Graph Neural Networks for Soft Semi-Supervised Learning on Hypergraphs

Naganand Yadati[1], Tingran Gao[2], Shahab Asoodeh[3], Partha Talukdar[1], and Anand Louis[1]

[1] Indian Institute of Science, Bangalore, Karnataka 560012, India
[2] University of Chicago, IL 60637, United States,
tingrangao@galton.uchicago.edu
[3] Harvard University, Cambridge, MA 02138, United States,
shahab@seas.harvard.edu
{naganand,ppt,anandl}@iisc.ac.in

Abstract. Graph-based semi-supervised learning (SSL) assigns labels to initially unlabelled vertices in a graph. Graph neural networks (GNNs), esp. graph convolutional networks (GCNs), are at the core of the current-state-of-the art models for graph-based SSL problems. GCNs have recently been extended to undirected hypergraphs in which relationships go beyond pairwise associations. There is a need to extend GCNs to directed hypergraphs which represent more expressively many real-world data sets such as co-authorship networks and recommendation networks. Furthermore, labels of interest in these applications are most naturally represented by probability distributions. Motivated by these needs, in this paper, we propose a novel GNN-based method for directed hypergraphs, called Directed Hypergraph Network (DHN) for semi-supervised learning of probability distributions (Soft SSL). A key contribution of this paper is to establish generalisation error bounds for GNN-based soft SSL. In fact, our theoretical analysis is quite general and has straightforward applicability to DHN as well as to existing hypergraph methods. We demonstrate the effectiveness of our method through detailed experimentation on real-world datasets. We have made the code available.

1 Introduction

In the last decade, deep learning models have been successfully embraced in many different fields and have been shown to achieve excellent performance on a vast range of applications. Graph Convolutional Networks (GCNs) [16] have been recently proposed as an adaptation of a particular deep learning model (i.e., convolutional neural networks) to enable handling of graph-structured data. GCN has been shown to be effective especially in semi-supervised learning on attributed graphs. GCNs have inspired the current state-of-the art models for graph-based SSL [28,24].

While graphs are powerful data representations for pairwise relationships, hypergraphs provide more flexible data representations for relationships beyond pairwise associations. A hypergraph relaxes the notion of an edge (commonly called hyperedge) to contain more than two vertices. Real-world datasets such as co-authorship networks, recommendation networks, email communication networks, protein-protein interaction networks, etc. can be flexibly modelled by hypergraphs. For example, in a co-authoship network, a document (hyperedge) can be be co-authored by more than two authors (vertices). The existence of such relationships naturally motivates the problem of hypergraph-based semi-supervised learning (SSL).

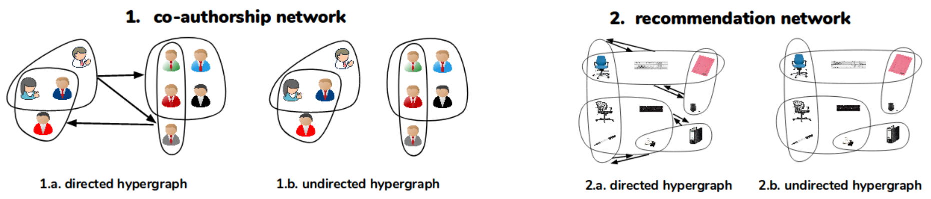
1. co-authorship network
1.a. directed hypergraph
1.b. undirected hypergraph
2. recommendation network
2.a. directed hypergraph
2.b. undirected hypergraph

Fig. 1. (Best seen in colour) Examples of real-world networks modelled as directed hypergraphs and undirected hypergraphs. To the left is 1. co-authorship network in which vertices are authors, and hyperedges are collaborations (documents). 1.a. shows the network modelled as a directed hypergraph in which directions are citations among documents. 1.b. shows the undirected version in which the citation relationships are absent. To the right is 2. recommendation network in which vertices are products, and each hyperedge contains all products bought by a user. 2.a. shows the directed hypergraph in which directions represent user similarity (two-way) and 2.b. shows the undirected version. We are interested in semi-supervised vertex classification of probability distributions in these networks. The vertex labels in the examples are research topic interests for co-authorship and product ratings for recommendation networks.

There exist GNN-based methods for semi-supervised vertex classification on undirected hypergraphs [7,29]. However, these methods do not work for directed hypergraphs. Directed hypergraphs encode additional relationships as illustrated in Figure 1. For example, in a co-authorship network, documents (hyperedges) are related by directed citation relationships. Motivated by this, our focus in this paper is on semi-supervised vertex classification on the more powerful directed hypergraphs which encode an additional layer of relationships (Figure 1).

