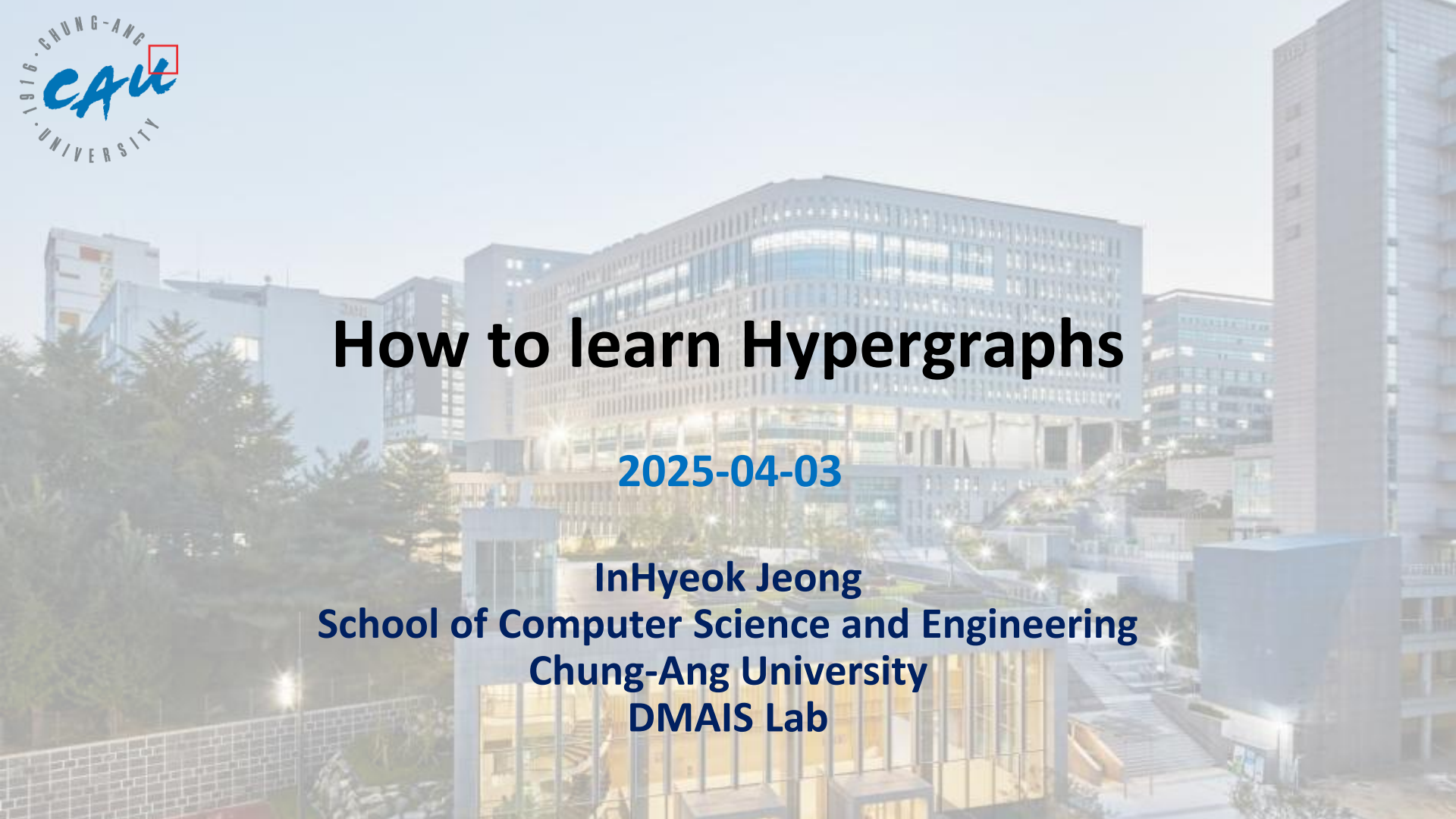


How to learn Hypergraphs

2025-04-03

InHyeok Jeong
School of Computer Science and Engineering
Chung-Ang University
DMAIS Lab

