Stability and Generalization of Hypergraph Collaborative Networks

Michael K. Ng[1,2]   Hanrui Wu[3]   Andy Yip[2]

[1] Institute of Data Science, The University of Hong Kong, Hong Kong, China

[2] Department of Mathematics, The University of Hong Kong, Hong Kong, China

[3] College of Information Science and Technology, Jinan University, Guangzhou 510006, China

Abstract: Graph neural networks have been shown to be very effective in utilizing pairwise relationships across samples. Recently, there have been several successful proposals to generalize graph neural networks to hypergraph neural networks to exploit more complex relationships. In particular, the hypergraph collaborative networks yield superior results compared to other hypergraph neural networks for various semi-supervised learning tasks. The collaborative network can provide high quality vertex embeddings and hyperedge embeddings together by formulating them as a joint optimization problem and by using their consistency in reconstructing the given hypergraph. In this paper, we aim to establish the algorithmic stability of the core layer of the collaborative network and provide generalization guarantees. The analysis sheds light on the design of hypergraph filters in collaborative networks, for instance, how the data and hypergraph filters should be scaled to achieve uniform stability of the learning process. Some experimental results on real-world datasets are presented to illustrate the theory.

Keywords: Hypergraphs, vertices, hyperedges, collaborative networks, graph convolutional neural networks (CNNs), stability, generalization guarantees.

1 Introduction

Many real-world applications involve datasets that exhibit graph structures that depict pairwise relationships between vertices. These applications span a wide range of domains, including text analysis[1, 2], social network analysis[3], molecule classification in chemistry[4], point cloud processing[5], mesh generation[6], and knowledge graphs[7], to name a few. In some applications, the only available data are graphs, while in some other applications, features about the vertices and edges are also available. Very naturally, machine learning algorithms that can exploit both graphs and features often yield better results than algorithms that solely rely on graph structures. Of particular interest is the class of graph convolution network (GCN) methods[8–10]. These methods use neural network layers whose output at a vertex depends mostly on others that are deemed relevant according to the graph structure. A very common point of view is that GCN is a generalization of the traditional spatial filters, where graphs can represent neighbors that are not limited to spatial closeness.

While graphs can depict pairwise relationships, hypergraphs can represent relationships between multiple vertices. A hypergraph is a graph in which each edge, or hyperedge, can connect to multiple vertices. Hypergraphs can therefore represent more complex relationships. For example, in traditional paper-authorship networks, two articles (vertices) are connected if they share one or more coauthors. In this way, authorship information, which can provide important clues to the topic of the articles, is lost. Hypergraphs come to the rescue by treating each author as a hyperedge and each article as a vertex. Several works have been devoted to hypergraph learning. In [11], spectral clustering and semi-supervised parametric models on graphs were extended to hypergraphs. In [12], a tensor representation of hypergraphs was proposed to make the optimization of hypergraphs more amenable. Generalizations of graph convolution networks to hypergraph convolution networks have been studied in [13–16]. The main task is to define a convolution operator in a hypergraph such that the transition probability between two vertices can be measured, and the embeddings (or features) of each vertex can be propagated in a hypergraph convolution network. More propagations should be done between vertices connected by a common hyperedge.

Recently, Wu and Ng[17, 18] studied and developed the

---

convolution of vertex and hyperedge features to suitably aggregate values in hypergraphs. It was shown that such hypergraph collaborative networks (HCoN) have obtained superior results in some semi-supervised learning problems compared to other hypergraph neural network models. The proposal is to formulate the learning of vertex and hyperedge embeddings as a joint optimization problem to allow for updating the vertex and hyperedge embeddings simultaneously. The authors showed that the performance can be further boosted by incorporating a hypergraph reconstruction error as one of the objectives.

Apart from the development of machine learning algorithms and applications, efforts on the theory have also been made in the past decades, from general learning theory to specific properties of a class of algorithms. In this paper, we aim to study the algorithmic stability and generalization of the class of HCoN. We show that, in the single-layer case, the HCoN is algorithmically stable. As a result, the generalization gap (difference between training error and testing error) converges to zero as the size of the training set increases. We should explain the notions of training and testing in our context in Section 3.6 below. The analysis sheds light on the design of hypergraph filters in HCoNs, for instance, how the data and filters should be scaled to achieve the uniform stability of the learning process. Our generalization result is also valid for hypergraph convolution networks with the propagation of vertex embeddings only.

The rest of the paper is organized as follows. In Section 2, we review some works related to hypergraph neural networks and generalization guarantee studies. In Section 3, we introduce the HCoN model, some assumptions about the loss function and activation function, and some preliminary results. The results of the algorithmic stability of the HCoN and the generalization guarantee are established in Section 4. In Section 5, we present some experimental studies on the generalization gap to illustrate the theory. Some concluding remarks are given in Section 6.

2 Related work

Generalization guarantees concern the expected difference between training and testing errors. Bousquet and Elisseeff[19] and Mukherjee et al.[20] showed that under suitable regularity assumptions, the stability of an algorithm implies generalization. In addition to the general theory, Bousquet and Elisseeff[19] also studied the stability and generalization of the global minimizer of some regularized learning models. Such results are therefore independent of the particular algorithm used to compute an approximate optimizer. Stochastic gradient descent (SGD) is a popular algorithm to obtain a suboptimal solution to optimization problems. Some classical results on the generalization of SGD in the case of a single pass

of data are reported in [21]. The analysis in the case of multiple passes of SGD is reported in [22].

Stability and generalization analysis of SGD applied to GCNs is studied in [23]. The stability is established in terms of the dominant eigenvalue of the graph filter. A difference between [19] and [23] is that the former considers the model parameters computed via SGD, whereas the latter considers the theoretical optimal model parameters. Our work generalizes the analysis of GCNs in [23] to hypergraph collaborative networks[17, 18] which also include some edge features. A technical difference between our work and [23] is that the latter assumes some Lipschitz conditions on the composition of the loss function and the neural network outputs but we require some Lipschtiz conditions on the loss function only. Ours are therefore easier to verify and more fundamental.

Various kinds of stability have been proposed and studied in the literature[19, 20]. Similar to [23], we use uniform stability which yields tighter bounds than other forms of stability, such as error stability, hypothesis stability and pointwise hypothesis stability[19]. Finally, another related work is [24], which studied regularized graph learning problems and devised some generalization guarantees. However, the resulting generalization gap is inversely proportional to the second smallest eigenvalue of the graph Laplacian matrix. Thus the bound can grow with the size of the graph. The bounds devised by us and [23] are independent of the graph size, and the generalization gap converges to zero with the sample size and the graph size.

3 Hypergraph convolution networks

3.1 Hypergraphs

A hypergraph is a graph where a hyperedge can connect to any number of vertices. In the undirected version, each hyperedge is represented by a subset of vertices. A hypergraph is denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices and $\mathcal{E} = \{e : e \subset \mathcal{V}\}$ is the set of hyperedges. The number of vertices and the number of hyperedges are denoted by $N = |\mathcal{V}|$ and $M = |\mathcal{E}|$, respectively. A hypergraph can also be represented by an incidence matrix $H \in \mathbf{R}^{N \times M}$, whose (i, j)-th equals 1 if the j-th hyperedge is connected to the i-th vertex, and equals 0 otherwise. For simplicity, we consider unweighted graphs, but the weights can be introduced into the hypergraph filters easily as done in [13–16].

Generalizations of graph convolution networks to hypergraph convolution networks have been studied. Given a feature matrix $X_V \in \mathbf{R}^{N \times F_V}$ (where F_V is the dimension of the vertex features) and the dimension O of the output embeddings, the convolution operator in a hypergraph for the propagation of vertex features (or embed-

dings) is constructed as follows:

$$f(X_V|Q_V) = \sigma\left(D_V^{-\frac{1}{2}} H D_E^{-1} H^{\mathrm{T}} D_V^{-\frac{1}{2}} X_V Q_V\right) \quad (1)$$

where $\sigma(\cdot)$ is an activation function, $Q_V \in \mathbf{R}^{F_V \times O}$ is a matrix of learnable parameters, $H \in \mathbf{R}^{N \times M}$ is the incident matrix of the hypergraph, and $D_V = \mathrm{diag}(H\mathbf{1})$ and $D_E = \mathrm{diag}(H^{\mathrm{T}}\mathbf{1})$ are diagonal matrices with the degrees of the vertices and the degrees of the hyperedges on the diagonals. We note in (1) that the operator $D_V^{-\frac{1}{2}} H D_E^{-1} H^{\mathrm{T}} D_V^{-\frac{1}{2}}$ mixes the feature vectors of vertices connected by a common hyperedge. In this way, feature vectors of neighboring vertices of a vertex are propagated into the vertex and aggregated to form a new feature vector for the vertex. This model does not make use of hyperedge features.

