ProtoHG: Prototype-Enhanced Hypergraph Learning for Heterogeneous Information Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

The variety and complexity of relations in real world data lead to Heterogeneous Information Networks (HINs). Capturing the semantics from such networks requires approaches capable of utilizing the full richness of the HINs. Existing methods for modeling HINs employ techniques originally designed for graph neural networks in combination with HIN decomposition analysis, especially using manually predefined metapaths. In this paper, we introduce a novel hypergraph learning approach for node classification in HINs. Using hypergraphs instead of graphs, our method captures higher-order relationships among heterogeneous nodes and extracts semantic information without relying on metapaths. Our method leverages the power of prototypes to improve the robustness of the hypergraph learning process, and we further discuss the advantages that our method can bring to scalability, which due to their expressiveness is an important issue for hypergraphs. Extensive experiments on three real-world HINs demonstrate the effectiveness of our method.

1 introduction

Many real-world data collections can be effectively formulated as HINs, where different types of nodes and edges embody multiple types of entities and relations. For example, as shown in Figure 1, an academic network has several types of nodes: Paper, Author, and Subject, as well as different types of relations, each associated with different semantics, such as Author$\xrightarrow{\text{writes}}$Paper, Paper$\xrightarrow{\text{belongs to}}$Subject. These relations can be aggregated to give rise to higher-order semantic associations. Examples are the triadic (ternary) relationships Paper-Author-Paper representing a co-author relationship and Paper-Subject-Paper conveying a same topic connection. Modeling the relational and semantic richness of HINs requires the development of specialized models to provide effective analysis and interpretation.

Recent years have brought a rapid development of Graph Neural Networks (GNNs) in pursuit of performance improvement in graph representation learning [24, 25]. GNNs are primarily designed for homogeneous graphs associated with a single type of nodes and edges, and follow a neighborhood aggregation scheme to capture the structural information of a graph [13, 21]. Thus, most GNNs are not well-equipped to deal with HINs, which also have rich semantic information induced by different types of nodes, as well as by varied structural information [20].

Various Heterogeneous Graph Neural Networks (HGNNs) have been introduced as effective tools for the extraction and incorporation of semantic knowledge, yielding remarkable performance in representation learning for HINs [27, 22]. With regard to their approach to relation handling, these techniques can be broadly grouped into two categories: *Metapath-based methods* and *Metapath-free methods*. Metapath-based methodologies leverage metapath—sequential arrangements of node and edge types. Due to the semantic expressiveness of metapaths, many techniques initially extract

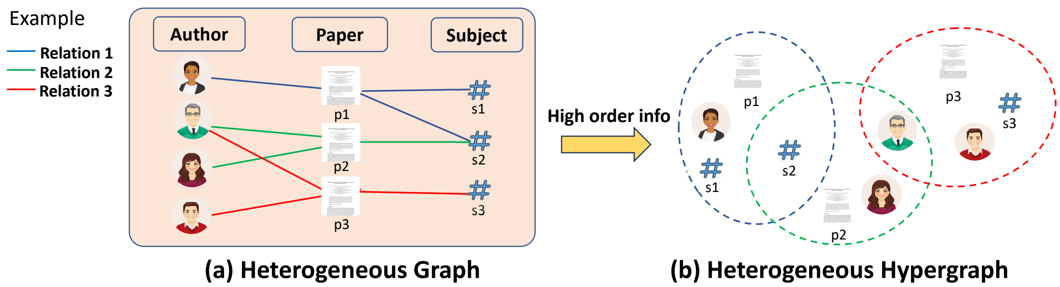

Figure 1: **Comparison between conventional heterogeneous graphs (e.g., an academic network) and their corresponding heterogeneous hypergraph**. In a conventional heterogeneous graph, different nodes are connected by different pairwise links and cannot explicitly capture the high-order complex relation among those nodes. For example, in the academic network, the interactions are not only among authors creating papers and papers belonging to a subject period but also high-order information, e.g., *several authors collaborated on a paper which spans multiple subjects*.

various substructures from the original HINs, each possessing distinct semantic characteristics. This extraction process is guided by a set of predefined metapaths, which subsequently serve as the topology for representation learning on these substructures [5, 31, 23, 7]. Although Metapath-based methods have achieved state-of-the-art performance on plenty of tasks, they are usually limited in that (1) Metapaths have to be specified in advance, requiring domain-specific knowledge or even exhaustive enumeration of schemes, strategies often associated with prohibitively high manual and computational costs. (2) They primarily focus on pairwise connections, and it is hard to capture the complex higher-order interactions implicitly contained in HINs. Metapath-free methods are proposed to address the first limitation. They aggregate information from neighboring nodes by an attention mechanism or a usually relation-dependent graph transformer. This category of methods operates by using one-hop relations as input to the layers of a GNN and subsequently stacking multiple layers to facilitate the learning of multi-hop relations [29, 10, 16]. However, this strategy can be challenged by the intricacies inherent in capturing higher-order relations.