Furthermore, vertex labels in these applications involving directed hypergraphs e.g. research interests of authors in co-authorship networks, product ratings of products in recommendation networks, etc. are most naturally represented by probability distributions (soft labels). Following up on a prior work [20] that generalised label propagation to graph-based SSL of probability distributions (graph-based *soft* SSL) and motivated by the fact that directed hypergraphs and soft-labels occur *simultaneously* in real-world, we make the following contributions.

– We explore GNNs for (hyper)graph-based soft SSL. We propose DHN (Directed Hypergraph Network), a novel GNN-based method for directed hypergraphs. DHN can be applied for soft SSL using existing tools from optimal transportation.

- Our second contribution is to provide generalisation error bounds for GNN-based soft SSL. Our effort in this direction has lead to deriving generalisation error bounds for GNNs within the framework of algorithmic stability. We establish that such models, which use filters with bounded eigenvalues independent of graph size, can satisfy the strong notion of uniform stability and thus are generalisable. In particular, the algorithmic stability of GNNs depends on the largest absolute eigenvalue of the graph convolution filter . Our analysis is quite general and the error bounds can be easily established for DHN and existing hypergraph neural methods.
- We demonstrate DHN's effectiveness through detailed experimentation on real-world data. In particular, we demonstrate superiority over state-of-the-art hypergraph-based neural networks such as HGNN [7] and HyperGCN [29]. We provide new empirical benchmarks for soft-SSL. We have made the code available to foster reproducible research.

We have made the code and supplementary pdf available at this link

2 Related work

Geometric deep learning is an umbrella phrase for emerging techniques attempting to generalise (structured) deep neural networks to non-Euclidean domains such as graphs and manifolds. GCN [16] and their various extensions are the current state-of-the art for graph-based SSL [28] and graph-based unsupervised learning [12] problems. The reader is referred to recent books [13,18] on this topic. Recently, graph-based deep models (also message-passing neural networks [11]) have been analysed theoretically [25].

Learning on hypergraphs: Hypergraph is a combinatorial structure consisting of vertices and hyperedges, where each hyperedge is allowed to connect any number of vertices, thus generalizing graphs. This additional flexibility facilitates the capture of higher order interactions among objects; applications have been found in many fields such as computer vision, network clustering , folksonomies, cellular networks, and community detection.

The seminal work on hypergraphs [32] introduced the popular clique expansion [7] of a hypergraph. Hypergraph neural networks (HGNN) [7] use the clique expansion while HyperGCN [29] uses the mediator-based Laplacian to extend GCNs to hypergraphs. Another line of work uses the mathematically appealing tensor methods but they are limited to uniform hypergraphs. Recent developments work for arbitrary hypergraphs and fully exploit the hypergraph structure [15,30,2].

Graph-based soft SSL: Researchers have shown that using unlabelled data during training can improve label prediction significantly [22]. While most methods assume that labels of interest are numerical or categorical variables, other works "soften" this assumption and handle "soft labels" such as histograms [4,23]. One way of propagating histograms is to minimise the Kullback-Leibler (KL) divergence [21]. Recent studies have replaced the metric-agnostic KL divergence with metric-aware Wasserstein distance (interactions between histogram bins) for graphs [20] and hypergraphs [10].

Embeddings in Wasserstein space: There exist at least a couple of recent works that embed Gaussian distributions in the Wasserstein space [19]. Inspired by a recent work [8], in this work, we focus on embedding input data as a discrete probability distrirbution on a fixed support set. The Wasserstein distance and its gradient require the solution of a linear program [27] and are costly to compute. A popular efficient approximation is the Sinkhorn divergence [5] in which the underlying problem is regularised and is computed efficiently by a fixed-point iteration.

In all the papers that we have discussed above, the proposed methods are either restricted to graphs or undirected hypergraphs and do not work for directed hypergraphs. Also, none of the GNN-based methods discusses soft SSL. Our contributions are precisely to address these limitations.

3 Method

In this section, we first describe soft SSL on directed hypergraphs and then propose DHN (Directed Hypergraph Network) for the problem.