3.2 Hypergraph collaborative network model

Unlike the hypergraph convolution networks in (1), the HCoN in [17, 18] aims to predict vertex-level and hyperedge-level labels together based on the hypergraph structure and a set of vertex features and hyperedge features. The single-layer hypergraph collaborative network we consider is given by

$$f(X_V, X_E|\theta) = \sigma\left(\tilde{H}\tilde{H}^{\mathrm{T}} X_V Q_V + \tilde{H} D_E^{-\frac{1}{2}} X_E Q_E\right) \quad (2)$$

$$g(X_V, X_E|\theta) = \sigma\left(\tilde{H}^{\mathrm{T}}\tilde{H} X_E P_E + \tilde{H}^{\mathrm{T}} D_V^{-\frac{1}{2}} X_V P_V\right). \quad (3)$$

In addition to the notations in (1), here $X_E \in \mathbf{R}^{M \times F_E}$ is a given matrix of hyperedge features, F_E is the dimension of the hyperedge features, $P_V \in \mathbf{R}^{F_V \times O}$, $Q_E \in \mathbf{R}^{F_E \times O}$ and $P_E \in \mathbf{R}^{F_E \times O}$ are the additional learnable parameters, $\theta = (Q_V, Q_E, P_V, P_E)$, and $\tilde{H} = D_V^{-\frac{1}{2}} H D_E^{-\frac{1}{2}}$ is the normalized incident matrix. The functions f and g are referred to as the vertex encoder and the hyperedge encoder, respectively. They are used to produce labels at vertices and hyperedges, respectively. In the sequel, we focus on the analysis of vertex encoder (2) because the analysis of (3) is essentially the same.

We remark that the HCoN model described in [18] reads

$$f(X_V, X_E|\theta) = \sigma\Big(\alpha\tilde{H}W\tilde{H}^{\mathrm{T}} X_V Q_V +$$
$$(1-\alpha)\tilde{H}W D_E^{-\frac{1}{2}} X_E Q_E\Big) \quad (4)$$

$$g(X_V, X_E|\theta) = \sigma\Big(\beta\tilde{H}^{\mathrm{T}} U\tilde{H} X_E P_E +$$
$$(1-\beta)\tilde{H}^{\mathrm{T}} U D_V^{-\frac{1}{2}} X_V P_V\Big). \quad (5)$$

In this work, we assume that the vertex weights W and hyperedge weights U are set to identity matrices for ease of presentation. We also absorb the constants α and β into the parameters Q_V, Q_E, P_V, and P_E; the degree of freedom of the model remains unchanged.

3.3 Activation function

Algorithmic stability concerns the change in the loss function value with respect to the change in the data. Since SGD aggregates gradients, it is very natural that the stability must rely on the regularity of the activation function and the loss function. The activation function $\sigma : \mathbf{R} \to \mathbf{R}$ is assumed to satisfy the following. Standard activations such as sigmoid, exponential linear unit (ELU), and tanh verify these assumptions. RELU fails the σ-smoothness. However, one can consider a smoothed RELU to restore the theory: $\sigma(x) = $
$$\begin{cases} 0, & \text{if } x < -\epsilon \\ \left(\frac{x+\epsilon}{2\epsilon}\right)^2, & \text{if } -\epsilon \leq x \leq \epsilon \\ x, & \text{if } x > \epsilon. \end{cases}$$

1) σ continuous differentiable.
2) σ-Lipschitz:

$$|\sigma(x) - \sigma(y)| \leq \alpha_\sigma|x - y|, \qquad \text{for all } x, y \in \mathbf{R}. \quad (6)$$

It follows that the derivative σ' is bounded, i.e., $|\sigma'(x)| \leq \alpha_\sigma$.
3) σ-smooth:

$$|\sigma'(x) - \sigma'(y)| \leq \nu_\sigma|x - y|, \quad \text{for all } x, y \in \mathbf{R}. \quad (7)$$

3.4 Loss function

Let $\hat{y}$ be an estimated label and let y be the true label. The loss function $\ell : [y_{\min}, y_{\max}] \times [y_{\min}, y_{\max}] \to \mathbf{R}^+$ is denoted by $\ell(\hat{y}, y)$. The following assumptions are made.

1) Continuous w.r.t. $(\hat{y}, y)$ and continuously differentiable w.r.t. $\hat{y}$.
2) ℓ-Lipschitz w.r.t. $\hat{y}$:

$$|\ell(\hat{y}, y) - \ell(\hat{y}', y)| \leq \alpha_\ell|\hat{y} - \hat{y}'| \quad (8)$$

for all $\hat{y}$, $\hat{y}'$, $y \in [y_{\min}, y_{\max}]$. This implies that $|\frac{\partial \ell}{\partial \hat{y}}(\hat{y}, y)| \leq \alpha_\ell$.
3) ℓ-smooth w.r.t. $\hat{y}$:

$$\left|\frac{\partial \ell}{\partial \hat{y}}(\hat{y}, y) - \frac{\partial \ell}{\partial \hat{y}}(\hat{y}', y)\right| \leq \nu_\ell|\hat{y} - \hat{y}'| \quad (9)$$

for all $\hat{y}, \hat{y}', y \in [y_{\min}, y_{\max}]$.

By the nonnegativity and continuity of $\ell(\cdot, \cdot)$ and the compactness of $[y_{\min}, y_{\max}] \times [y_{\min}, y_{\max}]$, the loss func-

tion is bounded:

$$0 \leq \ell(\hat{y}, y) \leq \gamma_\ell, \quad \text{for all } \hat{y}, y \in [y_{\min}, y_{\max}]. \qquad (10)$$

In the case of the binary cross-entropy $\ell(\hat{y}, y) = -y \ln \hat{y}$, the Lipschitzity condition does not hold. However, it can be easily remedied by rescaling or clipping $\hat{y}$ to the range of $[\epsilon, 1]$ for a $0 < \epsilon < 1$, so that $\ell(\hat{y}, y) \leq -\ln \epsilon$.

We remark that the following Lipschitzity of a composite function is assumed in [23]:

$$\left\| \nabla_\theta [\ell(f(v|\theta), y)] - \nabla_\theta [\ell(f(v|\theta'), y)] \right\|_2 \leq$$
$$\alpha_\ell \| \nabla_\theta f(v|\theta) - \nabla_\theta f(v|\theta') \|_2.$$

Here, $\| \cdot \|_2$ denotes the Euclidean norm and $f(v|\theta)$ denotes the output of a neural network with parameters θ at vertex v. The θ and θ' are two sets of parameters for the network. The ∇_θ denotes the gradient operator w.r.t. θ (i.e., $\frac{\partial}{\partial \theta}$). Our assumption (9) is solely in terms of $\frac{\partial \ell}{\partial \hat{y}}$ and is more fundamental.

3.5 Network outputs at a single vertex

In this subsection, we provide more details about the characteristics of the network outputs at each vertex and devise a bound of the hypergraph filter outputs that will be useful in the next section. We remark that a similar analysis can also be conducted for network outputs at each hyperedge. Here, we focus on the analysis of vertex propagation (2) only.

For a vertex $v \in \mathcal{V}$, let $\boldsymbol{e}(v)$ be a binary vector whose j-th entry is 1 if v is the j-th vertex and is 0 otherwise. Denote the learnable parameters with $\theta = (Q_V^T, Q_E^T)^T \in \mathbf{R}^{(F_V + F_E) \times O}$. The output (2) for a vertex v is given by

$$f(v|\theta) = \sigma\left(A_v Q_V + B_v Q_E\right) \in \mathbf{R}^{1 \times O}$$

where $A_v := \boldsymbol{e}(v)^T \tilde{H} \tilde{H}^T X_V$ and $B_v := \boldsymbol{e}(v)^T \tilde{H} D_E^{-\frac{1}{2}} X_E$. Note that A_v is a linear combination of the vertex feature vectors over the neighboring vertices of v and B_v is a linear combination of the edge feature vectors over the hyperedges joining v. For notational simplicity, we assume that the output dimension is $O = 1$. The analysis also works for a general O with minor modifications. Let

$$d_v = A_v Q_V + B_v Q_E.$$

Then, we have

$$f(v|\theta) = \sigma(d_v) \in \mathbf{R} \qquad (11)$$

$$\nabla_\theta f(v|\theta) = \sigma'(d_v)\left(A_v \ B_v\right)^T \in \mathbf{R}^{(F_V + F_E) \times 1}. \qquad (12)$$

Here, ∇_θ is the gradient operator w.r.t. θ and σ' is the derivative of σ.