To deal with the complexity of higher-order relations, in this paper, we present a hypergraph learning approach for node classification, which aims to preserve the high-order relations present in HINs and simultaneously capture the semantic information in them. Our model leverages the power of a hypergraph representation, a structure that generalizes graphs by allowing edges to connect more than two nodes. By representing higher-order relationships more explicitly, hypergraphs provide a natural framework for capturing complex dependencies and group information. Traditional hypergraph approaches simplify a hypergraph to a regular graph, e.g., by applying star or clique expansion [6, 26]. This simplification facilitates learning hypergraph representations, but inherently leads to loss of information. Recently, there have been works on applying deep neural network message passing to propagate vertex-hyperedge-vertex information through the hypergraph. This allows direct learning from the hypergraph topology [2, 32, 8]. This way of learning avoids reducing the high-order relations into pairwise ones, hence no information loss, and provides a new way to model semantic information without relying on metapaths. In line with previous studies on HINs [23, 16], for practical reasons we specifically focus on utilizing symbolic relations. This means we do not incorporate node content similarity to define network structure.

Modeling symbolic relations with current hypergraph models is known to be sensitive to noisy information in the nodes and hyperedges [2]. Hence, our method utilizes prototypes, representative nodes that summarize groups or similar entities in the data structure, in two ways to regularize the hypergraph learning process. First, we design a prototype-based hyperedge regularization method, where each hyperedge serves as a prototype and forces its nodes to be close to it in the embedding space to stabilize the optimization. Second, to further improve the robustness of our model for node classification against noise or small changes in the initial samples, we make the prototypes learnable and learn multiple prototypes to represent different classes.

Our contributions are three-fold:

1. We introduce a novel approach that uses the power of hypergraphs and prototypes for node classification in HINs. The hypergraphs allow for higher-order relationships among nodes to capture complex semantic dependencies and group information in the data.

2. We show how to use prototypes, being representative landmarks, to enhance the robustness of hypergraph model learning for HINs.
3. We demonstrate the effectiveness of the proposed method through experiments conducted on two real-world HINs benchmarks.

2 Related work

In this section, we reflect on related work about heterogeneous information networks in Section 2.1 and hypergraph learning in Section 2.2.

2.1 Heterogeneous Information Networks

Different from homogeneous networks, heterogeneous networks consist of different types of nodes and edges. Most recent methods for analyzing heterogeneous graphs concentrate on decomposing the network into homogeneous sub-graphs and deploying GNNs. Metapath-based methods first extract substructures according to hand-crafted metapaths and then learn node representations based on these sub-structures. For instance, HAN is a representative method that applies hierarchical attention to aggregate information from metapath-based neighbors [23]. MAGNN [7] improves on that by simultaneously aggregating information from the intermediate nodes. In contrast, Metapath-free methods adhere to a different paradigm, aggregating messages from neighbors within the immediate one-hop vicinity akin to traditional GNNs, disregarding the specific node types involved. They augment this process with additional modules, such as attention mechanisms, to encode semantic information like node types and edge types into the propagated messages, thereby enriching the data representation. GTN [29] can discover valuable meta-paths automatically with the intuition that a metapath neighbor graph can be obtained by multiplying the adjacency matrices of several sub-graphs. However, GTN consumes a gigantic amount of time, and memory, e.g., 400 $\times$ time and 120 $\times$ the memory of Graph Attention Networks (GATs) [17]. HGT [10] builds on the transformer to handle large academic heterogeneous graphs, focusing on web-scale graphs via a graph sampling strategy. HGB [16] instead uses a GAT network as the backbone, with the help of learnable edge-type attention, L2 normalization, and residual attention for node representation generation. All of the above methods focus on the pairwise relations in the network and ignore high-order relations in HINs, inherently leading to semantic information loss.