3.1 Directed hypergraph

A directed hypergraph [9] is an ordered pair $\mathcal{H} = (V, E_d)$ where $V = \{v_1, \cdots, v_n\}$ is a set of n vertices and $E_d = \{(t_1, h_1), \cdots, (t_m, h_m)\} \subseteq 2^V \times 2^V$ is a set of m directed hyperedges. Each element in E_d is an ordered pair (t, h) where $t \subseteq V$ is the *tail* and $h \subseteq V$ is the *head* with $t \neq \emptyset$, $h \neq \emptyset$. Denote the set of all undirected hyperedges by E i.e., $E = \bigcup_{(t,h) \in E_d} \Big(t \cup h \Big)$. Denote $I \in \{0,1\}^{|V| \times |E|}$ to be the incidence matrix of E i.e. $I(v, e) = 1$ if $v \in e$ and 0 otherwise.

3.2 Soft SSL on directed hypergraphs

We consider the problem of predicting probability distributions for the vertices in $\mathcal{H} = (V, E_d)$ given a typically small subset $V_k \subseteq V$ of vertices with known distributions. In this work, we are concerned with discrete distributions modelled on a metric space i.e. an ordered pair (M, C) in which M is a set and $C : M \times M \to \mathbb{R}$ is the cost function (metric) associated with the set. Furthermore, we assume that we are provided with a feature matrix, $X_V \in \mathbb{R}^{n \times D_V}$, in which each vertex $v \in V$ is represented by a D_V-dimensional feature vector x_v (here $n = |V|$). We are also provided with a hyperedge feature matrix $X_E \in \mathbb{R}^{m \times D_E}$ with $x_e, e \in E$ as D_E-dimensional feature representations (here $m = |E_d|$).

Our objective is to learn a labelling function $Z = \phi\big(\mathcal{H}, X_V, X_E\big)$ that maps each vertex to a probability distribution in the space of discrete probability distributions $\mathcal{P}_F(M)$ on F atoms (F is number of histogram bins) defined on the metric space (M, C). The cost function C can be represented by a non-negative symmetric matrix of size $F \times F$. Note that each row of $Z \in [0,1]^{n \times F}$ maps each vertex $v \in V$ to a probability distribution $Z_v \in [0,1]^F$. The function h is going to be trained on a supervised loss, L ,w.r.t to the vertices in V_k so that the trained h can be used to predict distributions of all the vertices in $V \setminus V_k$. We now give an example application and then the details of the labelling function h followed by the supervised loss L.

Example application: Predicting topic distributions of authors in co-authorship networks can be posed as a soft SSL problem on directed hypergraphs. V represents the set of authors, E the set of all collaborations (documents), E_d the citation relationships among the documents, F the number of possible research interests of authors (Machine Learning, Theoretical Computer Science, etc.), X_V and X_E any available features on the authors and documents respectively (e.g. text attributes).

3.3 Directed Hypergraph Network (DHN)

Hypergraphs contain hyperedges in which relationships can go beyond pairwise and hence are challenging to deal with. A flexible way to embed vertices of a hypergraph is to "approximate" the hypergraph by a suitable graph and then apply traditional graph-based methods on the vertices. Two notable candidates of ϕ are Hypergraph neural network (HGNN) [7] and Hypergraph Convolutional Network (HyperGCN) [29]. HGNN uses the clique expansion of the hypergraph [32] while HyperGCN uses the mediator-based Laplacian [2] to approximate the input hypergraph. However, they are restricted to undirected hyperedges and also cannot exploit the hyperedge feature matrix X_E.

A key idea of our approach is to treat each hyperedge $e \in E$ as a vertex of the graph $\mathcal{G} = (E, E_d)$. We then pass $\mathcal{G}$ through a graph neural network to obtain $H_E = f_{GNN}(\mathcal{G}, X_E)$ so that the initial features, X_E, are refined to H_E. We then propose the layer-wise propagation rule of DHN as:

$$H_V^{(t+1)} = \sigma\left(\left[H_V^{(t)}, \quad I \cdot H_E^{(t)} \cdot \Theta^{(t)}\right]\right), \quad t = 0, \cdots, \tau - 1 \tag{1}$$

where $[\cdot,\cdot]$ denotes concatenation, t is the time step, I is the incidence matrix, $H_E^{(t+1)} = \sigma_1\Big(I^T H_V^{(t)}\Big)$ for $t = 1, \cdots, \tau - 1$, $H_E^{(0)} = f_{GNN}(\mathcal{G}, X_E)$, σ and σ_1 are non-linear activation functions, and τ is the total number of propagation steps with $H_V^{(0)} = X_V$. Note that the labelling function $Z = \phi(\mathcal{H}, X_V, X_E) = \text{softmax}\Big(H_V^{\tau}\Big)$ where softmax is applied row-wise.