Denote by $\| \cdot \|_2$ the matrix 2-norm. Note that

$$\|(A_v \ B_v)\|_2^2 = \left\| \boldsymbol{e}(v)^T \tilde{H} \tilde{H}^T X_V \right\|_2^2 + \left\| \boldsymbol{e}(v)^T \tilde{H} D_E^{-\frac{1}{2}} X_E \right\|_2^2 \leq$$
$$\|X_V\|_2^2 \| \tilde{H} \tilde{H}^T \|_2^2 + \|X_E\|_2^2 \| \tilde{H} \|_2^2 \| D_E^{-\frac{1}{2}} \|_2^2 \leq$$
$$\|X_V\|_2^2 \, \mu(\tilde{H})^4 + \|X_E\|_2^2 \, \mu(\tilde{H})^2$$

where $\mu(\tilde{H}) = \| \tilde{H} \|_2$ is the largest singular value of $\tilde{H}$. The last inequality follows from the fact that the diagonal entries of D_E are positive integers so that $\| D_E^{-\frac{1}{2}} \|_2 \leq 1$. Let

$$g_{\max} := \mu(\tilde{H}) \sqrt{\|X_V\|_2^2 \, \mu(\tilde{H})^2 + \|X_E\|_2^2}. \qquad (13)$$

It follows that

$$\left\| (A_v \ B_v) \right\|_2 \leq g_{\max}. \qquad (14)$$

Moreover, by (12) and (6), we have

$$\| \nabla_\theta f(v|\theta) \|_2 \leq \alpha_\sigma g_{\max}. \qquad (15)$$

Finally, we remark that if the two terms in the sum in (2) are weighted by α and $1 - \alpha$, respectively as described in (4) and (5), then (14) and (15) hold with

$$g_{\max} := \mu(\tilde{H}) \sqrt{\alpha^2 \|X_V\|_2^2 \, \mu(\tilde{H})^2 + (1-\alpha)^2 \|X_E\|_2^2}. \qquad (16)$$

3.6 SGD algorithm

Let $\mathcal{D}$ be an unknown joint distribution of the vertex and the associated label. Let

$$\mathcal{S} = \{(v_1, y_1), (v_2, y_2), \cdots, (v_n, y_n)\}$$

be a set of n i.i.d. samples from $\mathcal{D}$. The set $\mathcal{S}$ serves as the training set for the HCoN. The objective function of an HCoN is given by

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^n \ell(f(v_i|\theta), y_i).$$

The learning task is a transductive semi-supervised task. The incident matrix H and the feature matrices X_V and X_E are considered given. The formation of a training set is a sampling of vertices (and the associated labels) in the network. The training process (or learning process) refers to the minimization of $\mathcal{L}$ via SGD. A testing sample is another i.i.d. sample of a vertex in the network. The testing process refers to taking the output at the testing sample and comparing it to the known label. The sample space is therefore a finite space, to which the concentration inequality we used below applies.

We consider T iterations of the SGD algorithm, where the batch size is 1. At the t-th iteration, a sample

(v_{i_t}, y_{i_t}) is drawn from $\mathcal{S}$ with replacement. The parameters $\theta = (Q_V^{\mathrm{T}}, Q_E^{\mathrm{T}})^{\mathrm{T}}$ are updated as follows:

$$\theta_t = \theta_{t-1} - \eta \nabla_\theta \ell(f(v_{i_t}|\theta_{t-1}), y_{i_t})$$

for $t = 1, 2, \cdots, T$, where $\eta > 0$ is the learning rate. The final parameters learnt are θ_T, also denoted by $\theta(\mathcal{S}, A)$. The variable A denotes a particular randomization of SGD, i.e., the sequence $(i_1, i_2, \cdots, i_T)$.

Let $\mathcal{S}' = \{(v_1', y_1'), (v_2', y_2'), \cdots, (v_n', y_n')\}$ be a training set that differs from $\mathcal{S}$ with one sample. That is, for some $1 \le i^* \le n$,

$$\begin{cases} (v_i', y_i') = (v_i, y_i), & \text{if } i \ne i^* \\ (v_i', y_i') \ne (v_i, y_i), & \text{if } i = i^*. \end{cases}$$

Let $\theta' = (Q_V'^{\mathrm{T}}, Q_E'^{\mathrm{T}})^{\mathrm{T}}$ be the parameters learned with $\mathcal{S}'$. The SGD update is given by

$$\theta_t' = \theta_{t-1}' - \eta \nabla_\theta \ell(f(v_{i_t}'|\theta_{t-1}'), y_{i_t}')$$

for $t = 1, 2, \cdots, T$. The initial parameters for $\mathcal{S}$ and $\mathcal{S}'$ are set equal, i.e., $\theta_0 = \theta_0'$. The parameters learned, using the same randomization A as that for $\mathcal{S}$, are denoted by $\theta_T' = \theta'(\mathcal{S}', A)$.

The difference between the parameters θ_t and θ_t' at the t-th SGD iteration is

$$\Delta\theta_t := \theta_t - \theta_t' = \begin{bmatrix} Q_{V,t} - Q_{V,t}' \\ Q_{E,t} - Q_{E,t}' \end{bmatrix}.$$

Since $\mathcal{S}$ and $\mathcal{S}'$ are identical except for the i^*-th sample, if t^* is the first iteration at which (v_{i^*}, y_{i^*}) and (v_{i^*}', y_{i^*}') are sampled, we have $\Delta\theta_t = 0$ for all $t < t^*$. Note that

$$\Delta\theta_t = \Delta\theta_{t-1} - \eta[\nabla_\theta \ell(f(v_{i_t}|\theta_{t-1}), y_{i_t}) - \\ \nabla_\theta \ell(f(v_{i_t}'|\theta_{t-1}'), y_{i_t}')]. \tag{17}$$

4 Main results

Following the approach of [22, 23], we first establish the uniform stability of SGD and then obtain the generalization guarantees.

4.1 Uniform stability

Recall that $\mathcal{S}$ and $\mathcal{S}'$ are two training sets that differ only by the i^*-th sample. In Lemmas 1 and 2, we bound the gradient difference when the same sample and different samples are used, respectively. Then, we devise the difference between the SGD updates for $\mathcal{S}$ and $\mathcal{S}'$ in Lemma 3. The stability then follows in Theorem 1. Note that σ' is the derivative of σ. Other variables with a prime (') denote quantities derived from the perturbed training set $\mathcal{S}'$.

Lemma 1. At the t-th SGD iteration, we have

$$\left\| \nabla_\theta \ell(f(v|\theta_{t-1}), y) - \nabla_\theta \ell(f(v|\theta_{t-1}'), y) \right\|_2 \le \\ (\alpha_\ell \nu_\sigma + \nu_\ell \alpha_\sigma^2) g_{\max}^2 \|\Delta\theta_{t-1}\|_2.$$

Proof. Let $f = f(v|\theta_{t-1})$ and $f' = f(v|\theta_{t-1}')$. Let

$$d = A_v Q_{V,t-1} + B_v Q_{E,t-1}$$
$$d' = A_v Q_{V,t-1}' + B_v Q_{E,t-1}'.$$

By (11) and (12), we have $f = \sigma(d)$, $f' = \sigma(d')$, and

$$\nabla_\theta f = \sigma'(d) (A_v \ B_v)^{\mathrm{T}}$$
$$\nabla_\theta f' = \sigma'(d') (A_v \ B_v)^{\mathrm{T}}.$$