2.2 Hypergraph Learning

Hypergraph learning is related to graph learning since a hypergraph is a graph generalization that allows edges to connect 2 to n nodes, where n is the number of nodes/vertices. Hypergraph learning was first introduced in [33] and can be seen as a propagation process along the hypergraph structure in analyzing categorical data with complex relationships. In its original form it conducts transductive learning and aims to minimize the label difference among vertices having stronger connections in the hypergraph. Inspired by the immense success of deep learning, recently effective approaches to deep learning on hypergraphs have been proposed [9]. Hypergraph neural networks design a vertex-hyperedge-vertex information propagating pattern to iteratively learn the data representation, and some of them have the ability of inductive learning [6, 4, 11, 2, 32, 8]. Recently, there has been growing interest in utilizing hypergraphs to model structured data in HINs [15, 14]. However, the above hypergraph approaches for HINs necessitate predefined metapaths with domain knowledge as a foundation for constructing hyperedges. This reliance on metapaths introduces several challenges, including the need for manual specification, making them less suitable for diverse datasets. Additionally, capturing high-order information directly from the data can be challenging within this metapath-centric paradigm.

3 Methodology

Here, we present our methodology closely following Figure 2. We start by providing some general notation.

Notations In this work, we focus on the heterogeneity of nodes. A heterogeneous hypergraph is represented as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{T}_{\sqsubseteq}\}$, where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ is the node set, $\mathcal{E} = \{e_1, e_2, ..., e_m\}$

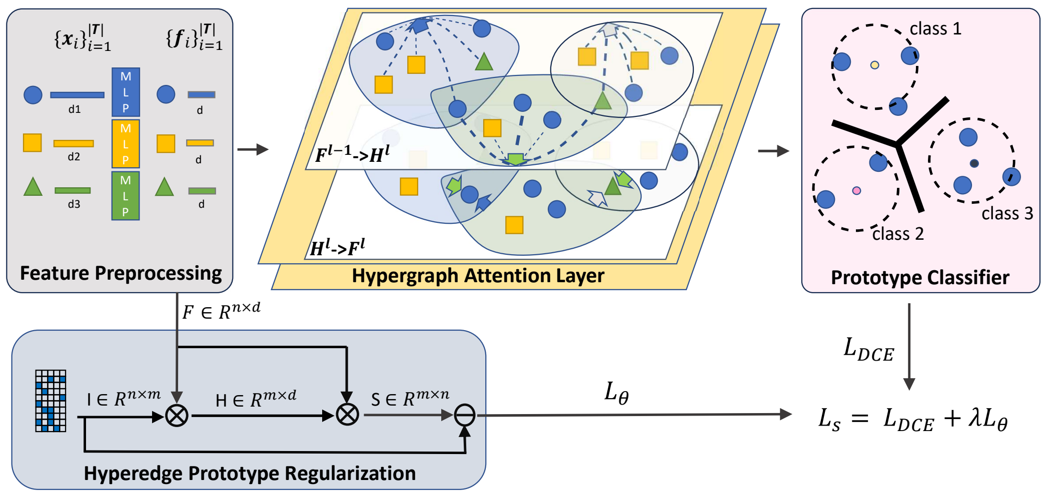

Figure 2: **An illustration of prototype-enhanced hypergraph learning model**. First, linear layers map heterogeneous nodes with varying embedding lengths into a shared space. Then, high-order message passing occurs among different nodes based on the topology of the hyperedges. Hyperedge prototype regularization constrains node embeddings based on their proximity to their respective hyperedges. Finally, nodes are classified by learnable prototype-based classifiers according to their representations.

represents the set of hyperedges, and $\mathcal{T}$ is the set of node types. Each hyperedge has 2 or more nodes. When $|\mathcal{T}| \geq 2$, the hypergraph is heterogeneous. The relationship between nodes and hyperedges can be represented by an incidence matrix $I \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{E}|}$, with entries defined as:

$$I(v,e) = \begin{cases} 1, & \text{if } v \in e \\ 0, & \text{otherwise} \end{cases}$$

Let $D_e \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$ denotes the diagonal matrices containing hyperedge degrees, where $D_e(i,i) = \sum_{u\in\mathcal{V}} I(u,i)$. The hyperedge degree is a valuable parameter for normalization purposes.

Problem Statement. In the context of HIN, we capture and retain their implicit high-order relations, effectively forming a heterogeneous hypergraph denoted as $\mathcal{G}$. Then we aim to learn a low dimensional representations $\mathbf{f} \in \mathbb{R}^d$ for nodes in $\mathcal{G}$ with $d \ll |v|$ while fully considering both the high-order relations and heterogeneity implied in $\mathcal{G}$. This representation can be used for downstream predictive applications such as node classification.

3.1 Feature Preprocessing

Due to the heterogeneity of nodes, different types of nodes are originally represented in different semantic/feature spaces associated with their specific probability distributions. Therefore, for each node type t_i, we design a learnable type-specific transformation matrix O_{t_i} to project the heterogeneous nodes with varying embedding lengths into a space with the same dimensions. This allows for messages to be passed among them in a common space throughout the training process. The projection process can be represented as follows:

$$\mathbf{f}_i = O_{t_i} \cdot \mathbf{x}_i \tag{1}$$

where $\mathbf{f}_i$ and $\mathbf{x}_i$ are the projected and original features of node v_i, respectively. With the learnable type-specific projection operation, we address the network heterogeneity introduced by the node type. Following this transformation, projected features of all nodes are unified to have the same dimensions, facilitating the subsequent aggregation process in the next model component.