3.4 The supervised loss L

A crucial observation here is that because of the softmax layer, the output of h is (already) inherently a probability distribution. For each vertex $v \in V_k$, the predicted distribution Z_v and the (known) true distribution Y_v must be “close” to each other. A natural way to compare probability distributions is to use the KL-divergence between Y_v and Z_v. However, KL-divergence cannot exploit the metric space (M, C) and suffers from stability issues [3]. In this work, we use the more stable Wasserstein distance to exploit the metric space [10].

$$L = \sum_{v \in V_k} W_p\Big(Z_v, Y_v\Big), \quad W_p(\mu, \nu) = \left(\inf_{\pi \in \Pi(\mu,\nu)} \int_{M \times M} C(x_1, x_2)^p d\pi(x_1, x_2)\right)^{\frac{1}{p}}.$$

For discrete distributions, W_p is the solution of a linear program. For practical purposes, we compute the regularised distance using the Sinkhorn algorithm. Please see the supplementary material for more details.

Optimisation: We call DHN optimised with the Wasserstein loss as Soft-DHN. All parameters are learned using stochastic gradient descent (SGD). Please see the supplementary material for time complexity.

4 Theoretical analysis: Generalisation error

A key contribution of this chapter is to provide generalisation error bounds for (hyper)graph-based soft-SSL. Our effort to derive these bounds has lead us to a more powerful outcome, namely, proving generalisation error bounds for a one-layer GNN by extending the results of a traditional GCN [25] to the soft SSL setting with Wasserstein loss. The main novelty is to generalise the error bounds to the learning problem “valued in the Wasserstein space.” The main challenge is that the Wasserstein space is an abstract metric space without linear structure.

The section is organised as follows. We first introduce all the notations needed (ego-graph view, semi-supervised learning setting, etc.). We then give single layer and SGD bounds using the notations. We finally give the main result (proposition 1) which states that a GNN trained with Wasserstein loss has the same generalisation error bound as the traditional GCN (trained with cross entropy).

Let $G = (V, E)$ be a graph with $|V| = n$. We consider a one-layer GNN

$$f(X, \Theta) = \sigma(KX\Theta) \tag{2}$$

where $X \in \mathbb{R}^{n \times d}$ is the feature matrix (n is the number of vertices in a graph, d is the dimension of the feature vectors), $K = g(L_G)$ is a graph filter (typically symmetrically normalised adjacency with self loops, and $L_G \in \mathbb{R}^{n \times n}$ is the graph Laplacian), and $\Theta \in \mathbb{R}^{d \times F}$ is the set of parameters. We note that our proposed DHN falls under this formulation in special circumstances. Specifically if the non-linearity σ_1 in Equation 1 is removed we get the kernel $K = II^T$ (also known as the clique expansion of the hypergraph [32].) The non-linearity σ in Equation 2 is the softmax function acting on each row of the product $KX\Theta \in \mathbb{R}^{n \times F}$; the output is of dimension $n \times F$, where each output row is a discrete probability distribution, i.e., $f(X,\Theta) \geq 0$ and $f(X,\Theta)\mathbf{1}_F = \mathbf{1}_n$ where $\mathbf{1}_F = (1,\ldots,1)^F \in \mathbb{R}^F$, and similarly for $\mathbf{1}_n$. Without loss of generality, we assume $d = 1$. Note that in order for the output to be nontrivial probability distributions, we must assume $F > 1$.

We adopt an ego-graph view [25] to simplify our discussion for local behavior of the soft GCN at a particular vertex. Whenever no confusion arises, we identify a vertices x and χ in the graph G with their respective D-dimensional feature vectors. Thus the output of f at $x \in V$ is $f(x,\Theta) = \sigma\left(\sum_{\chi\in\mathcal{N}(x)} K_{x\chi}\chi\Theta\right) = \sigma\left(\left(\sum_{\chi\in\mathcal{N}(x)} K_{x\chi}\chi\right)\cdot\Theta\right)$ where $\mathcal{N}(x)$ denotes for the one-hop neighborhood of x with respect to the adjacency relation defined by matrix K, and $K_{x\chi} \in \mathbb{R}$ stands for the entry in $K \in \mathbb{R}^{n\times n}$ that describes the adjacency relation between vertices x and χ. Let $E_x := \sum_{\chi\in\mathcal{N}(x)} K_{x\chi}\chi \in \mathbb{R}$ so that $f(x,\Theta) = \sigma(E_x\cdot\Theta)$.

