised graph classification, graph clustering and graph visualization.

Introduction

Graph with a set of objects and their relationships is a significant kind of data structure, which has been adopted in various areas such as social networks (Hamilton, Ying, and Leskovec 2017) and protein-protein interaction networks (Fout et al. 2017). Graph convolutional networks (GCNs), as a typical deep learning technique on graph data, have attracted considerable attentions (Defferrard, Bresson, and Vandergheynst 2016; Kipf and Welling 2017). GCNs extend the traditional convolution operation to graph and it learns the node representation by propagating its neighbor information on graph. With the learned node representations, we can perform various tasks on graphs such as node clustering, classification and link prediction (Zhang, Cui, and Zhu 2018; Wu et al. 2019)

Despite the remarkable performance, the existing graph convolutional networks generally learn node representations by absorbing the node's neighborhood as a perceptual whole, and the representations they have learned are prone to over-smoothing faced by a deep GCNs (Li, Han, and Wu 2018). Yet graphs are usually formed by highly complex interactions of many latent factors. For example, a person in a social network usually connects with others for various reasons (e.g., hobby, education and work), hence it simultaneously contains several partial information from its neighbors. Generally speaking, latent factors can take into account the nuances between the different parts of the neighborhood and make node representations more informative. Furthermore, latent factors can enable to discover the underlying structures in graph data, such as neighborhood and community. Therefore, how to learn representations that capture desired latent factors behind a graph is of great importance for GCNs.

However, disentangling the latent factors behind a graph has two main challenges. One is how to learn node disentangled representations which need extract the different parts of its neighborhood but not the whole. Yet graph has numerous nodes and an unusually complex network structure. The other is how to encourage independence among the learned latent factors. Taking social networks as an example, although "roommate" and "classmate" are two different factors that lead people to connect each other, the former is obviously a subset of the latter, hence they are not sufficiently diversified and will inevitably result in redundant representations. We argue that the representations of latent factors should be good and different. Yet the relationship between these factors is usually very complicated and nonlinear, and the solution needs to be differentiable so as to support end-to-end training in GCNs. So it is of very importance to encourage independence among the learned latent factors. We notice that Ma et al. (Ma et al. 2019) presented a disentangled graph convolutional network to learn disentangled node representations. However, they only consider the disentangled representation learning while neglecting the independence of the latent factors. Therefore, how to learn representations that disentangle the latent factors with desired independence property remains largely unexplored
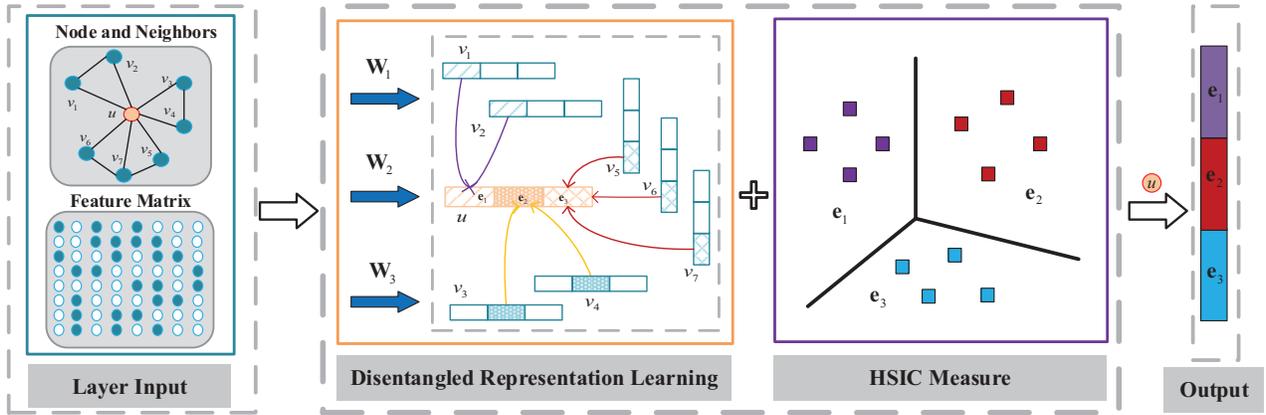
Figure 1: Illustration of the proposed IPGDN's layer. It attempts to disentangle latent factors in the graph data, and simultaneously respects independence across different representations. To disentangle node u, IPGDN's layer first construct features from different aspects of its neighbors via disentangled representation learning, then encourage the independence among latent representations through minimize HSIC to obtain the final result. This example assumes that there are three latent factors, corresponding to the three channels $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$.

in the field of graph convolutional networks.

In this paper, we propose Independence Promoted Graph Disentangled Networks (IPGDN), a novel approach for disentangled representation learning that can automatically discover the independent latent factors in graph data. Specifically, on the one hand, we present disentangled representation learning by neighborhood routing mechanism. On the other hand, to enforce the independence among different latent representations, our model minimize the dependence among different representations with a kernel-based measure, in particular the Hilbert-Schmidt Independence Criterion (HSIC). The node disentangled representation learning and independence regularization are jointly optimized in a unified framework so that it finally leads to a better graph disentangled representation. The experimental results on benchmark datasets demonstrate the superb performance of our approach on graph analytic tasks, including semi-supervised node classification, node clustering, and graph visualization.