3.2 Hypergraph Attention Layer

To capture the heterogeneous high-order context information of a hypergraph, we employ node and hyperedge attention mechanisms. For layer l_0, we define node representation $F^0 = \{\mathbf{f}_1^{l_0}, \mathbf{f}_2^{l_0}, ..., \mathbf{f}_n^{l_0}\}$ and incidence matrix $I \in \mathbb{R}^{n \times m}$. The target of the heterogeneous hypergraph layer l is to update node representations through hypergraph message passing by calculating hypergraph attention.

Node-level Attention $\mathbf{h}_j^l = \mathrm{AGGR}_{\text{node}}^{l-1}\left(\left\{\mathbf{f}_k^{l-1} \mid \forall v_k \in e_j\right\}\right)$. In this step, we calculate hyperedge representation $H^l = \{\mathbf{h}_1^l, \mathbf{h}_2^l, ..., \mathbf{h}_m^l\}$ given node embedding $F^{l-1} \in \mathbb{R}^{n \times d}$

$$\mathbf{h}_j^l = \sigma\left(\sum_{v_k \in e_j} \alpha_{jk} W_h \mathbf{f}_k^{l-1}\right), \tag{2}$$

where σ is a nonlinearity, in our case $\mathrm{LeakyReLU}$, and W_h is a trainable matrix. α_{jk} is a coefficient to control how much information is contributed from node v_k to hyperedge e_j, which can be computed by:

$$\begin{aligned} \alpha_{jk} &= \frac{\exp\left(\mathbf{a}_1^{\mathrm{T}} \mathbf{u}_k\right)}{\sum_{v_p \in e_j} \exp\left(\mathbf{a}_1^{\mathrm{T}} \mathbf{u}_p\right)}, \\ \mathbf{u}_k &= \mathrm{LeakyReLU}\left(W_h \mathbf{f}_k^{l-1}\right), \end{aligned} \tag{3}$$

where $\mathbf{a}_1^T$ is a trainable weight vector.

Hyperedge-level Attention $\mathbf{f}_i^l = \mathrm{AGGR}_{\text{hyperedge}}^l\left(\mathbf{f}_i^{l-1}, \left\{\mathbf{h}_k^l \mid \forall e_k \in \int_i\right\}\right)$. Then with all hyperedge representations $\{\mathbf{h}_j^l \mid \forall e_j \in \int_i\}$, where $\int_i$ is the set of associated hyperedges given vertex v_i, we update node representation $F^l = \{\mathbf{f}_1^l, \mathbf{f}_2^l, ..., \mathbf{f}_n^l\}$ based on updated hyperedge representations H^l.

$$\begin{aligned} \mathbf{f}_i^l &= \mathrm{LeakyReLU}\left(\sum_{e_k \in \int_i} \beta_{ik} W_e \mathbf{h}_k^l\right), \\ \beta_{ik} &= \frac{\exp\left(\mathbf{a}_2^{\mathrm{T}} \mathbf{v}_k\right)}{\sum_{v_q \in e_i} \exp\left(\mathbf{a}_2^{\mathrm{T}} \mathbf{v}_q\right)}, \\ \mathbf{v}_k &= \mathrm{LeakyReLU}(\left[W_h \mathbf{f}_i^{l-1} \| W_e \mathbf{h}_k^l\right]), \end{aligned} \tag{4}$$

where $\mathbf{f}_i$ is the update representation to node v_i and W_e is a weight matrix. β_{ik} denotes the attention coefficient of hyperedge $\mathbf{h}_k$ to node $\mathbf{f}_i$. $\mathbf{a}_2^T$ is another weight vector measuring the importance of the hyperedges. $||$ here is the concatenation operation.

We extend hypergraph attention (HAT) into multi-head hypergraph attention(MH-HAT) by concatenating multiple HATs together to expand the model's representation ability.

$$\text{MH-HAT}(F, I) = \Big\|_{i=1}^{K} \sigma(\mathrm{HAT}_i(F, I)). \tag{5}$$

By harnessing the MH-HAT structure, we effectively capture the high-order relationship and semantic information present in a HIN.