We consider the supervised learning setting, and learn GNN from the training set $\{z_i = (x_i, y_i), i = 1,\ldots,m\}$ sampled i.i.d. from the product space $V \times \mathcal{P}_F$ with respect to probability distribution $\mathcal{D}$ on this product space, where $\mathcal{P}_F$ is the space of discrete probability distributions on F atoms. The output of softmax lies in $\mathcal{P}_F$, which is a convex cone. For any new data $z = (x,y) \sim \mathcal{D}$, we evaluate the performance of GNN f using a Wasserstein cost $\ell(f(\cdot,\Theta), z) = \ell(f(\cdot,\Theta),(x,y)) = W(f(x,\Theta), y)$.

Here the Wasserstein cost is defined with respect to a cost function penalizing moving masses across bins. Since we are working only with histograms in GNN, we shall use a cost function $C \in \mathbb{R}^{F\times F}$ that is defined for pairs of histogram bins. The transport problem is a linear program with $z = f(x,\Theta)$.

4.1 Assumptions/Notations

To avoid unnecessary technical complications, assume the histogram admits a geometric realisation over the one-dimensional Euclidean space, such that the ith bin is placed at location $b_i \in \mathbb{R}$, and set $C_{ij} := |b_i - b_j|, \quad \forall 1 \leq i,j \leq F$. Without loss of generality, we assume $b_1 \leq b_2 \leq \cdots \leq b_F$, and write $h_i := b_{i+1} - b_i \geq 0$ for all $i = 1,\ldots,F-1$. Denote the diameter of the support by $D := \max_{1\leq i,j\leq F} |b_i - b_j| = b_F - b_1$. We take the Wasserstein cost as the Wasserstein-1 distance: $W(\mu,\nu) := W_1(\mu,\nu)$. In this particular one-dimensional setting, we have a particularly simple form for the cost function:

$$W_1(\mu,\nu) = \int_0^1 \left|F_\mu^{-1}(s) - F_\nu^{-1}(s)\right| \mathrm{d}s = \int_{-\infty}^{\infty} |F_\mu(t) - F_\nu(t)|\, \mathrm{d}t \tag{3}$$

where $F_\mu : \mathbb{R} \to [0,1]$, $F_\nu : \mathbb{R} \to [0,1]$ are the cumulative distribution functions of μ, ν, respectively; F_μ^{-1}, F_ν^{-1} are the *generalized inverses* of F_μ and F_ν, respectively, defined as (similar for F_ν^{-1})

$$F_\mu^{-1}(t) := \inf\{b \in \mathbb{R} : F_\mu(b) > t\}, \qquad \forall t \in [0,1]. \tag{4}$$

This characterisation is seen in any standard literature on optimal transport, e.g., [26].

4.2 Definitions: Generalisation and Empirical Errros

Let the learning algorithm A_S on a dataset S be a function from ζ^m to $(\mathcal{Y})^X$. where $\mathcal{X}$ is the input Hilber space, $\mathcal{Y}$ is the output Hilbert space, $\zeta = \mathcal{X} \times \mathcal{Y}$. The training set of datapoints, labels is $S = \{z_1 = (x_1, y_1), \cdots, z_k = (x_k, y_k))\}$. Let the loss function be $\ell : \zeta^m \times \zeta \rightarrow \mathbb{R}$. Then the generalisation error or risk $R(A_S)$ is deined as $R(A_S) := \mathbb{E}\Big[\ell(A_S, z)\Big] = \int \ell(A_S, z)p(z)dz$ where $p(z)$ is the probability of seeing the sample $z \in S$. The empirical error, on the other hand, is $R_{\text{emp}}(A_S) := \frac{1}{k}\sum_{j=1}^{k} \ell(A_S, z_j)$

4.3 Extension to GNNs on hypergraphs

We note that our proposed DHN falls under this formulation in special circumstances. In particular, if the non-linearities in Equation 1 are removed, our DHN can be seen as $K = IA^2$ where $f_{GNN}(\mathcal{G}, X_E)$ is the simple graph convolution operator [28] and A is the symmetrically normalised adjacency (with self loops) of the graph $\mathcal{G}$. Also, the analysis can be extended to exisiting hypergraph GNNs such as hypergraph neural network (HGNN [7]) where $K = II^T$ (also known as the clique expansion [32] of the hypergraph). Hence, our theoretical analysis is quite general that can be easily appplied to DHN and existing hypergraph neural methods. The full proof is given in the supplementary material.