We summarize the main contributions as follows:

- To our best knowledge, this is the first attempt to enforce the independence property among different latent representations in graph disentangled representation learning. We propose a novel independently regularized disentanglement framework for graph convolutional network, which represents topological structure and node content into different latent factors.

- We present a kernel Hilbert-Schmidt Independence Criterion to effectively measure dependence among different latent representations, which can promote the quality of disentangled representations of the graph.

- Experiments on benchmark graph datasets demonstrate that our graph disentangled networks outperform the others on three popular tasks.

Graph Convolutional Networks

First, let us define some notations used throughout this paper. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ be an attributed information network, where $\mathcal{V}$ denotes the set of n nodes and $\mathcal{E}$ represents the set of edges. $\mathbf{X} \in \mathbb{R}^{n \times f}$ is a matrix that encodes all node attribute information, and $\mathbf{x}_i$ describes the attribute associated with node i. We denote $(u, v) \in \mathcal{E}$ as an edge between nodes u and v. Here we primarily focus on undirected graphs.

Graph convolutional networks generalize traditional convolutional neural networks to the graph domain. Though a number of different GCNs methods have been proposed, here we focus on a representative one proposed by Kipf and Welling (Kipf and Welling 2017). Given a graph with adjacency matrix $\mathbf{A}$ and node feature matrix $\mathbf{X}$, its propagation rule is as follows:

$$\mathbf{H}^{(l)} = \rho\big(\widetilde{\mathbf{D}}^{-\frac{1}{2}} \widetilde{\mathbf{A}} \widetilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{H}^{(l-1)} \mathbf{W}^{(l-1)}\big), \qquad (1)$$

where $\mathbf{A}$ is the adjacent matrix, $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\widetilde{\mathbf{D}} = \sum_j \widetilde{A}_{ij}$. $\mathbf{H}^{(l)}$ is the matrix of activations in the l-th layer, and $\mathbf{H}^{(0)} = \mathbf{X}$. $\rho(\cdot)$ is a non-linear activation function such as ReLU and $\mathbf{W}^{(l-1)}$ are trainable parameters. The general philosophy is that nodes should exchange information with their immediate neighbors in each convolutional layer, followed by applying learnable filters and some non-linear transformation. This architecture can be trained end-to-end using task-specific loss function, for example, the cross entropy loss in semi-supervised node classification as follows:

$$\mathcal{L}_{cel} = -\sum_{v \in \mathcal{V}^L} \sum_{c=1}^{C} \mathbf{Y}_{v,c} log(\widetilde{\mathbf{Y}}_{v,c}), \qquad (2)$$

where $\mathcal{V}^L$ is the set of labeled nodes, C is the number of classes, $\mathbf{Y}$ is the label matrix and $\widetilde{\mathbf{Y}} = softmax(\mathbf{H}^{(L)})$ are predictions of GCNs by passing the hidden representation in the final layer $\mathbf{H}^{(L)}$ to a softmax function.

IPGDN: the Proposed Model

This paper is concerned with independently disentangled representations for processing graph data. As shown in Figure 1, we design our IPGDN's layer simultaneously integrating disentangled representation learning and independence among different representations into an unified framework. In this section, we firstly present a way of disentangled representation learning in graph. Secondly, we propose a measure of dependence across latent representations based on HSIC. Thirdly, we introduce our IPGDN architecture: the model to enhance independent representations in semi-supervised classification task.

Disentangled Representation Learning

We assume that each node is composed of M independent components, i.e., there are M latent factors (corresponding to $\mathbf{M}$ channels) to be disentangled. Given nodes $\mathbf{x}_u \in \mathbb{R}^f$ and $\{\mathbf{x}_v \in \mathbb{R}^f : (u,v) \in \mathcal{E}\}$ as input, we define $\mathbf{h}_u = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_M] \in \mathbb{R}^{f_l}$ as the hidden representations of the node u, where $\mathbf{e}_m \in \mathbb{R}^{\frac{f_l}{M}}(1 \le m \le M)$ is the m-th disentangled representation. The component $\mathbf{e}_m$ is for describing the aspect of node u that are pertinent to the m-th factor.

For a node $o \in \{u\} \cup \{v : (u,v) \in \mathcal{E}\}$, to extract its features from different aspects of its neighbors, we firstly project its feature vector $\mathbf{x}_o$ into different subspaces as follows:

$$\mathbf{z}_{o,m} = \rho(\mathbf{W}_m^T \mathbf{x}_o + \mathbf{b}_m), \qquad (3)$$

where $\mathbf{W}_m \in \mathbb{R}^{f_{l-1} \times \frac{f_l}{M}}$ and $\mathbf{b}_m \in \mathbb{R}^{\frac{f_l}{M}}$ are the parameters of m-th channel. To ensure numerical stability and prevent the neighbors with overly rich features, we use ℓ_2 norm for normalization as $\mathbf{z}_{o,m}/\|\mathbf{z}_{o,m}\|_2$.