3.3 Learnable Prototype Classifier

To enhance the robustness of hypergraph message passing, we replace the softmax layer used in conventional neural networks. Instead, we utilize a node classification approach rooted in learnable prototypes. Here, we assume each class has an equal number of k prototypes, and this assumption can be effortlessly relaxed in other use cases. The prototypes are denoted as $\mathbf{m}_{ij}$ where $i \in \{1, 2, ..., c\}$

represents the index of the category and $j \in \{1, 2, ..., k\}$ represents the index of the prototypes in each category.

In the prediction phase, utilizing the feature vector $\mathbf{f}$ generated by the heterogeneous message passing module, we first employ a linear layer to integrate node representations, represented as $g(\mathbf{f}^l; \theta)$. Then, we measure the distance of an input pattern to all prototypes and classify it into the category associated with the nearest prototype.

$$\begin{aligned} \hat{y} &= \arg \min_{i \in c, j \in k} \|\mathbf{m}_{ij} - \mathbf{z}\|, \\ \mathbf{z} &= g(\mathbf{f}; \theta). \end{aligned} \tag{6}$$

We use distance-based cross-entropy loss (DCE) [28] to measure the similarity between the samples and the prototypes. Thus, for a sample characterized by feature $\mathbf{x}$ and category y, the probability of $(\mathbf{x}, y)$ belonging to the prototype $\mathbf{m}_{ij}$ can be measured by the distance between them:

$$p(y|\mathbf{z}) = p(\mathbf{z} \in \mathbf{m}_{ij} \mid \mathbf{z}) = -\|\mathbf{z} - \mathbf{m}_{ij}\|_2^2. \tag{7}$$

Based on the probability of $\mathbf{x}$, we can define the cross entropy (CE) in our framework as:

$$L_{DCE} = l((\mathbf{z}, y); \theta, M) = -\log p(y|\mathbf{z}) \tag{8}$$

3.4 Hyperedge Regularizated Training

Hypergraph modeling has been observed to exhibit heightened sensitivity to noise [2], and the presence of heterogeneity further amplifies this sensitivity. To mitigate the destabilizing impact of noisy nodes, we introduce a novel hyperedge-based regularization technique, tailored to enhance training stability. In this regularization scheme, each hyperedge assumes the role of a prototype, imposing constraints that compel its associated nodes to maintain a defined proximity (could be incidence matrix I or a learned attention map) in the embedding space. This strategic approach aims to curtail the influence of noise on the message-passing dynamics along the hypergraph topology.

As mentioned in Notations, we use incidence matrix $I \in \mathbb{R}^{n \times m}$ to denote the presence of nodes in different hyperedges. Hence, I indicates the relations between nodes and hyperedges. Before applying hypergraph message passing, we can get the hyperedge representation $H \in \mathbb{R}^{m \times d}$ by node representation $F \in \mathbb{R}^{n \times d}$ and incidence matrix I, respectively. Then, we define our regularization normalized by inverse hyperedge degree $D_e^{-1} \in \mathbb{R}^m$ as:

$$\begin{aligned} L_\theta &= I - FH^T D_e^{-1}. \\ &= I - FF^T I D_e^{-1} \end{aligned} \tag{9}$$

Then, the total loss function is defined as:

$$L_s = L_{DCE} + \lambda L_\theta, \tag{10}$$

Where λ denotes the weight to balance the above two tasks. Through the introduced regularization, we mitigate the influence of noise and further enhance the model's robustness throughout the training process.

Table 1: Statistics of HIN datasets

Dataset	Node	Node Type	Hyperedges	Target	Class
ACM	10,942	paper, author, subject	3025	paper	3
DBLP	26,128	paper, author, venue, term	4057	author	4

4 Experimental Setup

In this section, we introduce the datasets and evaluation metrics, followed by a detailed explanation of method implementations.

4.1 Datasets

We perform experimental evaluations on two well-established heterogeneous academic structure datasets presented in Table 1. The datasets, DBLP and ACM, are obtained from the Heterogeneous Graph Benchmark (HGB) [16]. We adopt them here to facilitate comparison with a broader range of existing methods. We strategically connect target nodes (with labels to be classified) using pertinent relationships to construct hyperedges within the datasets. For instance, in the ACM dataset, the target nodes are research papers, and we form hyperedges by linking each paper to its respective authors, references, and venue. Similarly, within the DBLP dataset, authors are targets and connected to their associated papers, the venue, and their terms.

4.2 Implementation Details

Node classification in our experiments follows a transductive setting using the train/val/test split from the HGB benchmark [16]. We implement our method in PyTorch and optimize it using the Adam optimizer [12] with an initial learning rate of 0.001. Hyperparameter settings for all baselines are consistent with those reported in their original papers. Early stopping is applied with a patience rate of 60 and all reported scores are averages from 5 different random seeds. We use hidden dimension $= 64$, layer $= 3$, $K = 1$ to ACM and DBLP datasets. For λ value, we use 10^{-3} for DBLP, and 10^{-6} for ACM. Our experiments reveal that the optimal λ value is contingent on the dataset's heterogeneity. Specifically, datasets with a greater variety of node types tend to benefit from larger λ values.