Note that

$$d - d' = (A_v \ B_v) \Delta\theta_{t-1}.$$

Therefore, by (14), we have

$$|d - d'| \le g_{\max} \|\Delta\theta_{t-1}\|_2. \tag{18}$$

Hence,

$$\|\nabla_\theta \ell(f, y) - \nabla_\theta \ell(f', y)\|_2 = \\ \left\| \frac{\partial}{\partial \hat{y}} \ell(f, y) \nabla_\theta f - \frac{\partial}{\partial \hat{y}} \ell(f', y) \nabla_\theta f' \right\|_2 \le \\ \left\| \frac{\partial}{\partial \hat{y}} \ell(f, y) \nabla_\theta f - \frac{\partial}{\partial \hat{y}} \ell(f, y) \nabla_\theta f' \right\|_2 + \\ \left\| \frac{\partial}{\partial \hat{y}} \ell(f, y) \nabla_\theta f' - \frac{\partial}{\partial \hat{y}} \ell(f', y) \nabla_\theta f' \right\|_2 = \\ \left| \frac{\partial}{\partial \hat{y}} \ell(f, y) \right| |\sigma'(d) - \sigma'(d')| \|(A_v \ B_v)\|_2 + \\ \left| \frac{\partial}{\partial \hat{y}} \ell(f, y) - \frac{\partial}{\partial \hat{y}} \ell(f', y) \right| |\sigma'(d')| \|(A_v \ B_v)\|_2 \le \\ \alpha_\ell \nu_\sigma |d - d'| g_{\max} + \nu_\ell |f - f'| \alpha_\sigma g_{\max} \le \\ \alpha_\ell \nu_\sigma |d - d'| g_{\max} + \nu_\ell \alpha_\sigma^2 |d - d'| g_{\max} \le \\ (\alpha_\ell \nu_\sigma + \nu_\ell \alpha_\sigma^2) g_{\max}^2 \|\Delta\theta_{t-1}\|_2.$$

□

Lemma 2. At the t-th SGD iteration, we have

$$\left\| \nabla_\theta \ell(f(v|\theta_{t-1}), y) - \nabla_\theta \ell(f(v'|\theta_{t-1}'), y') \right\|_2 \le 2\alpha_\ell \alpha_\sigma g_{\max}.$$

Proof. Let $f = f(v|\theta_{t-1})$ and $f' = f(v'|\theta_{t-1}')$. By (8) and (15), we have

$$\|\nabla_\theta \ell(f, y) - \nabla_\theta \ell(f', y')\|_2 = \\ \left\| \frac{\partial}{\partial \hat{y}} \ell(f, y) \nabla_\theta f - \frac{\partial}{\partial \hat{y}} \ell(f', y') \nabla_\theta f' \right\|_2 \le \\ \left| \frac{\partial}{\partial \hat{y}} \ell(f, y) \right| \|\nabla_\theta f\|_2 + \left| \frac{\partial}{\partial \hat{y}} \ell(f', y') \right| \|\nabla_\theta f'\|_2 \le \\ 2\alpha_\ell \alpha_\sigma g_{\max}.$$

□

Lemma 3. Let $\mathcal{S}$ and $\mathcal{S}'$ be two training sets that differ by one sample. Let $\theta = \theta(\mathcal{S}, A)$ and $\theta' = \theta'(\mathcal{S}', A)$ be the graph filter parameters of the HCoN models trained using SGD for T iterations on $\mathcal{S}$ and $\mathcal{S}'$, respectively. Then, the expected difference in the filter parameters is bounded by

$$\mathrm{E}_A[\|\Delta\theta_T\|_2] \leq \frac{\kappa_0}{n}$$

where

$$\kappa_0 := \frac{2\alpha_\ell\alpha_\sigma \left\{ \left[1 + \eta(\alpha_\ell\nu_\sigma + \nu_\ell\alpha_\sigma^2)g_{\max}^2\right]^T - 1 \right\}}{(\alpha_\ell\nu_\sigma + \nu_\ell\alpha_\sigma^2)g_{\max}}.$$

Proof. Let $1 \leq t \leq T$ and let

$$\boldsymbol{h} = \nabla_\theta\ell(f(v|\theta_{t-1}), y) - \nabla_\theta\ell(f(v'|\theta'_{t-1}), y').$$

By (17), we have

$$\mathrm{E}_A[\|\Delta\theta_t\|_2] \leq \mathrm{E}_A[\|\Delta\theta_{t-1}\|_2] + \eta\mathrm{E}_A[\|\boldsymbol{h}\|_2].$$

Note that the probabilities of the two scenarios considered in Lemmas 1 and 2 are $(n-1)/n$ and $1/n$, respectively. By Lemmas 1 and 2, we have

$$\mathrm{E}_A[\|\boldsymbol{h}\|_2] \leq \frac{n-1}{n} \times \mathrm{E}_A[C\|\Delta\theta_{t-1}\|_2] + \frac{1}{n} \times \mathrm{E}_A[C'] \leq$$
$$C\mathrm{E}_A[\|\Delta\theta_{t-1}\|_2] + \frac{1}{n} \times C'$$

where $C := (\alpha_\ell\nu_\sigma + \nu_\ell\alpha_\sigma^2)g_{\max}^2$ and $C' := 2\alpha_\ell\alpha_\sigma g_{\max}$. Hence, we have

$$\mathrm{E}_A[\|\Delta\theta_t\|_2] \leq (1 + \eta C)\mathrm{E}_A[\|\Delta\theta_{t-1}\|_2] + \frac{\eta C'}{n}.$$

Solving the above recursion with the initial condition $\Delta\theta_0 = 0$ yields

$$\mathrm{E}_A[\|\Delta\theta_T\|_2] \leq \frac{\eta C'}{n} \times \frac{(1 + \eta C)^T - 1}{\eta C} = \frac{\kappa_0}{n}.$$

$\square$

Next, we prove the uniform stability of the single-layer HCoN model trained using the SGD algorithm. Uniform stability has been introduced in [19] for the study of nonrandomized algorithms, where the algorithms are assumed to be insensitive to the order of the training set. However, SGD is a randomized algorithm that depends on the order of the random samples. In [22], the notion of uniform stability was extended to randomized algorithms. A randomized algorithm is said to be uniformly stable if there exists a constant β, possibly depending on n, such that

$$\sup_{\mathcal{S},\mathcal{S}',v,y} |\mathrm{E}_A[\ell(f(v|\theta), y) - \ell(f(v|\theta'), y)]| \leq \beta.$$

Here, $\theta = \theta(\mathcal{S}, A)$ and $\theta' = \theta'(\mathcal{S}', A)$ are the parameters learned with the datasets $\mathcal{S}$ and $\mathcal{S}'$, respectively. Thus the difference in the loss function values is averaged over all randomizations. In this paper, we adopt this notion of uniform stability to establish a bound of the generalization gap.

Theorem 1 (Uniform stability). The single-layer HCoN model trained using the SGD algorithm for T iterations is uniformly stable:

$$\sup_{\mathcal{S},\mathcal{S}',v,y} |\mathrm{E}_A[\ell(f(v|\theta), y) - \ell(f(v|\theta'), y)]| \leq \frac{\kappa}{n}$$

where

$$\kappa := \alpha_\ell\alpha_\sigma g_{\max}\kappa_0 =$$
$$\frac{2\alpha_\ell^2\alpha_\sigma^2 \left\{ \left[1 + \eta(\alpha_\ell\nu_\sigma + \nu_\ell\alpha_\sigma^2)g_{\max}^2\right]^T - 1 \right\}}{\alpha_\ell\nu_\sigma + \nu_\ell\alpha_\sigma^2}. \quad (19)$$

Proof. Let $f = f(v|\theta)$ and $f' = f(v|\theta')$. By (11), we have $f = \sigma(d)$ and $f' = \sigma(d')$, where

$$d = A_v Q_V + B_v Q_E$$
$$d' = A_v Q_V' + B_v Q_E'.$$

Hence, by (6), (8), (18) and Lemma 3, we have

$$|\mathrm{E}_A\left[\ell(f(v|\theta), y) - \ell(f(v|\theta'), y)\right]| \leq$$
$$\mathrm{E}_A\left[|\ell(f(v|\theta), y) - \ell(f(v|\theta'), y)|\right] \leq$$
$$\alpha_\ell\mathrm{E}_A\left[|f(v|\theta) - f(v|\theta')|\right] \leq$$
$$\alpha_\ell\alpha_\sigma\mathrm{E}_A\left[|d - d'|\right] \leq$$
$$\alpha_\ell\alpha_\sigma g_{\max}\mathrm{E}_A\left[\|\Delta\theta_T\|_2\right] \leq$$
$$\alpha_\ell\alpha_\sigma g_{\max} \times \frac{\kappa_0}{n} = \frac{\kappa}{n}.$$