Content

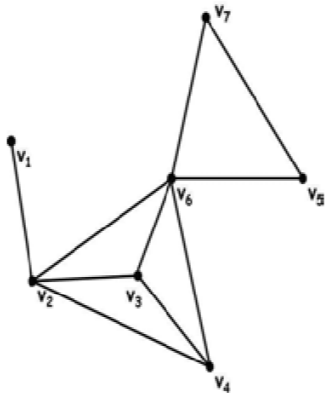
- **Introduction**
  - What is Hypergraph?
  - Why the hypergraph is important?
- **Preliminaries**
- **Several methods**
  - Generalize spectral clustering to hypergraphs
  - Apply Graph Convolution Network to hypergraphs
  - Propose a novel expansion
- **Conclusion**
- **Appendix**

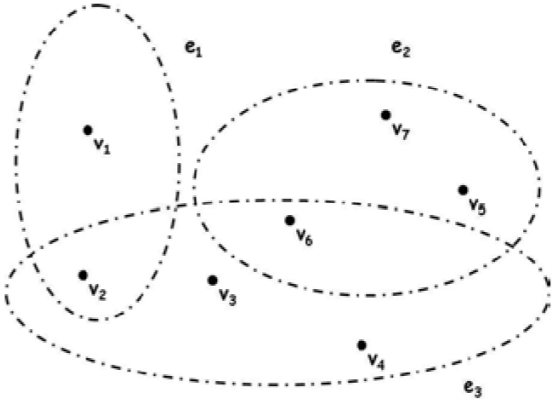
Introduction

What is Hypergraph?

- A graph in which an edge can connect more than two vertices
  - Edge is a subset of vertices

	e_1	e_2	e_3
v_1	1	0	0
v_2	1	0	1
v_3	0	0	1
v_4	0	0	1
v_5	0	1	0
v_6	0	1	1
v_7	0	1	0
v_2	1	0	1



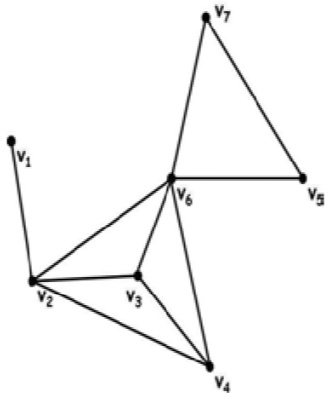


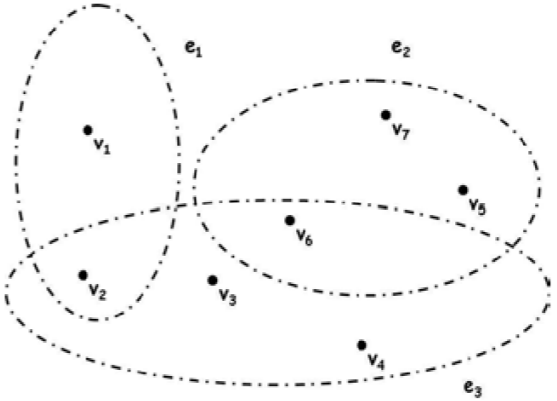
Introduction

Why the hypergraph is important?

- The hypergraph can represent more complex relationships among the objects

	e_1	e_2	e_3
v_1	1	0	0
v_2	1	0	1
v_3	0	0	1
v_4	0	0	1
v_5	0	1	0
v_6	0	1	1
v_7	0	1	0
v_2	1	0	1