5 Experimental Results

In this section, several experiments and their results are discussed to answer the following research questions:

1. Is deep learning on heterogeneous hypergraphs effective in node classification for HINs?
2. What are the benefits of the prototype classifier and prototype-based hyperedge regularization in node classification for HINs?
3. Can prototypes facilitate the interpretation of HINs?

5.1 Heterogeneous Hypergraph Modelling

To answer whether heterogeneous hypergraphs are effective in node classification for HIN data, we conduct a transductive node classification experiment. Table 2 show the results of our methods on three datasets compared with the results of baselines, including metapath-based methods (RGCN [19], HetGNN [31], HAN [23], MAGNN [7]) and metapath free-based methods (GTN [29], HetSANN [34], HGT [10], He-GCN [13], He-GAT [21]). Our approach is the overall best performer. A nuanced picture emerges when we delve into the differences between Micro-F1 and Macro-F1 scores. While it undeniably outperforms alternative methods in terms of Micro-F1 scores, the advantages of our approach concerning the Macro-F1 measure are not as readily apparent. This observation implies potential challenges in effectively handling imbalanced data. Varying number of nodes in imbalanced datasets lead to imbalanced hyperedge density, which results in data sparsity issues, challenging learning meaningful patterns for the underrepresented class nodes.

5.2 Learnable Prototype Classifier and Prototype Regularization for Hypergraph Modeling

To understand the contribution of prototype regularization and the prototype classifier to the model's overall performance, we performed an ablation study, where we trained our methods with and without the prototype classifier and prototype regularization. Results in Table 3 show that training a heterogeneous hypergraph model with prototype classifier and regularization can improve both the Macro-F1 and Micro-F1 performance of HIN modeling. The result also shows that a learnable prototype classifier contributes more to the performance than prototype regularization. For a more intuitive comparison, we learn the node embedding space on the proposed model and project the learned embedding into a 2-dimensional space by UMAP [18]. Figure 3 shows that there is a clearer

Table 2: **Semi-supervised node classification task results on ACM and DBLP datasets**. Bold font denotes the best-performing results. Each model was run five runs, and we report the mean ± standard deviation. Baseline performance metrics are extracted from the HGB benchmark [16], with 'Mph' representing the metapath.

		ACM		DBLP	
		Micro-F1	Macro-F1	Micro-F1	Macro-F1
w/ Mph	RGCN	91.41±0.75	91.55±0.74	92.07±0.50	91.52±0.50
	HetGNN	86.05±0.25	85.91±0.25	92.33±0.41	91.76±0.43
	HAN	90.79±0.43	90.89±0.43	92.05±0.62	91.67±0.49
	MAGNN	90.77±0.65	90.88±0.64	93.76±0.45	93.28±0.51
w/o Mph	GTN	91.20±0.71	91.31±0.70	93.97±0.54	93.52±0.55
	RSHN	90.32±1.54	90.50±1.51	93.81±0.55	93.34±0.58
	HetSANN	89.91±0.37	90.02±0.35	80.56±1.50	78.55±2.42
	HGT	91.00±0.76	91.12±0.76	93.49±0.25	93.01±0.23
	He-GCN	92.12±0.23	92.17±0.24	91.47±0.34	90.84±0.32
	He-GAT	92.19±0.93	92.26±0.94	93.39±0.30	**93.83 ± 0.27**
	Ours	**93.30 ± 0.59**	**93.49 ± 0.56**	**94.12 ± 0.65**	93.09±0.44

Table 3: **Performance comparison between the model with/without Prototype Regularization and Prototype Classifier**. The $\surd$ indicates the model with our proposed modules. The $\times$ indicates the model with a standard softmax classifier and no regularization.

Module		ACM	
P-Regularization	P-Classifier	Micro-F1	Macro-F1
$\times$	$\times$	90.42±1.36	90.61±1.30
$\surd$	$\times$	91.04±0.67	91.21±0.68
$\times$	$\surd$	93.05±0.22	93.15±0.23
$\surd$	$\surd$	**93.30 ± 0.59**	**93.49 ± 0.56**

inter-class decision boundary for classes with learnable prototype classifiers and regularization, and intra-class clusters are also more compact.