$\square$

4.2 Generalization gap

Consider an HCoN trained on $\mathcal{S}$ with a randomization A. Denote the output of the trained network by $f(v|\theta(\mathcal{S}, A))$. Let $z = (v, y)$ be a random sample from $\mathcal{D}$. We denote the loss w.r.t. z by

$$\ell(\mathcal{S}, A, z) := \ell(f(v|\theta(\mathcal{S}, A)), y).$$

The generalization error is defined by

$$R(\mathcal{S}, A) := \mathrm{E}_z[\ell(\mathcal{S}, A, z)].$$

The empirical risk (a.k.a. training error) is defined by

$$R_{\mathrm{emp}}(\mathcal{S}, A) := \frac{1}{n}\sum_{i=1}^{n}\ell(\mathcal{S}, A, z_i)$$

where $z_i = (v_i, y_i)$ is the i-th sample in $\mathcal{S}$. The

generalization gap is given by

$$G(\mathcal{S}) = \mathrm{E}_A[R(\mathcal{S}, A) - R_{\mathrm{emp}}(\mathcal{S}, A)].$$

Next, we present a perturbation result for G. It is a general result for stable algorithms. The proof can be found in the Appendix of [23]. However, we also include the proof here for completeness and we have fixed a few minor typos of [23]. Recall that by Theorem 1, we have the uniform stability

$$|\mathrm{E}_A[\ell(f(v|\theta), y) - \ell(f(v|\theta'), y)]| \leq \frac{\kappa}{n}.$$

Lemma 5 (Gap perturbation). Let $\mathcal{S}$ and $\mathcal{S}'$ be two training sets that differ by their i^*-th samples. Then,

$$|G(\mathcal{S}) - G(\mathcal{S}')| \leq \frac{2\kappa + \gamma_\ell}{n}.$$

Here, κ is given by (19) and γ_ℓ is the upper bound of ℓ.
Proof. First, consider the perturbation of $R(\mathcal{S}, A)$:

$$
\begin{aligned}
&|\mathrm{E}_A\left[R(\mathcal{S}, A) - R(\mathcal{S}', A)\right]| = \\
&\left|\mathrm{E}_A\left[\mathrm{E}_z\left[\ell(\mathcal{S}, A, z) - \ell(\mathcal{S}', A, z)\right]\right]\right| = \\
&\left|\mathrm{E}_z\left[\mathrm{E}_A\left[\ell(\mathcal{S}, A, z) - \ell(\mathcal{S}', A, z)\right]\right]\right| \leq \\
&\mathrm{E}_z\left[\left|\mathrm{E}_A\left[\ell(\mathcal{S}, A, z) - \ell(\mathcal{S}', A, z)\right]\right|\right] \leq \frac{\kappa}{n}.
\end{aligned}
$$

Second, consider the perturbation of $R_{\mathrm{emp}}(\mathcal{S}, A)$:

$$
\begin{aligned}
&|\mathrm{E}_A\left[R_{\mathrm{emp}}(\mathcal{S}, A) - R_{\mathrm{emp}}(\mathcal{S}', A)\right]| \leq \\
&\left|\frac{1}{n}\sum_{j \neq i^*} \mathrm{E}_A\left[\ell(\mathcal{S}, A, z_j) - \ell(\mathcal{S}', A, z_j)\right]\right| + \\
&\left|\frac{1}{n}\mathrm{E}_A\left[\ell(\mathcal{S}, A, z_{i^*}) - \ell(\mathcal{S}', A, z_{i^*}')\right]\right| \leq \\
&\frac{1}{n}\sum_{j \neq i^*}\left|\mathrm{E}_A\left[\ell(\mathcal{S}, A, z_j) - \ell(\mathcal{S}', A, z_j)\right]\right| + \\
&\frac{1}{n}\mathrm{E}_A\left[\left|\ell(\mathcal{S}, A, z_{i^*}) - \ell(\mathcal{S}', A, z_{i^*}')\right|\right] \leq \\
&\frac{n-1}{n} \times \frac{\kappa}{n} + \frac{\gamma_\ell}{n} \leq \frac{\kappa + \gamma_\ell}{n}.
\end{aligned}
$$

Finally, we have

$$
\begin{aligned}
|G(\mathcal{S}) - G(\mathcal{S}')| \leq\ &\left|\mathrm{E}_A\left[R(\mathcal{S}, A) - R(\mathcal{S}', A)\right]\right| + \\
&\left|\mathrm{E}_A\left[R_{\mathrm{emp}}(\mathcal{S}, A) - R_{\mathrm{emp}}(\mathcal{S}', A)\right]\right|
\end{aligned}
$$

so that the result follows. □
For reference, we state the classical McDiarmid's concentration inequality[25], which provides the chance of an inequality with respect to a random training set.
Proposition 1 (McDiarmid's concentration inequality). If $F(\mathcal{S}) = F(z_1, z_2, \cdots, z_n)$ is a function of n i.i.d. random variables that satisfies

$$|F(\mathcal{S}) - F(\mathcal{S}')| \leq c$$

for all $\mathcal{S}$ and $\mathcal{S}'$ that differ by one coordinate, then, for any $\epsilon > 0$, we have

$$P\left(F(\mathcal{S}) \leq \mathrm{E}_{\mathcal{S}}[F(\mathcal{S})] + \epsilon\right) \geq 1 - \mathrm{e}^{-\frac{2\epsilon^2}{nc^2}}.$$

We are now in position to present our main result for the generalization gap.
Theorem 2 (Generalization gap). Consider a single-layer HCoN model trained on a dataset $\mathcal{S}$ using the SGD algorithm for T iterations. The following expected generalization gap holds for all $0 < \delta < 1$, with probability at least $1 - \delta$,

$$\mathrm{E}_A[R(\mathcal{S}, A) - R_{emp}(\mathcal{S}, A)] \leq \frac{\kappa}{n} + \frac{2\kappa + \gamma_\ell}{\sqrt{n}} \times \sqrt{\frac{\ln\frac{1}{\delta}}{2}}.$$

Here, κ is given by (19) and γ_ℓ is the upper bound of ℓ.
Proof. Let $F(\mathcal{S}) = G(\mathcal{S})$ be the generalization gap. By Lemma 5 and McDiarmid's concentration inequality, we have

$$P\{G(\mathcal{S}) \leq \mathrm{E}_{\mathcal{S}}[G(\mathcal{S})] + \epsilon\} \geq 1 - \mathrm{e}^{-\frac{2\epsilon^2}{nc^2}}$$

where $c := \frac{2\kappa + \gamma_\ell}{n}$. Let $\delta = \mathrm{e}^{-\frac{2\epsilon^2}{nc^2}}$. Note that

$$\epsilon = c\sqrt{\frac{n\ln\frac{1}{\delta}}{2}} = \frac{2\kappa + \gamma_\ell}{\sqrt{n}} \times \sqrt{\frac{\ln\frac{1}{\delta}}{2}}.$$

Hence, the following holds with a probability of at least $1 - \delta$:

$$\mathrm{E}_A[R(\mathcal{S}, A) - R_{\mathrm{emp}}(\mathcal{S}, A)] \leq \mathrm{E}_{\mathcal{S}}[G(\mathcal{S})] + \frac{2\kappa + \gamma_\ell}{\sqrt{n}} \times \sqrt{\frac{\ln\frac{1}{\delta}}{2}}.$$