To comprehensively capture aspect m of node u, we need to construct $\mathbf{e}_m$ from both $\mathbf{z}_{u,m}$ and $\{\mathbf{z}_{v,m} : (u,v) \in \mathcal{E}\}$, which can identify the latent factor and assign the neighbor to a channel. Here we leverage neighborhood routing mechanism (Ma et al. 2019) as follows:

$$\mathbf{e}_m^{(t)} = \frac{\mathbf{z}_{u,m} + \sum_{v:(u,v)\in\mathcal{E}} p_{v,m}^{(t-1)} \mathbf{z}_{v,m}}{\left\| \mathbf{z}_{u,m} + \sum_{v:(u,v)\in\mathcal{E}} p_{v,m}^{(t-1)} \mathbf{z}_{v,m} \right\|_2}, \qquad (4)$$

$$p_{v,m}^{(t)} = \frac{exp(\mathbf{z}_{v,m}^T \mathbf{e}_m^{(t)})}{\sum_{m=1}^M exp(\mathbf{z}_{v,m}^T \mathbf{e}_m^{(t)})}, \qquad (5)$$

where iteration $t = 1, 2, ..., T$. $p_{v,m}$ indicates the probability that factor m is the reason why node u reaches neighbor v, and satisfies $p_{v,m} \ge 0, \sum_{m=1}^M p_{v,m} = 1$. Also, $p_{v,m}$ is the probability that we should use neighbor v to construct $\mathbf{e}_m$. The neighborhood routing mechanism will iteratively infer $p_{v,m}$ and construct $\mathbf{e}_m$.

Independence across Latent Representations

To enhance the disentangled informativeness, we encourage the representations of different factors to be of sufficient independence by using HSIC.

For latent factors $1 \le i, j \le M, i \ne j$, let $\mathbf{e}_i$ and $\mathbf{e}_j$ be disentangled representations from two latent factors containing $\frac{f_l}{M}$ data points $\{(e_{i,p}, e_{j,p}) \in \mathcal{X} \times \mathcal{Y}\}_p^{\frac{f_l}{M}}$, which are jointly drawn from a probability distribution $P_{\mathbf{e}_i, \mathbf{e}_j}$. $\phi(\mathbf{e}_i)$ and $\psi(\mathbf{e}_j)$ are functions that map $\mathbf{e}_i \in \mathcal{X}$ and $\mathbf{e}_j \in \mathcal{Y}$ to kernel space $\mathcal{F}$ and Ω with respect to the kernel functions $k(e_{i,p}, e_{i,q}) = <(\phi(e_{i,p}), \phi(e_{i,q})>$ and $s(e_{j,p}, e_{j,q}) = <(\phi(e_{j,p}), \phi(e_{j,q})>$. Note that $\mathcal{F}$ and Ω are Reproducing Kernel Hilbert Space (RKHS) on $\mathcal{X}$ and $\mathcal{Y}$, respectively. The cross-covariance is a function that gives the covariance of two random variables and defined as follows:

$$\mathcal{C}_{\mathbf{e}_i, \mathbf{e}_j} = E_{\mathbf{e}_i, \mathbf{e}_j}[(\phi(\mathbf{e}_i) - \mu_{\mathbf{e}_i}) \otimes (\psi(\mathbf{e}_j) - \mu_{\mathbf{e}_j})], \qquad (6)$$

where $\otimes$ is the tensor product. Then, HSIC as the Hilbert-Schmidt norm of the associated cross-covariance operator $\mathcal{C}_{\mathbf{e}_i, \mathbf{e}_j}$ is defined as follows:

$$\text{HSIC}(P_{\mathbf{e}_i, \mathbf{e}_j}, \mathcal{F}, \Omega) := \left\| \mathcal{C}_{\mathbf{e}_i, \mathbf{e}_j} \right\|_{HS}^2, \qquad (7)$$

where $\|\mathbf{A}\|_{HS} = \sqrt{\Sigma_{i,j} a_{ij}^2}$.

Accordingly, the empirical version of HSIC (Gretton et al. 2005) is given as follows:

$$\text{HSIC}(\mathbf{e}_i, \mathbf{e}_j) = (\frac{f_l}{M} - 1)^{-2} tr(\mathbf{K R S R}), \qquad (8)$$

where $\mathbf{K}$ and $\mathbf{S}$ are the Gram matrices with $k_{p,q} = k(e_{i,p}, e_{i,q}), s_{p,q} = s(e_{j,p}, e_{j,q})$. $r_{i,j} = \delta_{i,j} - \frac{M}{f_l}$ centers the Gram matrix to have zero mean in the feature space. There are two main advantages to use the HSIC to measure the dependence of representations. First, HSIC maps representations into a reproducing kernel Hilbert space to measure their dependence such that correlations measured in that space correspond to high-order joint moments between the original distributions and more complicated (such as nonlinear) dependence can be addressed. Second, this method is able to estimate dependence between representations without explicitly estimating the joint distribution of the random variables. In our implementation, we use the inner product kernel function, i.e. $\mathbf{K} = \mathbf{e}_i \mathbf{e}_i^T$, and promising performances are achieved. Note that minimizing $\text{HSIC}(\mathbf{e}_i, \mathbf{e}_j)$ enhances the independence between $\mathbf{K}$ and $\mathbf{S}$, which penalties the consistency between kernel matrices from different latent representation parameterized by different projection matrices $\mathbf{W}$.