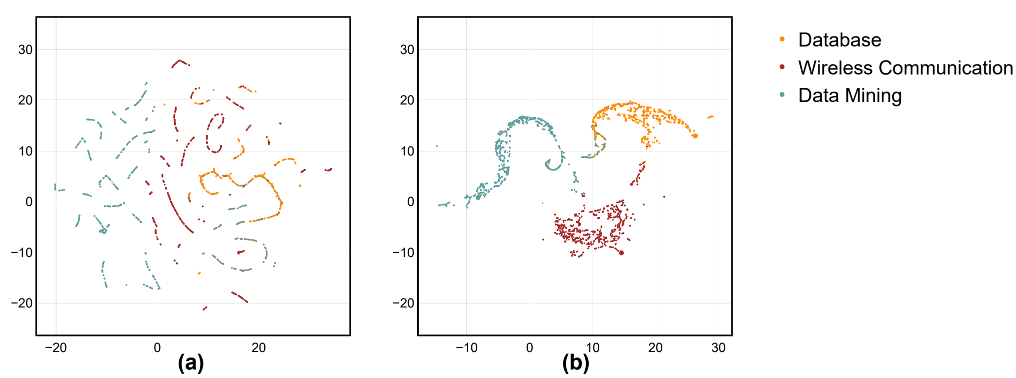

Figure 3: **UMAP representation vectors of nodes on ACM dataset.** Sub-figures (a) and (b) indicate paper representations learned without and with prototypes. The color represents paper class. We observe that when using prototype-enhanced learning, the distribution of the node representations is more clustered for paper class.

5.3 Prototype for Interpreting HINs

To explore the interpretative capabilities of prototype components within Heterogeneous Information Networks (HINs), we visualized the embedding space of sample subjects and areas. In Figure 4, we illustrate the neighboring subjects associated with each class prototype. Notably, in the embedding space, we observe greater diversity among the Paper classes that have a more interdisciplinary

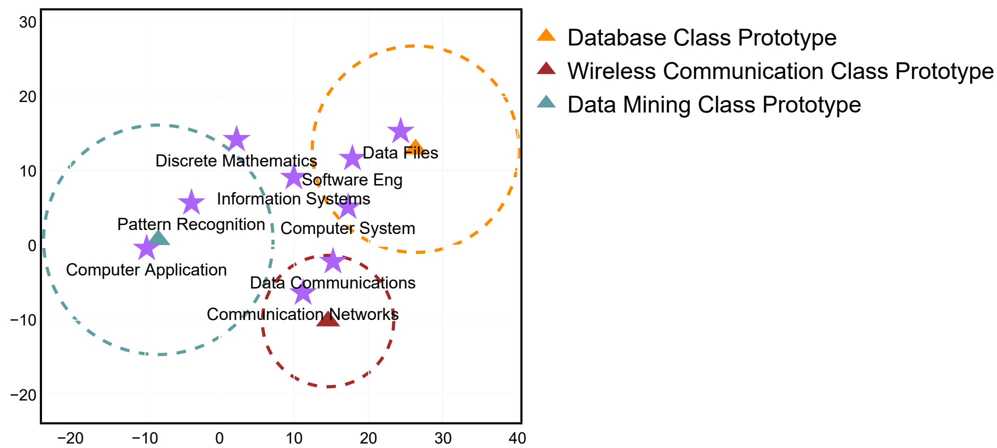

Figure 4: **Qualitative analysis of three learned area prototypes**. Examining three area prototypes, we observe that nearby segmented subject. Triangle means the class prototype and circle distance means the distance that covers 70% of papers for each class. We observe that although the target nodes to be classified are papers, the subjects associated with these papers closely align with their respective class categories. For instance, we find that *data communication* is closely linked to *Wireless Communication*, while *Data Files* exhibits a proximity to the *Database* class.

nature, such as *Data Mining*, as opposed to those like *Wireless Communication*. Furthermore, certain topics that span multiple classes tend to reside on the boundaries between different paper classes. Consequently, our prototypes serve as a valuable tool for revealing instances of academic HINs, thereby reaffirming the interconnectedness of these academic subjects within ACM.

6 Further Impact on Scalability

Scalability is crucial for real-world network analysis. In the case of citation networks, for example, answering the questions of relevance for business disciplines frequently requires the ability to process practically all publications from a given domain, thus millions of author, paper, and institution nodes. So far, very limited progress has been made towards facilitating hypergraph learning at scale [1]. With approaches ranging from straightforward creation of mini-batches for large-scale training [30] and graph partitioning [3] as part of preprocessing to the approaches. Through the incorporation of learnable prototypes into our methodologies, we acquire a versatile instrument for designing hypergraph sampling to enhance scalability. This enables us to construct well-balanced mini-batches by sampling data from neighboring prototypes, or clusters by partitioning nodes based on prototypes. Hence, this can ensure diversity and the inclusion of information-rich training data.