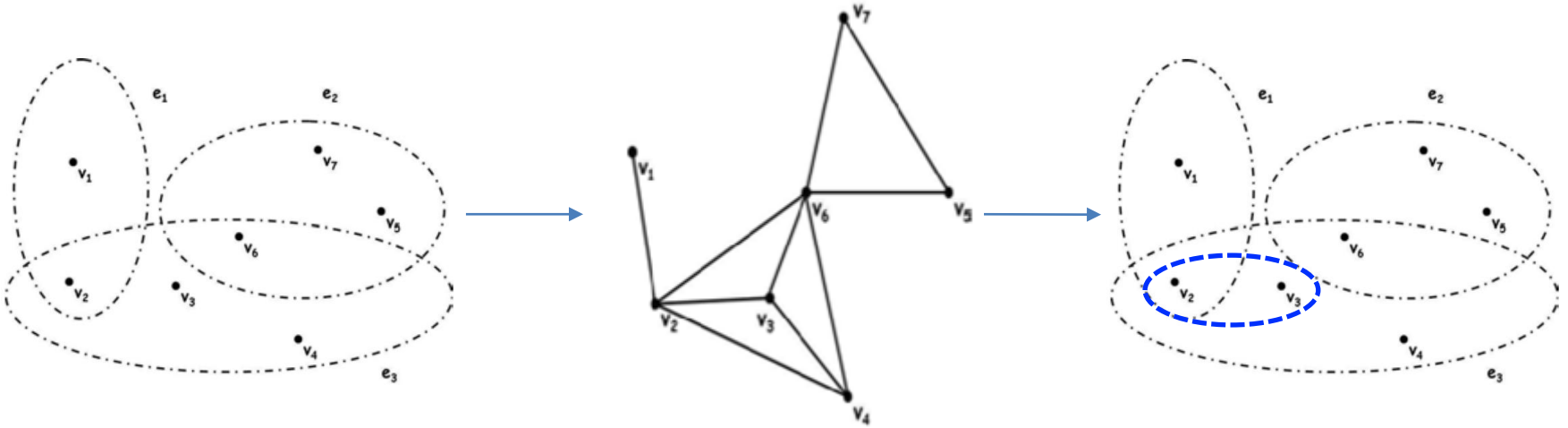




Introduction

- **Why the hypergraph is important?**
  - Unable to recover hypergraphs from simple graphs

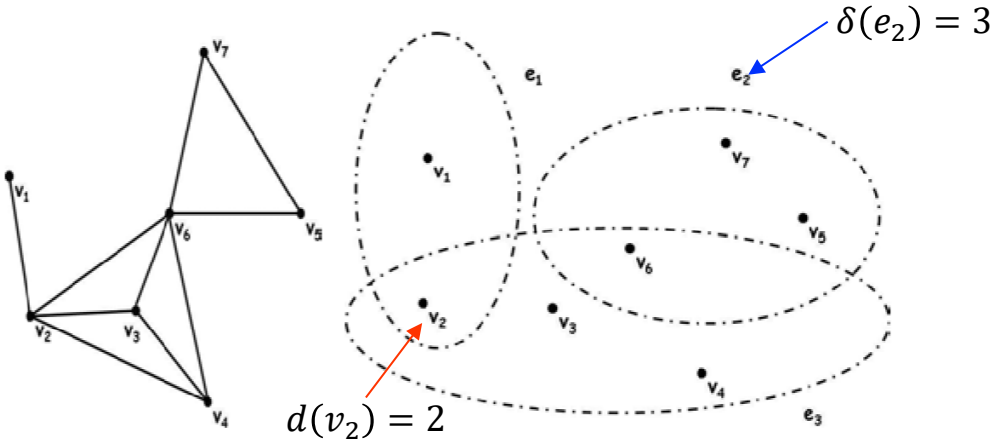
    → *Information loss!*



Preliminaries

- **A hypergraph $G = (V, E, w)$**
  - V : a finite set of objects, E : a family of subsets e of V s.t $\cup_{e \in E} = V$
  - w : a weight of hyperedge
  - $|V| \times |E|$ matrix H with entries $h(v, e) = 1$ if $v \in e$ and 0 otherwise
  - $d(v) = \sum_{e \in E} w(e) h(v, e)$, $\delta(e) = \sum_{v \in V} h(v, e)$

	e_1	e_2	e_3
v_1	1	0	0
v_2	1	0	1
v_3	0	0	1
v_4	0	0	1
v_5	0	1	0
v_6	0	1	1
v_7	0	1	0
v_2	1	0	1

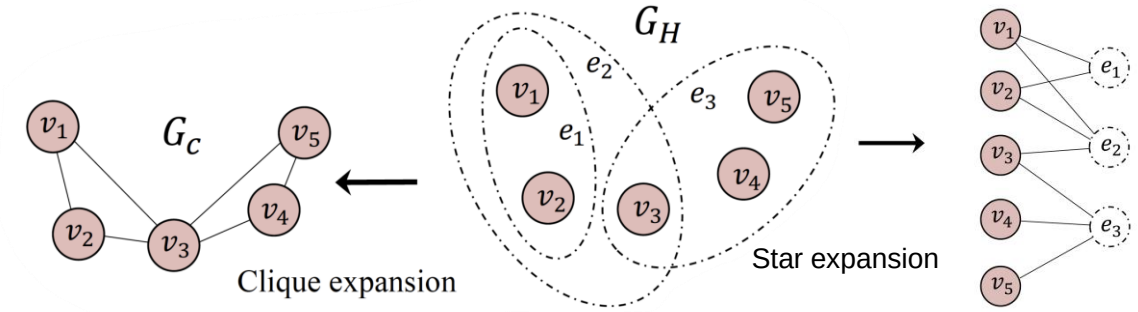


Preliminaries

How to transform hypergraph to simple graph

- The clique expansion
  - Replace each hypergraph with a clique, i.e., a fully connected subgraph
  - Loss of information