Theorem 1. *Let A_S be a one-layer GNN algorithm (of Equation 2) equipped with the graph convolutional filter $g(L_G)$ and trained on a dataset S for T iterations. Let the loss and activation functions be Lipschitz-continuous and smooth. Then the following expected generalisation gap holds with probability at least $1-\delta$, $\delta \in \{0, 1\}$:*

$$\mathbb{E}_{\text{SGD}}\Big[R(A_S) - R_{\text{emp}}(A_S)\Big] \leq \frac{1}{m}O\Big((\lambda_G^{max})^{2T}\Big) + \Big(O\Big((\lambda_G^{max})^{2T}\Big) + B\Big)\sqrt{\frac{\log\frac{1}{\delta}}{2m}} \quad (5)$$

where the expectation $\mathbb{E}_{SGD}$ is taken over the randomness inherent in SGD, m is the no. training samples, and B is a constant which depends on the loss function. Our theorem states that GNN trained with the Wasserstein loss enjoys the same generalisation error bound as the traditional GCN (trained with cross entropy). We establish that such models, which use filters with bounded eigenvalues independent of graph size, can satisfy the strong notion of uniform stability and thus is generalisable.

5 Experiments

We conducted experiments on 5 real-world directed hypergraphs (four are co-authorship datasets and one is a recommendation dataset). Statistics of the datasets are in the supplementary. We labelled 1% of the nodes in each dataset. Vertex labels in all our datasets are discrete distributions seen in the real world (not semi-synthetic). For example, in a co-authorship dataset, an author that has

Table 1. Results on real-world directed hypergraphs. We report 100× mean squared errors (lower is better) over 10 different train-test splits. All reported numbers are to be multiplied by 0.01 to get the actual numbers. Please see section 5 for details.

Method	**Cora**	**DBLP**	**ACM**	**Amazon**	**arXiv**
KL-MLP	8.94 ± 0.16	7.72 ± 0.14	8.47 ± 0.15	6.81 ± 0.16	10.87 ± 0.25
OT-MLP	7.45 ± 0.35	7.53 ± 0.18	7.85 ± 0.26	6.78 ± 0.24	10.01 ± 0.23
KLR-MLP	8.05 ± 0.22	7.35 ± 0.18	7.82 ± 0.29	6.74 ± 0.15	–
OTR-MLP	6.57 ± 0.43	7.24 ± 0.18	6.77 ± 0.32	6.72 ± 0.23	–
KL-HGNN	7.86 ± 0.25	7.17 ± 0.12	7.23 ± 0.19	6.71 ± 0.19	9.95 ± 0.25
KL-HyperGCN	7.95 ± 0.27	7.15 ± 0.17	7.53 ± 0.21	6.69 ± 0.17	9.99 ± 0.23
Soft-HGNN	5.97 ± 0.37	6.18 ± 0.37	6.02 ± 0.37	6.63 ± 0.39	8.61 ± 0.49
Soft-HyperGCN	6.02 ± 0.32	6.21 ± 0.35	6.04 ± 0.32	$\mathbf{6.61 \pm 0.30}$	8.60 ± 0.47
Soft-HGAT	5.83 ± 0.39	6.05 ± 0.33	5.94 ± 0.39	6.62 ± 0.45	8.47 ± 0.46
Soft-SAGNN	5.69 ± 0.32	6.07 ± 0.47	5.82 ± 0.41	$\mathbf{6.59 \pm 0.32}$	8.29 ± 0.27
Soft-HNHN	5.64 ± 0.37	6.11 ± 0.39	5.88 ± 0.27	6.64 ± 0.36	8.34 ± 0.35
Soft-DHGCN	5.62 ± 0.35	6.06 ± 0.45	5.84 ± 0.39	6.67 ± 0.33	8.31 ± 0.29
KL-DHN (ours)	7.04 ± 0.24	6.97 ± 0.22	7.16 ± 0.24	6.65 ± 0.17	9.34 ± 0.32
Soft-DHN (ours)	$\mathbf{4.87 \pm 0.40}$	$\mathbf{5.65 \pm 0.42}$	$\mathbf{5.12 \pm 0.34}$	$\mathbf{6.55 \pm 0.33}$	$\mathbf{7.69 \pm 0.36}$

published 7 papers in vision conferences, and 3 in NLP conferences, is assigned the soft label [0.7, 0.3] (this is neither multi-label nor single label). In fact, more than 80% vertices in all our datasets have such proper soft labels. For more details on dataset construction, please see Section 6 of the supplementary material. Our focus is on predicting true probability distributions. Existing hypergraph neural methods such as HGNN [7], and HyperGCN [29] were originally designed for multi-class SSL (Hard SSL). However, we adapted them for soft SSL by training them with KL and Wasserstein losses.