7 Conclusion

We have introduced prototype-enhanced hypergraph learning, a novel framework for modeling heterogeneous information networks without the need to build metapaths. Our research demonstrates that it is possible to analyze heterogeneous information networks effectively without relying on metapaths, and enabling the capture of high-order information within such networks. We also demonstrated the effectiveness of our method, by numerical and qualitative experiments on several representative heterogeneous datasets, showcasing its capabilities in capturing the rich relationships present in complex networks.

References

[1] A. Antelmi, G. Cordasco, M. Polato, V. Scarano, C. Spagnuolo, and D. Yang. A survey on hypergraph representation learning. *ACM Computing Survey*, 2023.

[2] D. Arya, D. K. Gupta, S. Rudinac, and M. Worring. Adaptive neural message passing for inductive learning on hypergraphs. *arXiv preprint arXiv:2109.10683*, 2021.

[3] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019.

[4] K. Ding, J. Wang, J. Li, D. Li, and H. Liu. Be more with less: Hypergraph attention networks for inductive text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020.

[5] Y. Dong, N. V. Chawla, and A. Swami. Metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

[6] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, 2019.

[7] X. Fu, J. Zhang, Z. Meng, and I. King. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference 2020*, 2020.

[8] Y. Gao, Y. Feng, S. Ji, and R. Ji. Hgnn+: General hypergraph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[9] Y. Gao, Z. Zhang, H. Lin, X. Zhao, S. Du, and C. Zou. Hypergraph learning: Methods and practices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[10] Z. Hu, Y. Dong, K. Wang, and Y. Sun. Heterogeneous graph transformer. In *Proceedings of The Web Conference*, 2020.

[11] J. Huang and J. Yang. Unignn: a unified framework for graph and hypergraph neural networks. *arXiv preprint arXiv:2105.00956*, 2021.

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

[13] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[14] M. Li, Y. Zhang, X. Li, Y. Zhang, and B. Yin. Hypergraph transformer neural networks. *ACM Trans. Knowl. Discov. Data*, apr 2023.

[15] J. Liu, L. Song, G. Wang, and X. Shang. Meta-hgt: Metapath-aware hypergraph transformer for heterogeneous information network embedding. *Neural Networks*, 2023.

[16] Q. Lv, M. Ding, Q. Liu, Y. Chen, W. Feng, S. He, C. Zhou, J. Jiang, Y. Dong, and J. Tang. Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021.

[17] Q. Lv, M. Ding, Q. Liu, Y. Chen, W. Feng, S. He, C. Zhou, J. Jiang, Y. Dong, and J. Tang. Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021.

[18] L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[19] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference*, 2018.

[20] C. Shi, Y. Li, J. Zhang, Y. Sun, and P. S. Yu. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering*, 2017.

[21] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[22] X. Wang, D. Bo, C. Shi, S. Fan, Y. Ye, and P. S. Yu. A survey on heterogeneous graph embedding: Methods, techniques, applications and sources. *IEEE Transactions on Big Data*, 2023.

[23] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. Heterogeneous graph attention network. In *The World Wide Web Conference*, 2019.

[24] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32, 2021.

[25] Y. Xie, Z. Xu, J. Zhang, Z. Wang, and S. Ji. Self-supervised learning of graph neural networks: A unified review. *IEEE transactions on pattern analysis and machine intelligence*, 2022.

[26] N. Yadati, M. Nimishakavi, P. Yadav, V. Nitin, A. Louis, and P. Talukdar. Hypergcn: A new method for training graph convolutional networks on hypergraphs. *Advances in neural information processing systems*, 2019.

[27] C. Yang, Y. Xiao, Y. Zhang, Y. Sun, and J. Han. Heterogeneous network representation learning: A unified framework with survey and benchmark. *IEEE Trans. on Knowl. and Data Eng.*, 2022.

[28] H.-M. Yang, X.-Y. Zhang, F. Yin, and C.-L. Liu. Robust classification with convolutional prototype learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[29] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim. Graph transformer networks. In *Advances in Neural Information Processing Systems*, 2019.

[30] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

[31] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[32] H. Zhang, X. Liu, and J. Zhang. HEGEL: Hypergraph transformer for long document summarization. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022.

[33] D. Zhou, J. Huang, and B. Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in neural information processing systems*, 2006.

[34] S. Zhu, C. Zhou, S. Pan, X. Zhu, and B. Wang. Relation structure-aware heterogeneous graph neural network. In *IEEE International Conference on Data Mining*, 2019.