Network Architecture

In this section, we describe the overall network architecture of IPGDN for performing node-related tasks.

By using disentangled representation learning module, we design the propagation of the first L layers' output as follows:

$$\mathbf{h}_u^{(l)} = \text{dropout}\big(h^{(l)}(\mathbf{h}_u^{(l-1)}, \{\mathbf{h}_v^{(l-1)} : (u,v) \in \mathcal{E}\})\big), \quad (9)$$

where $h^{(l)}$ denotes the output function of the l-th layer, $1 \le l \le L$. $u \in \mathcal{V}$ and $\mathbf{h}_u^{(0)} = \mathbf{x}_u$. The dropout operation is appended after every layer and is enabled only during training.

Algorithm 1: The Proposed IPGDN's Layer

Input: $\mathbf{x}_u \in \mathbb{R}^{f_{l-1}}$:the feature vector of node u;
  $\{\mathbf{x}_v \in \mathbb{R}^{f_{l-1}} : (u,v) \in \mathcal{G}\}$:its neighbors' features;
  M:number of channels;
  T:iterations of routing.
Output: $\mathbf{h}_u \in \mathbb{R}^{f_l}$: node disentangled representations.
Param: $\mathbf{W}_m \in \mathbb{R}^{f_{l-1} \times \frac{f_l}{M}}$, $\mathbf{b}_m \in \mathbb{R}^{\frac{f_l}{M}}$.
while *each* $o \in \{u\} \cup \{v : (u,v) \in \mathcal{G}\}$ **do**
    for *each* $m \in M$ **do**
        Calculate $\mathbf{z}_{o,m}$ by Eq. (3);
        $\mathbf{z}_{o,m} \longleftarrow \mathbf{z}_{o,m}/\|\mathbf{z}_{o,m}\|_2$.
    end
end
while *routing iteration* $t \in T$ **do**
    for *each* $v \in (u,v) \in \mathcal{G}$ **do**
        Calculate $p_{v,m}^{(t)}$ by Eq. (5).
    end
    for *each* $m \in M$ **do**
        Update $\mathbf{e}_m^{(t)}$ by Eq. (4).
    end
end
while *latent factors* $(i,j) \in M$ **do**
    Calculate dependence among representations by Eq. (8);
    Minimize Eq. (10).
end
$\mathbf{h}_u \longleftarrow$ the concatenation of $\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_M$.

To enhance that the learned representations are independent, we incorporate the HSIC regularization at the L-th output layer $\mathbf{h}_u^{(L)} = [\mathbf{e}_1^{(L)}, \mathbf{e}_2^{(L)}, ..., \mathbf{e}_M^{(L)}]$, which is shown as follows:

$$\mathcal{L}_{reg} = \sum_{u \in \mathcal{V}^L} \sum_{\mathbf{e}_i \neq \mathbf{e}_j} \text{HSIC}(\mathbf{e}_i^{(L)}, \mathbf{e}_j^{(L)}), \qquad (10)$$

For semi-supervised classification task, the final layer of the network architecture is a fully-connected layer as follows:

$$\mathbf{h}^{(L+1)} = (\mathbf{W}^{(L+1)})^T \mathbf{h}^{(L)} + \mathbf{b}^{(L+1)}, \qquad (11)$$

where transformation matrix $\mathbf{W}^{(L+1)} \in \mathbb{R}^{f_L \times C}$, parameters vector $\mathbf{b}^{(L+1)} \in \mathbb{R}^C$.

Next, $\mathbf{h}^{(L+1)}$ is passed to a softmax function to get the predicted labels:

$$\widetilde{\mathbf{y}} = softmax(\mathbf{h}^{(L+1)}). \qquad (12)$$

Then, we can use cross entropy loss $\mathcal{L}_{cel}$ defined in Eq. (2) as an objective function. Finally, we jointly minimize the loss function by combining the above terms:

$$\mathcal{L} = \mathcal{L}_{cel} + \lambda \mathcal{L}_{reg} \qquad (13)$$

where λ is hyper-parameters that control the impact of different regularizations. We compute the gradients via back-propagation, and optimize the parameters with Adam (Kingma and Ba 2014). Parameter $\lambda \geq 0$ is a trade-off parameter. The overall process of IPGDN's layer is shown in Algorithm 1.

To summarize, our approach has the following merits: (1) The HSIC penalty is only based on the learned representation, i.e. network parameters–not additional assumptions. (2) Both disentangled representation learning and independence of different representations are addressed in an unified framework; (3) Our approach is easy to optimize with the Adam, and since the value of our objective function converges fast, the algorithm is effective for graph disentangled representation.

Complexity Analysis for IPGDN Algorithm

In our model, operation of the IPGDN's layer can be parallelized across all nodes. Thus, it is highly efficient. The computation complexity of our method is $\mathcal{O}\Big(|\mathcal{E}| \sum_{l=0}^{L} f^{(l)} + |\mathcal{V}|(\sum_{l=1}^{L} f^{(l-1)} f^{(l)} + (f^{(L)})^2) + T \sum_{l=1}^{L} f^{(l)}\Big)$, where $|\mathcal{E}|$ is the number of edges, $|\mathcal{V}|$ is the number of nodes, T is the iteration times used in neighborhood routing mechanism and $f^{(l)}$ is the dimensionality of the l-th hidden layer. It is worth noting that our algorithm is linear with respect to the number of nodes and number of edges in the graph respectively, which is in the same order as other GCNs.