5.1 Experimental setup

We take extensive measures to ensure fairness of comparison with baselines. Inspired by the experimental setups of prior related works [16,17], we tune hyperparameters using the Cora citation network dataset alone and use the optimal hyperparameters for all the other datasets. We hyperparameterise the cost matrix (base metric of the Wasserstein distance) as $C_{ii} = 1, i = 1, \cdots, F$ and $C_{ij} = \eta, i \neq j$. The cost matrix C is an $F \times F$ matrix (F is the number of histogram bins) with ones on the diagonal and a hyperparameter η elsewhere. We could have used a matrix of all ηs. But it is no different from a matrix of all ones from the optimisation perspective and so we used the above more general matrix. Details of hyperparameter tuning and optimal hyperparameters of all methods (including baselines) are in the suplementary material. We use the mean squared error (MSE) between true and predicted distributions on the test set of vertices. Table 1 shows MSEs on the test split for all the five datasets.

5.2 Baselines

We used both Wasserstein distance and KL divergence to train different models. As already noted, we used the Sinkhorn algorithm to compute the (regularised) Wasserstein distance. We compared DHN with the following baselines:

- **KL-MLP**: We used a simple multi-layer perceptron (MLP) on the features of the vertices and trained it using KL-divergence
- **OT-MLP**: We trained another MLP with the Wasserstein distance as the loss function. Note that this baseline and the previous baseline do not use the structure (graph / hypergraph)
- **KLR-MLP**: We regularised an MLP with explicit KL-divergence-based regularisation that uses the structure (graph / hypergraph) [21].
- **OTR-MLP**: We regularised an MLP with explicit Wasserestein-distance-based regularisation that uses the structure (graph / hypergraph) [20]. For hypergraphs we used the clique expansion of the hypergraph [10].
- **KL-HGNN / KL-HyperGCN**: We trained the different GCN-based methods on hypergraphs with KL divergence loss function on the labelled vertices.
- **Soft-HGNN / Soft-HyperGCN**: We trained the different GCN-based methods on hypergraphs with the Wasserstein distance as the loss function.
- **Soft-Hyper-Atten**: This model is a generalisation of graph attention to hypergraphs [1] We trained it with the Wasserstein distance
- **Soft-Hyper-SAGNN**: We trained the recently proposed self-attention hypergraph-based method [31] with the Wasserstein distance loss.
- **Soft-HNHN**: We also used this very recent model [6] as a baseline (with Wasserstein loss). HNHN uses hypereges as neurons and computes hyperedge representations.
- **Soft-DHGCN**: This baselines [14] uses separate incidence matrices for tail and head. We trained it with Wasserstein loss.

5.3 Discussion

We used a simple one-layer architecture for our proposed DHN and a 2-hop simplified GCN [28] as the GNN model on the graph $\mathcal{G} = (E, E_d)$ i.e.

$$Z = softmax(I \cdot H_E \cdot \Theta_1), \quad H_E = A^2 X_E \Theta_2$$

where A is the symmetically normalised adjacency (with self loops) of the graph $\mathcal{G}$. We demonstrate that this simple model is effective enough through an ablation study in Section 5.1 of the supplementary. Please see Section 5 of the supplementary for more experiments on arXiv. Our results on real-world datasets demonstrate strong performances across all the datasets esp. on the co-authorships networks. Specifically, we observe that Soft models (that use the Wasserstein loss) are almost always superior to their counterparts that use the KL divergence as the loss function. This is because the Soft models can exploit the distance matrix C while KL-divergence does not. Moreover, our proposed DHN outperforms several hypergraph baselines. This is because they do not exploit the rich structural information in the directed hyperedges (connections among hyperedges) while our DHN does exploit them. Though baselines such as HNHN [6] compute representations for hyperedges, they do not exploit dependencies between them. The DHGCN baseline [14] does exploit such relationships. However, it does not treat the relationships between hyperedges as separate edges. The benefits of doing so of a separate graph are evident in the table where we report 2 hops of a GNN run on this graph. We also experimented on standard graph node-classification datasets such as Cora, Citeseer, and Pubmed by treating the class label as one-hot distribution. We used the Soft variants of GCN [16], Simple GCN [28], and GAT [24]. We achieved competitive results as in Section 5 (Table 3) of the supplementary.

6 Conclusion

We have proposed DHN, a novel method for soft SSL on directed hypergraphs. DHN can effectively propagate histograms to unknown vertices by integrating vertex features, directed hyperedges and undirected hypergraph structure. We have established generalisation bounds for DHN within the framework of algorithmic stability. Specifically we modified the "gradient" in Wasserstein space to satisfy Lipschitz condition required in the stability framework. DHN is effective compared to SOTA baselines.