It remains to be shown that $\mathrm{E}_{\mathcal{S}}[G(\mathcal{S})] \leq \kappa/n$. Note that

$$
\begin{aligned}
\mathrm{E}_{\mathcal{S}}[G(\mathcal{S})] =\ &\mathrm{E}_{\mathcal{S}}[\mathrm{E}_A[R(\mathcal{S}, A) - R_{\mathrm{emp}}(\mathcal{S}, A)]] = \\
&\mathrm{E}_{\mathcal{S}}[\mathrm{E}_A[\mathrm{E}_z[\ell(\mathcal{S}, A, z)]]] - \\
&\frac{1}{n}\sum_{j=1}^n \mathrm{E}_{\mathcal{S}}[\mathrm{E}_A[\ell(\mathcal{S}, A, z_j)]] = \\
&\mathrm{E}_{\mathcal{S}, z}[\mathrm{E}_A[\ell(\mathcal{S}, A, z)]] - \mathrm{E}_{\mathcal{S}}[\mathrm{E}_A[\ell(\mathcal{S}, A, z_i)]] = \\
&\mathrm{E}_{\mathcal{S}, z_i'}[\mathrm{E}_A[\ell(\mathcal{S}, A, z_i')]] - \mathrm{E}_{\mathcal{S}, z_i'}[\mathrm{E}_A[\ell(\mathcal{S}', A, z_i')]] \leq \\
&\mathrm{E}_{\mathcal{S}, z_i'}[|\mathrm{E}_A[\ell(\mathcal{S}, A, z_i') - \ell(\mathcal{S}', A, z_i')]|] \leq \\
&\mathrm{E}_{\mathcal{S}, z_i'}\left[\frac{\kappa}{n}\right] = \frac{\kappa}{n}.
\end{aligned}
$$

The last inequality is due to Theorem 1. □
Theorem 2 states that as $n \to \infty$, the gap converges to zero at the rate of $O(1/\sqrt{n})$, provided that κ in (19) does not grow with n and the network size (M and N). It boils down to the requirement that the $g_{\max}$ given in (13) does not grow with n, M and N. In general, $g_{\max}$ can grow with the size of the network. For example, if the

entries of X_V and X_E are uniformly distributed on the interval $[0,1]$, then $\|X_V\|_2 = O(\sqrt{NF_V})$ and $\|X_E\|_2 = O(\sqrt{MF_E})$. We therefore normalize each column of X_V to a unit vector. Thus, we have $\|X_V\|_2 \leq \|X_V\|_F = \sqrt{F_V}$, which is a constant independent of the graph size. Here, $\|\cdot\|_F$ denotes the Frobenius norm. Likewise, we also normalize X_E so that $\|X_E\|_2 \leq \sqrt{F_E}$. Another factor that may affect the growth of $g_{\max}$ is the incidence matrix H. When H is normalized to $\tilde{H}$, it can be shown that the dominant singular value $\mu(\tilde{H})$ is bounded above by 1 (see [18]). In contrast, we have $\mu(H) = O(\sqrt{NM})$. With both kinds of normalization in place, we have

$$g_{\max} \leq \sqrt{F_V + F_E} = O(1)$$

a constant independent of the graph size. Likewise, $g_{\max}$ in (16) is bounded by $\sqrt{\alpha^2 F_V + (1-\alpha)^2 F_E} = O(1)$.

It is worthwhile to compare our results with some related works. In the study of stable algorithms in [19], the learnt model θ is assumed to be a global optimizer of the loss function, which is assumed to be convex. There are no SGD iterations. As such, the corresponding constant κ is independent of T. However, the assumption of global optimality is impractical. The work in [22] considers regression models with $\theta_1, \cdots, \theta_T$ generated with SGD. The constant κ is shown to be $O(T)$. However, there are also some rather strong convexity assumptions in the model that are not applicable to neural networks with nonlinearities. The bound in [23] for GCNs has the same kind of exponential dependence on T as ours. This is due to the lack of convexity in the model. Moreover, SGD does not guarantee a monotonic reduction of the loss function even when the learning rate is coupled with a line search. Thus the situation encountered in [23] and in this paper is more sophisticated but more practical. However, we manage to establish the convergence with respect to the size n of the training set.

The result indicates the consistency between training and testing errors, which is important for the model to be useful. Of course, it relies on the well-known fundamental assumption that the training and testing sets are drawn from the same distribution. In Section 5, we will also numerically study the generalization gap with respect to different parameters.

In the paper by [17, 18], vertex and hyperedge classification is studied. Experimental results on several benchmark datasets have shown that the performance of the hypergraph collaborative network is better than that of the baseline methods. Our theoretical results for the stability and generalization of hypergraph collaborative networks can further confirm their usefulness.

Finally, we would like to remark that our theoretical analysis is also valid for hypergraph convolution networks (HCNs) in (1), where propagation of embedding is done on vertices only. The HCoN model reduces to an HCN when the hyperedge feature matrix X_E is set to the zero matrix and the hyperedge feature dimension F_E is set to zero.

Corollary 1 (Generalization gap). Consider a single layer HCN model trained on a dataset $\mathcal{S}$ using the SGD algorithm for T iterations. The following expected generalization gap holds for all $0 < \delta < 1$, with probability at least $1 - \delta$,

$$\mathrm{E}_A[R(\mathcal{S}, A) - R_{emp}(\mathcal{S}, A)] \leq \frac{\kappa}{n} + \frac{2\kappa + \gamma_\ell}{\sqrt{n}} \times \sqrt{\frac{\ln \frac{1}{\delta}}{2}}.$$

Here, κ is given by (19) and γ_ℓ is the upper bound of ℓ.

5 Experiments

The purpose of this section is to numerically study the behavior of the generalization gap of HCoNs. We refer the reader to [18] for the accuracy of the model.

5.1 Datasets

We use the widely used benchmark datasets of Citeseer[26], Cora[27] and PubMed[28] for evaluations. The three networks consist of 1 498, 16 313, and 3 840 vertices, respectively.

Citeseer and PubMed are cocitation datasets. In the hypergraph, each vertex represents a document. The set of citations of a document forms a hyperedge. The vertex features for Citeseer and PubMed are the bag-of-words vector representations and the term frequency-inverse document frequency (TF-IDF), respectively. Cora is a coauthorship dataset. Each vertex represents a document and each hyperedge represents an author connecting to documents of the author. The vertex features are the TF-IDF vectors of the documents. A more detailed description of the feature vector generation process can be found in [18]. In each of the three networks, 30% of the vertices are used as the test set; 30%−70% of the vertices are used as the training set.
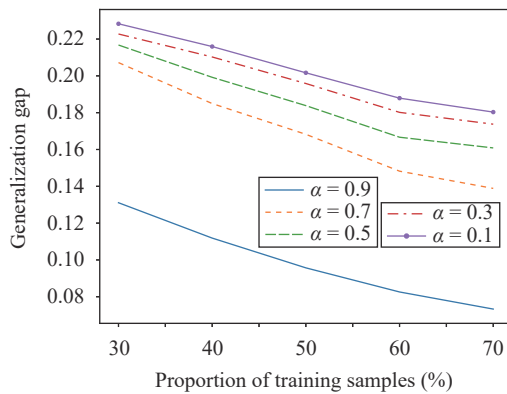
5.2 Effect of the weight