Independence Analysis of HSIC

In this section, we analyze theoretical properties of the Hilbert-Schmidt Independence Criterion for enhancing independence among different representations. Whenever $\mathcal{F}$, Ω are RKHS with characteristic kernels k, s (in the sense of (Fukumizu et al. 2008)), then $\text{HSIC}(P_{\mathbf{e}_i, \mathbf{e}_j}, \mathcal{F}, \Omega) = 0$ if and only if $\mathbf{e}_i$ and $\mathbf{e}_j$ are independent. Intuitively, a characteristic kernel leads to an injective embedding of probability measures into the corresponding RKHS. The HSIC is the squared RKHS distance between the embedded joint distribution and the embedded product of the marginals. Examples of characteristic kernels are Gaussian RBF kernels and Laplace kernels. HSIC is zero if two representations are independent. Note that non-characteristic and non-universal kernels can also be used for HSIC, although they may not guarantee that all dependence is detected. Different kernels incorporate distinctive prior knowledge into the dependence estimation, and they focus HSIC on dependence of a certain type. For instance, a linear kernel requires HSIC to seek only second order dependence, whereas a polynomial kernel restricts HSIC to test for dependences of its degree. In terms of disentangled representation, to explore different latent factors behind a graph, we try to select independent representations that minimize HSIC by the inner product kernel.

Experiments

In this section, we empirically perform comparative evaluation of IPGDN model against a wide variety of state-of-the-arts on three node-related tasks: semi-supervise node classification, node clustering and graph visualization.

Experimental Setup

Datasets We conduct our experiments on three standard citation network benchmark datasets, whose statistics are listed in Table 1. Cora, Citeseer and Pubmed (Sen et al.

Table 1: Dataset statistics

Dataset	Cora	Citeseer	Pubmed
Nodes	2708	3327	19717
Edges	5429	4732	44338
Classes	7	6	3
Features	1433	3703	500
Training Nodes	140	120	60
Validation Nodes	500	500	500
Test Nodes	1000	1000	1000

Table 2: Semi-supervised classification results (%).

Method	Metrics	Datasets		
		Cora	Citeseer	Pubmed
MLP		55.1	46.5	71.4
ManiReg		59.5	60.1	70.7
SemiEmb		59.0	59.6	71.1
LP		68.0	45.3	63.0
DeepWalk		67.2	43.2	65.3
ICA		75.1	69.1	73.9
Planetoid	ACC	75.7	64.7	77.2
ChebNet		81.2	69.5	75.0
GCN		81.4	71.2	79.0
MoNet		81.7	-	78.8
GAT		83.0	72.9	78.7
DisenGCN		83.7	73.2	80.5
IPGDN (our)		**84.1**	**74.0**	**81.2**
MLP		54.3	45.6	69.4
ManiReg		60.2	61.2	70.2
SemiEmb		57.9	59.1	70.0
LP		66.2	46.2	62.8
DeepWalk		70.3	51.5	66.1
ICA		74.8	70.2	74.3
Planetoid	F1	73.6	63.5	77.0
ChebNet		80.9	66.2	75.5
GCN		81.2	71.1	78.8
MoNet		81.5	-	77.9
GAT		82.5	71.4	77.7
DisenGCN		83.6	72.0	80.9
IPGDN (our)		**84.2**	**72.8**	**81.7**

2008) are all for semi-supervised node classification and node clustering. We follow the experimental setup of (Yang, Cohen, and Salakhutdinov 2016). In all of these datasets, nodes correspond to documents and edges to (undirected) citations. Node features correspond to elements of a bag-of-words representation of a document. Each node has a class label, i.e., research area. We allow for only 20 nodes per class to be used for training. The predictive power of the trained models is evaluated on 1000 test nodes, and we use 500 additional nodes for validation purposes (the same ones as used by (Kipf and Welling 2017; Veličković et al. 2017)).

Baselines We compare with some state-of-the-art baselines, including node embedding methods and graph neural network based methods, to verify the effectiveness of the proposed IPGDN.

- MLP: It is a multi-layer perception as a baseline.
- ManiReg (Belkin, Niyogi, and Sindhwani 2006): It is a semi-supervised learning model base on manifold regularization which allows to exploit the geometry of the marginal distribution.
- SemiEmb (Weston et al. 2012):It is a semi-supervised embedding learning model incorporating deep learning techniques.
- LP (Zhu, Ghahramani, and Lafferty 2003) : It is a label propagation approach based on a Gaussian random field model.
- DeepWalk (Perozzi, Al-Rfou, and Skiena 2014): A random walk based network embedding method for graph.
- ICA (Lu and Getoor 2003): It is a link-based classification method which supports discriminative models describing both the link distributions and the attributes of linked objects.
- Planetoid (Yang, Cohen, and Salakhutdinov 2016): This model is an inductive, embedding based approach to semi-supervised learning. It uses the graph structure as a form of regularization during training while not using graph structural information during inference.
- ChebNet (Defferrard, Bresson, and Vandergheynst 2016): It is a spectral graph convolutional network by means of a Chebyshev expansion of the graph Laplacian, removing the need to compute the eigenvectors of the Laplacian and yielding spatially localized filters.
- GCN (Kipf and Welling 2017): It is a simple yet effective ChebNet model by restricting the filters to operate in a 1-step neighborhood around each node.
- MoNet (Monti et al. 2017): This model is an extended CNN architectures by learning local, stationary, and compositional task-specific features for non-Euclidean data.
- GAT (Veličković et al. 2017): It enhances GCN by introducing multi-head self-attention to assign different weights to different neighbors.
- DisenGCN (Ma et al. 2019): This model is a graph convolutional network which tries to disentangle the latent factors among the complex graph by a neighborhood routing mechanism.