References

1. Bai, S., Zhang, F., Torr, P.H.S.: Hypergraph convolution and hypergraph attention. Pattern Recognition **110**, 107637 (2021)
2. Chan, T.H., Liang, Z.: Generalizing the hypergraph laplacian via a diffusion process with mediators. In: COCOON (2018)
3. Chen, Y., Ye, J., Li, J.: A distance for hmms based on aggregated wasserstein metric and state registration. In: ECCV. pp. 451–466 (2016)
4. Corduneanu, A., Jaakkola, T.S.: Distributed information regularization on graphs. In: NIPS, pp. 297–304. MIT Press (2005)
5. Cuturi, M.: Sinkhorn distances: Lightspeed computation of optimal transport. In: NIPS. Curran Associates, Inc. (2013)
6. Dong, Y., Sawin, W., Bengio, Y.: HNHN: hypergraph networks with hyperedge neurons. CoRR **abs/2006.12278** (2020)
7. Feng, Y., You, H., Zhang, Z., Ji, R., Gao, Y.: Hypergraph neural networks. In: AAAI (2019)
8. Frogner, C., Mirzazadeh, F., Solomon, J.: Learning entropic wasserstein embeddings. In: ICLR (2019)

9. Gallo, G., Longo, G., Pallottino, S., Nguyen, S.: Directed hypergraphs and applications. Discrete Appl. Math. (1993)
10. Gao, T., Asoodeh, S., Huang, Y., Evans, J.: Wasserstein soft label propagation on hypergraphs: Algorithm and generalization error bounds. In: AAAI (2019)
11. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: ICML (2017)
12. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NIPS. Curran Associates, Inc. (2017)
13. Hamilton, W.L.: Graph representation learning. Synthesis Lectures on Artificial Intelligence and Machine Learning **14**(3), 1–159 (2020)
14. Han, J., Cheng, B., Wang, X.: Two-phase hypergraph based reasoning with dynamic relations for multi-hop kbqa. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI). pp. 3615–3621 (2020)
15. Hein, M., Setzer, S., Jost, L., Rangapuram, S.S.: The total variation on hypergraphs - learning on hypergraphs revisited. In: NIPS. Curran Associates, Inc. (2013)
16. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
17. Liao, R., Zhao, Z., Urtasun, R., Zemel, R.S.: Lanczosnet: Multi-scale deep graph convolutional networks. In: ICLR (2019)
18. Ma, Y., Tang, J.: Deep Learning on Graphs. Cambridge University Press (2020)
19. Muzellec, B., Cuturi, M.: Generalizing point embeddings using the wasserstein space of elliptical distributions. In: NeurIPS. Curran Associates, Inc. (2018)
20. Solomon, J., Rustamov, R., Guibas, L., Butscher, A.: Wasserstein propagation for semi-supervised learning. In: ICML (2014)
21. Subramanya, A., Bilmes, J.: Semi-supervised learning with measure propagation. J. Mach. Learn. Res. **12**, 3311–3370 (2011)
22. Subramanya, A., Talukdar, P.P.: Graph-Based Semi-Supervised Learning. Morgan & Claypool Publishers (2014)
23. Tsuda, K.: Propagating distributions on a hypergraph by dual information regularization. In: ICML (2005)
24. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: ICLR (2018)
25. Verma, S., Zhang, Z.L.: Stability and generalization of graph convolutional neural networks. In: KDD (2019)
26. Villani, C.: Topics in optimal transportation theory (2003)
27. Villani, C.: Optimal transport – Old and new, vol. 338. Springer-Verlag (2008)
28. Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., Weinberger, K.: Simplifying graph convolutional networks. In: ICML (2019)
29. Yadati, N., Nimishakavi, M., Yadav, P., Nitin, V., Louis, A., Talukdar, P.: HyperGCN: A new method of training graph convolutional networks on hypergraphs. In: NeurIPS. Curran Associates, Inc. (2019)
30. Zhang, C., Hu, S., Tang, Z.G., Chan, T.H.H.: Re-revisiting learning on hypergraphs: Confidence interval and subgradient method. In: ICML (2017)
31. Zhang, R., Zou, Y., Ma, J.: Hyper-sagnn: a self-attention based graph neural network for hypergraphs. In: International Conference on Learning Representations (ICLR) (2020)
32. Zhou, D., Huang, J., Schölkopf, B.: Learning with hypergraphs: Clustering, classification, and embedding. In: Schölkopf, B., Platt, J.C., Hoffman, T. (eds.) NIPS. MIT Press (2007)