- The Star Expansion
  - Induce a heterogeneous graph structure
  - No explicit vertex-vertex link



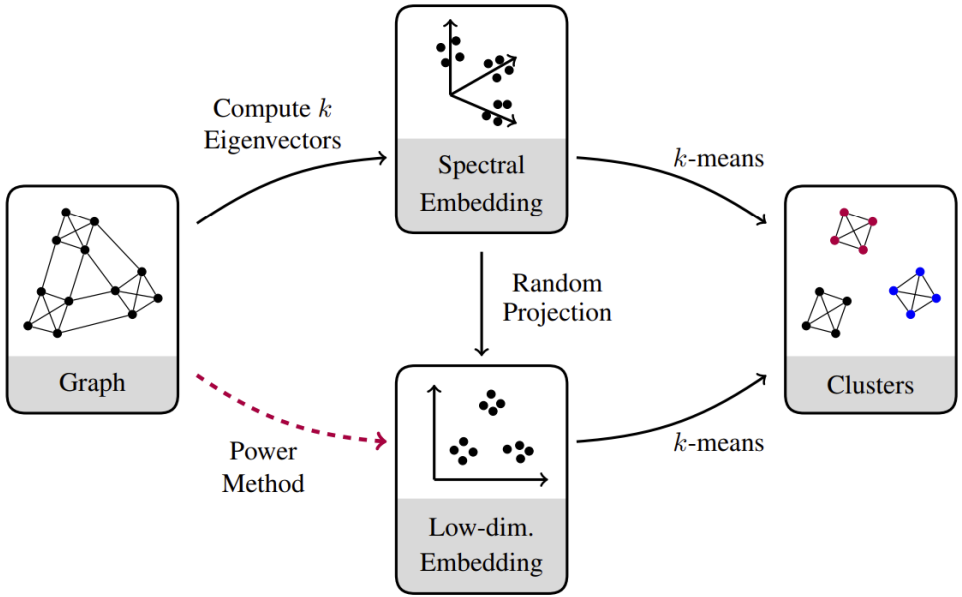
	e_1	e_2	e_3
v_1	1	1	0
v_2	1	1	0
v_3	0	1	1
v_4	0	0	1
v_5	0	0	1

Incidence matrix H

Generalize spectral clustering to hypergraphs

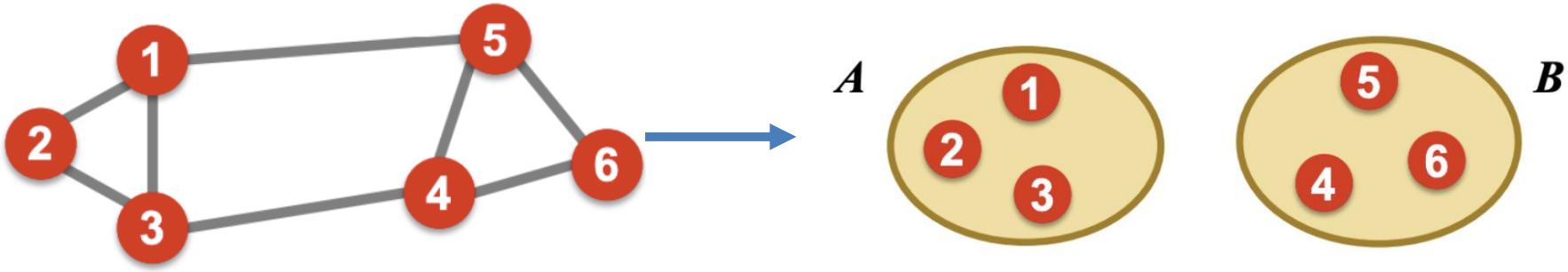
What is spectral clustering?