Implementation Details In semi-supervised classification tasks, for a simple completion, we set the output dimension of DisenGCN and IPGDN's layer is a constant, i.e. $M \times \triangle f$ is the same in each layer, where M is the number of channels used by the layer and $\triangle f$ is the output dimension of each channel. In node clustering task, we follow GAT and set the output dimension of a graph neural network to be 64. For DeepWalk method, we set window size to 5, walk length to 100, walks per node to 40 and the number of negative samples to 5. In our model, we use $K = 4, \triangle f = 16$ for the test. Following (Ma et al. 2019), we set iterations of neighbourhood routing $T = 7$. We then tune the hyper-parameters of both our model's and our baselines' automatically using hyperopt (Bergstra, Yamins, and Cox 2013). Specifically, we run hyperopt for 200 trials for each setting, with the hyper-parameter search space specified as follows: the learning rate $\backsim$ loguniform $[e^{-8}, 1]$, the ℓ_2 regularization term $\backsim$ loguniform$[e^{-10}, 1]$, dropout rate $\in \{0.05, 0.10, ..., 0.95\}$,

Table 3: Node clustering results(%).

Datasets	Metrics	SemiEmb	DeepWalk	Planetoid	ChebNet	GCN	GAT	DisenGCN	IPGDN
Cora	ACC	60.8	62.2	67.2	71.9	73.5	75.2	75.5	**76.1**
	NMI	48.7	50.3	52.0	49.8	51.7	57.0	58.4	**59.2**
	ARI	41.5	40.8	40.5	42.4	48.9	54.1	60.4	**61.0**
	Precision	21.6	22.6	24.0	27.6	28.5	23.7	27.1	**28.1**
	F1	24.3	21.8	25.0	25.9	25.6	23.6	24.8	**25.3**
Citeseer	ACC	51.1	52.1	61.0	65.0	67.7	68.0	68.2	**68.9**
	NMI	31.2	30.5	41.2	42.6	42.8	43.1	43.7	**44.3**
	ARI	21.5	20.6	22.1	41.5	42.8	43.6	42.5	**43.0**
	Precision	22.9	23.6	20.9	21.6	20.5	21.1	23.8	**23.9**
	F1	20.0	17.0	19.5	20.8	19.9	20.3	22.6	**23.2**
Pubmed	ACC	62.3	65.2	64.6	75.2	75.6	76.3	77.0	**77.8**
	NMI	27.8	29.6	32.5	35.6	35.0	35.0	36.1	**37.0**
	ARI	35.2	36.6	33.9	38.6	40.9	41.4	41.6	**42.0**
	Precision	21.3	28.3	30.0	29.0	32.5	34.5	33.6	**34.0**
	F1	26.1	25.9	28.9	31.3	29.8	30.3	31.2	**32.1**

the number of layers $L \in \{1, 2, ..., 6\}$. Then with the best hyper-parameters on the validation sets, we report the averaged performance of 50 runs on each semi-supervised dataset, and 30 runs in node clustering task.

Classification Analysis

We report the results in Table 2. From the Table 2, we can see that IPGDN achieves the best performance in term of classification Accuracy (ACC) and F-score (F1). More specifically, on the one hand, disentangled approaches, i.e. both DisenGCN and our model, outperform holistic approaches such as ChebNet, GCN and GAT, which indicates that the disentangled feature is helpful for graph representation. On the other hand, we are able to improve upon DisenGCN by a margin of 1.7%, 0.9% and 1.7% on Cora, Citeseer and Pubmed respectively, suggesting that encouraging independence between latent factors may be beneficial.

Clustering Analysis

To further evaluate the embeddings learned from the above algorithms, we also conduct the clustering task. For all the compared algorithms, we obtain its node embedding via feed forward when the model is trained. Then we use the KMeans to perform node clustering and the number of clusters K is set to the number of classes. The same ground-truth as in node classification analysis is utilized. Following (Pan et al. 2018), we employ five metrics to validate the clustering results: Accuracy (ACC), Normalized Mutual Information (NMI), Precision, F-score (F1) and Average Rand Index (ARI). Since the performance of KMeans is affected by initial centroids, we repeat the process for 20 times and report the average results in Table 3.

As can be seen in Table 3, we can see that IPGDN performs consistently much better than all baselines. Also, graph neural network based algorithms usually achieve better performance. Besides, with the constraint of independence, IPGDN performs significantly better than DisenGCN

. It shows that the proposed IPGDN can learn a more meaningful node embedding via enforcing independence between latent representations.