Fig. 1 shows the generalization gap as a function of the size of the training set. The weight α in (4) and (5) is varied to see its effects. The gap values are the average of 10 different randomizations to serve as a proxy for the expectation $\mathrm{E}_A[\cdot]$. As the size of the training set increases, the generalization gap decreases. This is predicted with our theory that the gap decreases as the size of the training set increases. For a fixed size of the training set, the gap is smaller as α increases; this indicates that for these datasets, the vertex features are more important for prediction than the hyperedge features.
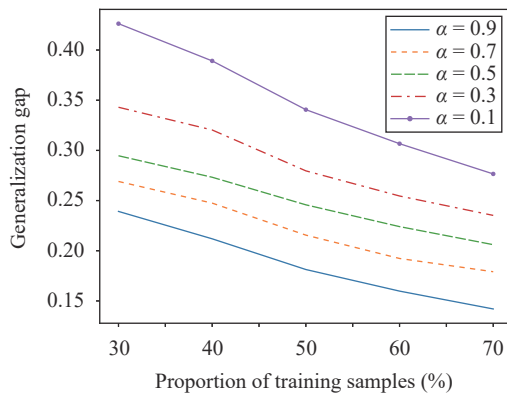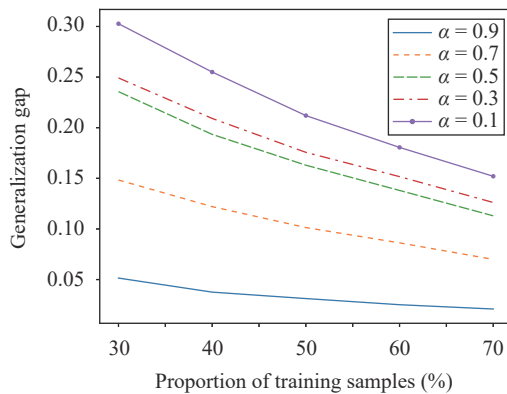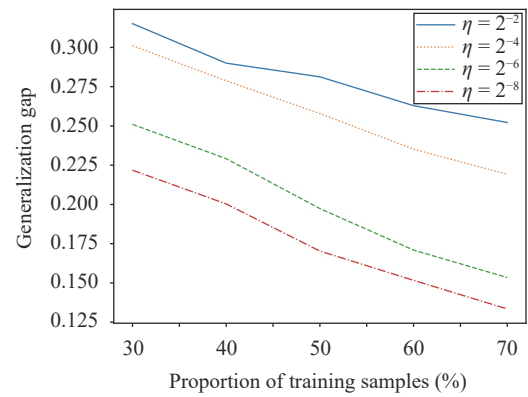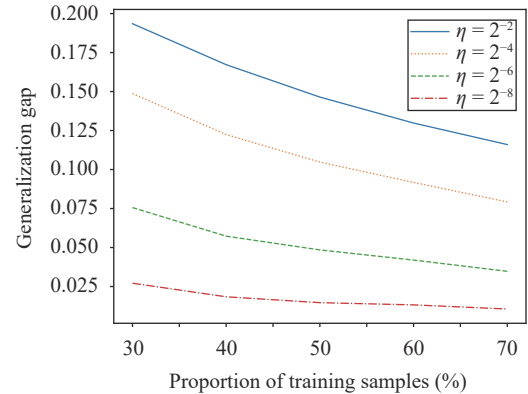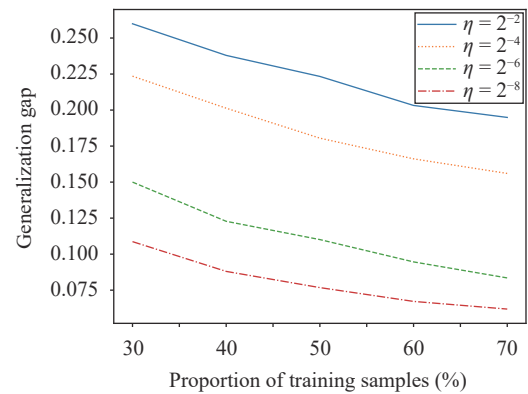
Fig. 1   Convergence of generalization for various $\alpha's$ as the training size n increases.


Fig. 2   Convergence of generalization for various $\eta's$ as the training size n increases.

5.3   Effect of the learning rate

Fig. 2 shows the generalization gap as a function of the size of the training set for different learning rates. The gap values are the average of 10 different randomizations. The generalization gap reduces as the size of the training set n increases. Moreover, the gap decreases with the learning rate for each fixed n. Such phenomena are consistent with the bound devised in Theorem 2.

To illustrate the convergence of the training process, we show the training and test loss function values at each epoch in Fig. 3. To avoid overwhelming with too many

figures, we show only the result for 70% training sample, $\alpha = 0.9$ and $\eta = 0.01$. The loss function values are the average of 10 different randomizations. The graph shows that: 1) The training loss function value is monotonically decreasing, and hence, the network fits better to the training data; 2) The test loss function value is also monotonically decreasing, and hence, the generalization error is improving. The starting training and test loss function values are similar because the network parameters are initialized randomly. The difference between the training and the test loss function values at the final epoch constitutes the generalization gap reported in
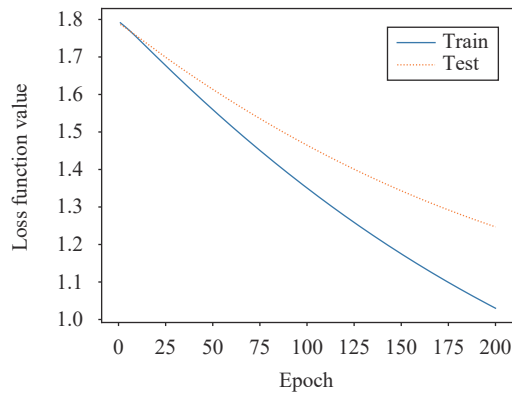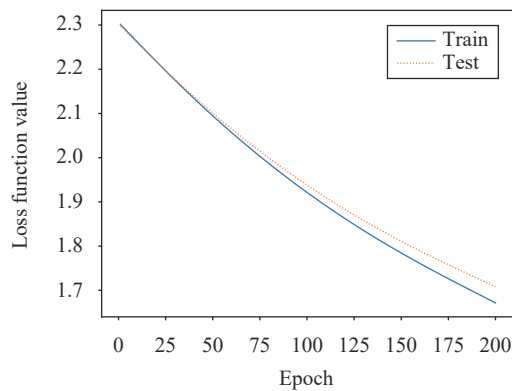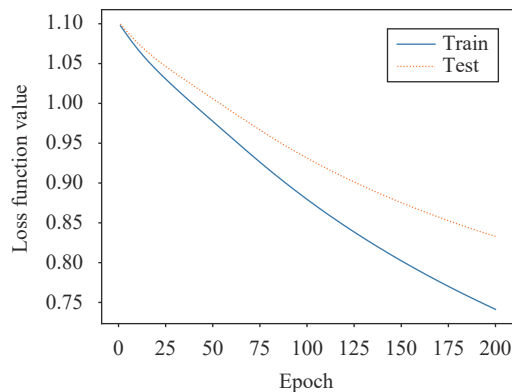
Fig. 3   Convergence of the training process as the number of iterations T increases

Fig. 4   Convergence of the generalization gap as the number of iterations T increases

Figs. 1 and 2. Our theory predicts that the gap at a fixed T decreases as the size n of the training set increases; see Figs. 1 and 2 for the convergence of the gap as a function of n.
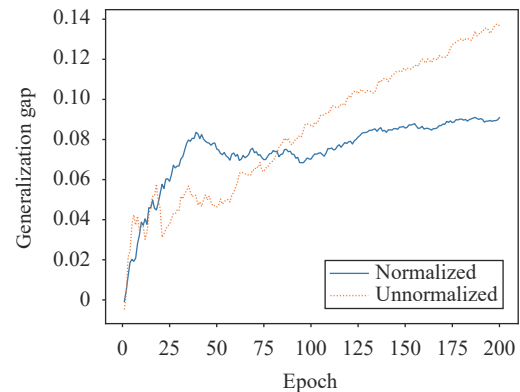
5.4 Effect of incidence matrix normalization

In the experiment, we demonstrate the effect of normalization of the incidence matrix H. The normalized model is given in (4) and (5). The unnormalized model is obtained by replacing $\tilde{H}$ in (4) and (5) with H. Fig. 4

shows the generalization gap as the SGD progresses. The size of the training set is fixed to 70%. The α is fixed to 0.9 because it yields the lowest gap and highest accuracy (see [18]). Each epoch represents a cycle of iterations over the whole training set. The gap values are obtained from a single randomization. First, the results show that when H is normalized to $\tilde{H}$, the gap reaches a smaller level and converges faster as the number of iterations increases. This is predicted with our theoretical bound of the generalization gap since $\mu(\tilde{H}) \leq 1$ whereas $\mu(H) = O(\sqrt{NM})$. The theory thus explains why normalization is

important. Second, regarding the results with normalized incidence matrices, although the theoretical bound grows with T (the number of iterations), the gap stabilizes quickly in practice. This is because the bound represents the worst-case scenario, which is pessimistic.

6 Conclusions

In this paper, we have established uniform stability and generalization guarantee for single-layer hypergraph collaborative networks. The results show the importance of the normalization of the vertex and hyperedge features and normalization of the hypergraph incidence matrix H. With these normalizations, we have a bound of the generalization gap independent of the graph size, and therefore, the generalization gap converges to zero. Some numerical experiments are presented to examine the convergence of the generalization gap in practice.

Several future research directions will be considered. 1) Extend the analysis to multilayer HCoNs, which are more useful in practice. The main challenge is that the gradients of the network become highly nonlinear and are more difficult to estimate. 2) Consider more general first-order stochastic optimization algorithms to include other commonly used algorithms such as SGD with momentum and adaptive moment estimation (ADAM). The problem is to devise estimates that can precisely reflect the potential acceleration delivered by these algorithms and study how the acceleration affects the stability and generalization gap. 3) While the present result has an advantage in that it does not assume any specific data distribution, it would also be useful to analyze the generalization gap in the presence of a data distribution. The idea is to improve the estimation of the expectations $E_{z\sim\mathcal{D}}$ and $E_{\mathcal{S}}$ in Lemma 5, Proposition 1, and Theorem 2. Specifically, determine the distribution of the gap from the assumed distribution of data, and the devise a special case of McDiarmid's concentration inequality with an improved bound.