Visualization

To present a better intuitively comparation, we provide a visualization of the t-SNE (Maaten and Hinton 2008) transformed feature representations. Specifically, we learn the node embedding based on the proposed model and project the learned embedding into a 2-dimensional space. Here we utilize t-SNE to visualize the author embedding on Cora dataset and coloured the nodes based on their research areas.

Base on Figure 2, we can see that IPGDN performs much better than other graph neural networks and slightly better than DisenGCN. It demonstrates that the embedding learned by IPGDN has high intra-class similarity and separates the article in different research area with distinct boundaries by encouraging independence among different latent representations. On the contrary, GCN and GAT which get holistic feature from their neighbors do not perform well. The authors belong to different research areas are mixed with each other.

Hyperparameter Sensitivity

We show parameter adjustment and algorithm convergence on Cora as an example in Figure 3 and 4, respectively. From the Figure 3, it can be seen that the parameter of independence term is relatively robust since the performance is stable while λ is chosen in a wide range. Specifically, the promising performance can be expected when the parameter λ is given in a range (e.g., $[10^{-5}, 10^{-6}]$). Yet the classification performance decreases dramatically when the value of λ exceeds a upper limit (e.g., 1.25×10^{-5}), which demonstrate that it is not good to overemphasize the independence between latent factors. From Figure 4, we can see that IPGDN converges within a small number of iterations, which empirically proves the efficiency of our model.

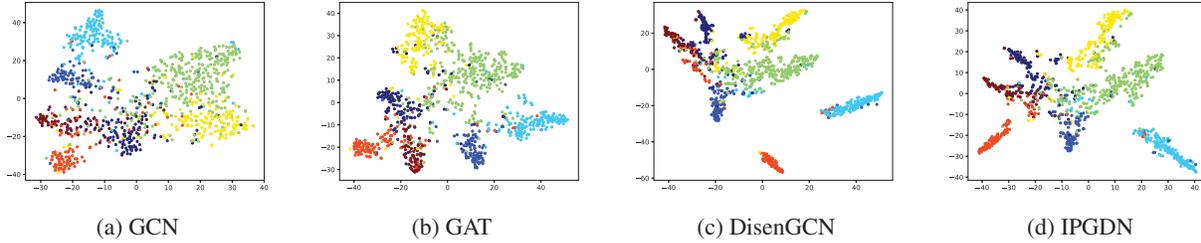| | | | |
| (a) GCN | (b) GAT | (c) DisenGCN | (d) IPGDN |

Figure 2: Visualization embedding on Cora. Each point indicates one author and its color indicates the research area. We can see that the embedding learned by IPGDN has high intra-class similarity and separates articles in different research area with distinct boundaries by encouraging independence among different latent representations.
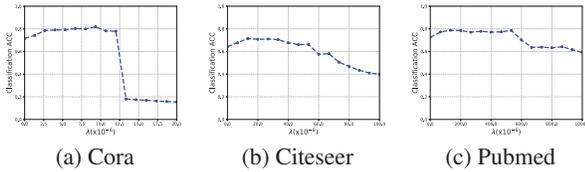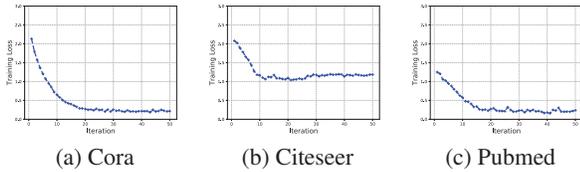

| (a) Cora | (b) Citeseer | (c) Pubmed |

Figure 3: Parameter tuning on datasets.

| (a) Cora | (b) Citeseer | (c) Pubmed |

Figure 4: Convergence results on datasets.

Related work

Inspired by the huge success of convolutional networks in the computer vision domain, recently there has been a large number of methods trying to apply convolution on non-Euclidean graph data. These approaches are under the umbrella of graph convolutional networks. Generally speaking, GCNs fall into two categories:spectral-based and spatial-based. Bruna et al. (Bruna et al. 2014) propose the fist prominent work on GCNs, which designs a variant of graph convolution in the light of spectral graph theory. Subsequently, a large of increasing improvements, extensions, and approximations on spectral-based graph convolutional networks (Defferrard, Bresson, and Vandergheynst 2016; Kipf and Welling 2017; Levie et al. 2018) have been proposed. As spectral methods usually deal with the whole graph simultaneously and are hard to scale or parallel to large graphs, spatial-based graph convolutional networks have rapidly developed recently (Hamilton, Ying, and Leskovec 2017; Monti et al. 2017; Gao, Wang, and Ji 2018). These approaches directly formulate graph convolutions as aggregating feature information from neighbors in the graph domain.

Furthermore, their computation can be largely decreased together with sampling strategies, which can be performed in a batch of nodes instead of the whole graph (Hamilton, Ying, and Leskovec 2017; Gao, Wang, and Ji 2018). Hence, they have the potential to improve efficiency.