Acknowledgements

Declarations of conflict of interest

Open Access

References

[1] L. Yao, C. S. Mao, Y. Luo. Graph convolutional networks for text classification. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, Honolulu, USA, pp. 7370–7377, 2018. DOI: 10.1609/aaai.v33i01.33017370.

[2] H. Xu, Y. Ma, H. C. Liu, D. Deb, H. Liu, J. L. Tang, A. K. Jain. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, vol. 17, no. 2, pp. 151–178, 2020. DOI: 10.1007/s11633-019-1211-x.

[3] Z. D. Chen, L. S. Li, J. Bruna. Supervised community detection with line graph neural networks. In *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, USA, 2020.

[4] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, G. E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, pp. 1263–1272, 2017.

[5] G. S. Te, W. Hu, A. M. Zheng, Z. Guo. RGCNN: Regularized graph CNN for point cloud segmentation. In *Proceedings of the 26th ACM international conference on Multimedia*, Seoul, Republic of Korea, pp. 746–754, 2018. DOI: 10.1145/3240508.3240621.

[6] O. Litany, A. Bronstein, M. Bronstein, A. Makadia. Deformable shape completion with graph convolutional autoencoders. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Salt Lake City, USA, pp. 1886–1895, 2018. DOI: 10.1109/CVPR.2018.00202.

[7] Y. Rui, V. I. S. Carmona, M. Pourvali, Y. Xing, W. W. Yi, H. B. Ruan, Y. Zhang. Knowledge mining: A cross-disciplinary survey. *Machine Intelligence Research*, vol. 19, no. 2, pp. 89–114, 2022. DOI: 10.1007/s11633-022-1323-6.

[8] T. N. Kipf, M. Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France, 2017. DOI: 10.48550/arXiv.1609.02907.

[9] R. Y. Li, S. Wang, F. Y. Zhu, J. Z. Huang. Adaptive graph convolutional neural networks. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, New Orleans, USA, pp. 3546–3553, 2018. DOI: 10.1609/aaai.v32i1.11691.

[10] F. P. Such, S. Sah, M. A. Dominguez, S. Pillai, C. Zhang, A. Michael, N. D. Cahill, R. Ptucha. Robust spatial filtering with graph convolutional neural networks. *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 884–896, 2017. DOI: 10.1109/JSTSP.2017.2726981.

[11] D. Y. Zhou, J. Y. Huang, Schölkopf, B. Learning with hypergraphs: Clustering, classification, and embedding. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, Vancouver, Canada, vol. 19, pp. 1601–1608, 2006.

[12] Y. Gao, Z. Z. Zhang, H. J. Lin, X. B. Zhao, S. Y. Du, C. Q. Zou. Hypergraph learning: Methods and practices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 5, pp. 2548–2566, 2022.

[13] Y. F. Feng, H. X. You, Z. Z. Zhang, R. R. Ji, Y. Gao. Hypergraph neural networks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, Honolulu, USA, vol. 33, pp. 3558–3565, 2019. DOI: 10.1609/aaai.v33i01.33013558.

[14] S. Bai, F. H. Zhang, P. H. S. Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, vol. 110, Article number 107637, 2021. DOI: 10.1016/j.patcog.2020.107637.

[15] Y. H. Dong, W. Sawin, Y. Bengio. HNHN: Hypergraph networks with hyperedge neurons. In *Graph Representations and Beyond Workshop at International Conference on Machine Learning*, [Online], Available: https://arxiv.org/abs/2006.12278, 2020.

[16] N. Yadati, M. Nimishakavi, P. Yadav, V. Nitin, A. Louis, P. Talukdar. HyperGCN: A new method of training graph convolutional networks on hypergraphs. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Vancouver, Canada, pp. 1511–1522, 2019.

[17] H. R. Wu, M. K. Ng. Hypergraph convolution on nodes-hyperedges network for semi-supervised node classification. *ACM Transactions on Knowledge Discovery from Data*, vol. 16, no. 4, Article number 80, 2022. DOI: 10.1145/3494567.

[18] H. R. Wu, Y. G. Yan, M. K. Ng. Hypergraph collaborative network on vertices and hyperedges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 3, pp. 3245–3258, 2023. DOI: 10.1109/TPAMI.2022.3178156.

[19] O. Bousquet, A. Elisseeff. Stability and generalization. *The Journal of Machine Learning Research*, vol. 2, pp. 499–526, 2002. DOI: 10.1162/153244302760200704.

[20] S. Mukherjee, P. Niyogi, T. Poggio, R. Rifkin. Learning theory: Stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Advances in Computational Mathematics*, vol. 25, no. 1–3, pp. 161–193, 2006. DOI: 10.1007/s10444-004-7634-z.

[21] A. S. Nemirovski, D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization*, New York, USA: Wiley Interscience, 1983.

[22] M. Hardt, B. Recht, Y. Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, New York, USA, pp. 1225–1234, 2015.

[23] S. Verma, Z. L. Zhang. Stability and generalization of graph convolutional neural networks. In *Proceedings of the 25th ACM/SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, Anchorage, USA, pp. 1539–1548, 2019. DOI: 10.1145/3292500.3330956.

[24] M. Belkin, I. Matveeva, P. Niyogi. Regularization and semi-supervised learning on large graphs. In *Proceedings of the 17th International Conference on Learning Theory*, Springer, Banff, Canada, pp. 624–638, 2004. DOI: 10.1007/978-3-540-27819-1_43.

[25] C. McDiarmid. On the method of bounded differences. *Surveys in Combinatorics*, J. Siemons, Ed., Cambridge, UK: Cambridge University Press, pp. 148–188, 1989. DOI: 10.1017/CBO9781107359949.008.

[26] I. Bhattacharya, L. Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, Article number 5, 2007. DOI: 10.1145/1217299.1217304.

[27] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, vol. 29, no. 3, Article number 93, 2008. DOI: 10.1609/aimag.v29i3.2157.

[28] G. Namata, B. London, L. Getoor, B. Huang. Query-driven active surveying for collective classification. In *Proceedings of Workshop on Mining and Learning with Graphs*, Edinburgh, UK, 2012.

Michael K. Ng received the B. Sc. and M. Phil. degrees in mathematics from The University of Hong Kong, China in 1990 and 1992, respectively, and the Ph. D. degree in mathematics from The Chinese University of Hong Kong, China in 1995. He was a research fellow of Computer Sciences Laboratory at Australian National University, Australia from 1995 to 1997, and an assistant/associate professor of the University of Hong Kong, China from 1997 to 2005. He was professor/Chair professor in Department of Mathematics at Hong Kong Baptist University, China from 2006 to 2019. He is currently a chair professor in Research Division of mathematical and statistical science at The University of Hong Kong, China. He is selected for the 2017 class of fellows of the Society for Industrial and Applied Mathematics. He obtained the Feng Kang Prize for his significant contributions in scientific computing. He serves on the edit-

orial board members of several international journals.

His research interests include bioinformatics, image processing, scientific computing and data mining.

E-mail: mng@maths.hku.hk

ORCID iD: 0000-0001-6833-5227
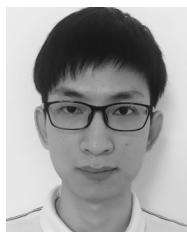
Hanrui Wu received the B. Sc. and Ph. D. degrees in software engineering from School of Software Engineering, South China University of Technology, China in 2013 and 2020, respectively. He is currently an associate professor with Department of Computer Science, Jinan University, China. Before that, he was a postdoctoral research fellow with Department of Mathematics, The University of Hong Kong, China from 2020 to 2021.

His research interests include transfer learning, zero-shot learning, hypergraph learning, recommendation systems and brain-computer interactions.

E-mail: hrwu@outlook.com

Andy Yip received the Ph. D. degree in mathematics from University of California at Los Angeles, USA in 2005. He is currently a senior researcher with The University of Hong Kong, China. He served as an assistant professor with National University Singapore, Singapore, and a lecturer with Hong Kong Baptist University, China. Since 2017, he has worked in the data science and finance industries.

His research interests include data science and image processing.

E-mail: mhyipa@hotmail.com (Corresponding author)

ORCID ID: 0000-0002-3965-2802