Another line of works related to ours is the disentangled representation learning. Early works that have demonstrated disentanglement in limited settings include (Desjardins, Courville, and Bengio 2012), and several prior researches have addressed the problem of disentanglement in supervised or semi-supervised settings (Kingma et al. 2014; Kulkarni et al. 2015). Recently, disentangled representation learning has gained considerable attention, in particular in the field of image representation learning (Alemi et al. 2016). It is designed for learning representations to separate the explanatory factors of variations behind the data. These representations have proven to be more resilient to complex variants (Bengio, Courville, and Vincent 2013) and can enhance enhanced generalization capabilities and improve the robustness of confrontational attacks (Alemi et al. 2016). Furthermore, The disentangled representations are inherently easier to interpret and hence may be helpful for debugging and auditing (Doshi-Velez and Kim 2017). More recently, there are several recent works that constrain the form of this decomposition to capturing purely independent factors of variation in unsupervised generative models (Cao et al. 2015; Chen et al. 2018; Kim and Mnih 2018). They typically evaluate the disentanglement using purpose-built, artificial, data and their generative factors are themselves independent by construction. However, these approaches are mainly applied to traditional data but not to graph data.

Conclusions

In this paper, we propose a novel independence promoted graph disentangled networks for graph representations. In our approach, disentangled representation learning and independence measure among latent representations are integrated into a unified framework. Specifically, we present a HSIC scheme to regularize latent representations and enforce them to be independent. The regularized module is jointly learned with a graph convolutional networks to produce the informative representations. Experimental results demonstrate that our algorithm IPGDN outperform the state-

of-the-arts in several graph node-related tasks.

Acknowledgments

References

Alemi, A. A.; Fischer, I.; Dillon, J. V.; and Murphy, K. 2016. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*.

Belkin, M.; Niyogi, P.; and Sindhwani, V. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research* 7(11):2399–2434.

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(8):1798–1828.

Bergstra, J.; Yamins, D.; and Cox, D. D. 2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *ICML*.

Bruna, J.; Zaremba, W.; Szlam, A.; and Lecun, Y. 2014. Spectral networks and locally connected networks on graphs. In *ICLR*.

Cao, X.; Zhang, C.; Fu, H.; Liu, S.; and Zhang, H. 2015. Diversity-induced multi-view subspace clustering. In *CVPR*, 586–594.

Chen, T. Q.; Li, X.; Grosse, R.; and Duvenaud, D. 2018. Isolating sources of disentanglement in variational autoencoders. In *NeurIPS*.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, 3844–3852.

Desjardins, G.; Courville, A. C.; and Bengio, Y. 2012. Disentangling factors of variation via generative entangling. *arXiv: Machine Learning*.

Doshi-Velez, F., and Kim, B. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Fout, A.; Byrd, J.; Shariat, B.; and Ben-Hur, A. 2017. Protein interface prediction using graph convolutional networks. In *NeurIPS*, 6530–6539.

Fukumizu, K.; Gretton, A.; Sun, X.; and Schölkopf, B. 2008. Kernel measures of conditional dependence. In *NeurIPS*, 489–496.

Gao, H.; Wang, Z.; and Ji, S. 2018. Large-scale learnable graph convolutional networks. In *KDD*, 1416–1424.

Gretton, A.; Bousquet, O.; Smola, A.; and Schölkopf, B. 2005. Measuring statistical dependence with hilbert-schmidt norms. In *International Conference on Algorithmic Learning Theory*, 63–77.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *NeurIPS*, 1024–1034.

Kim, D. H., and Mnih, A. 2018. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P.; Mohamed, S.; Rezende, D. J.; and Welling, M. 2014. Semi-supervised learning with deep generative models. In *NeurIPS*, 3581–3589.

Kipf, T., and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. *ICLR*.

Kulkarni, T. D.; Whitney, W. F.; Kohli, P.; and Tenenbaum, J. 2015. Deep convolutional inverse graphics network. In *NeurIPS*, 2539–2547.

Levie, R.; Monti, F.; Bresson, X.; and Bronstein, M. M. 2018. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing* 67(1):97–109.

Li, Q.; Han, Z.; and Wu, X. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, 3538–3545.

Lu, Q., and Getoor, L. 2003. Link-based classification. In *ICML*, 496–503.

Ma, J.; Cui, P.; Kuang, K.; Wang, X.; and Zhu, W. 2019. Disentangled graph convolutional networks. In *ICML*.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research* 9(11):2579–2605.

Monti, F.; Boscaini, D.; Masci, J.; Rodola, E.; Svoboda, J.; and Bronstein, M. M. 2017. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, 5115–5124.

Pan, S.; Hu, R.; Long, G.; Jiang, J.; Yao, L.; and Zhang, C. 2018. Adversarially regularized graph autoencoder for graph embedding. In *IJCAI*, 2609–2615.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *KDD*, 701–710.

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

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Weston, J.; Ratle, F.; Mobahi, H.; and Collobert, R. 2012. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*. Springer. 639–655.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2019. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*.

Yang, Z.; Cohen, W. W.; and Salakhutdinov, R. 2016. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*.

Zhang, Z.; Cui, P.; and Zhu, W. 2018. Deep learning on graphs: A survey. *arXiv preprint arXiv:1812.04202*.

Zhu, X.; Ghahramani, Z.; and Lafferty, J. D. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 912–919.