- An algorithm designed to find k clusters in a graph G
- Use eigenvectors of the graph Laplacian matrix



Generalize spectral clustering to hypergraphs

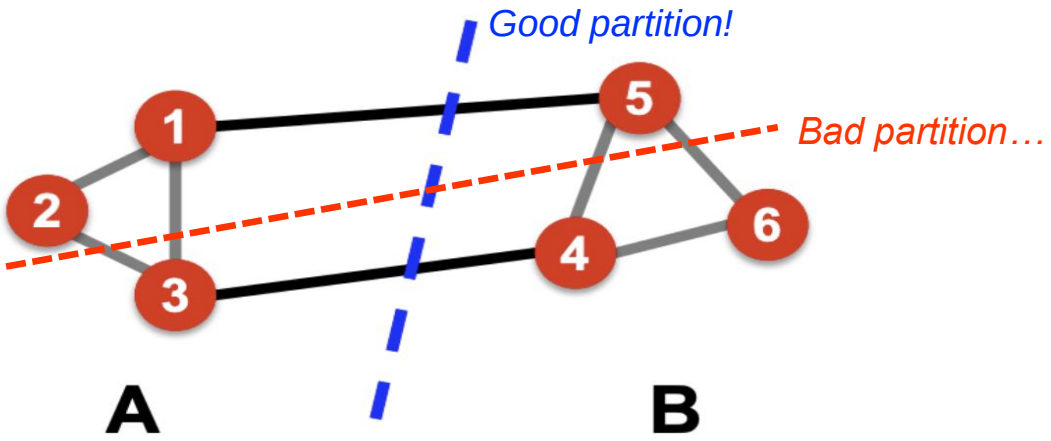
- **Graph Partitioning**
  - Bi-partitioning task
    - Divide vertices into two disjoint groups $\boldsymbol{A}, \boldsymbol{B}$



How can we define a "good" partition of G?

Generalize spectral clustering to hypergraphs

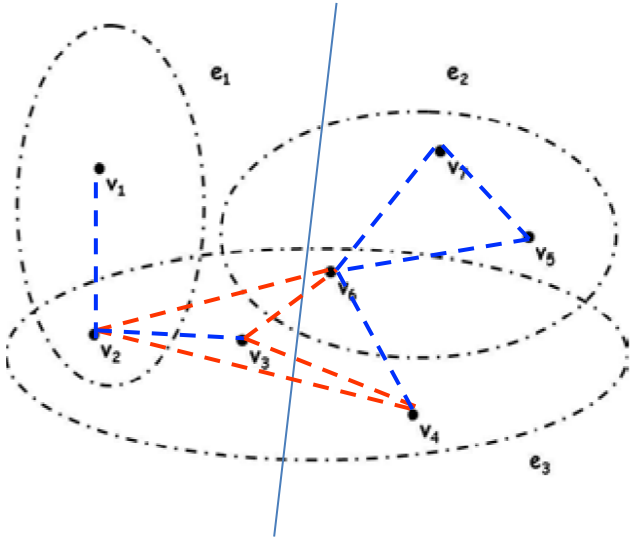
- **What makes a good partition?**
  - Maximize the number of within-group connections
  - Minimize the number of between-group connections



We must define hyperedge cut!

Generalize spectral clustering to hypergraphs

- **Normalized hypergraph cut**
  - Regard each hyperedge as a clique
  - Assign the same weight to all subedges

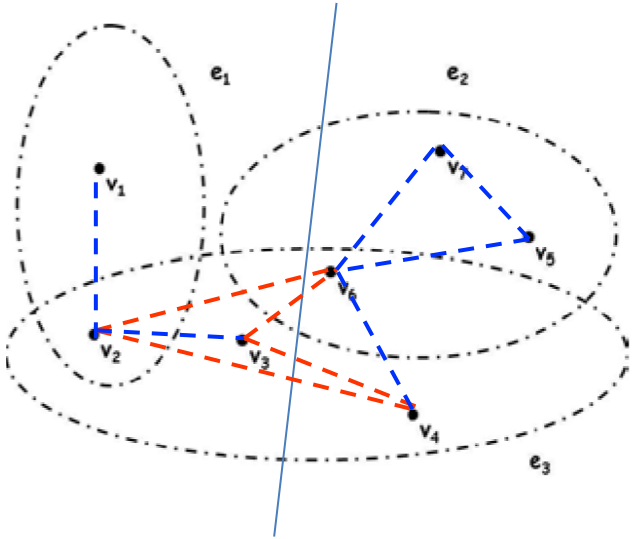


Appendix

Generalize spectral clustering to hypergraphs

Random walk explanation

- Maximize the probability with which the random walk stays in the same cluster
- Minimize the probability with which the random walk crosses different clusters



Generalize spectral clustering to hypergraphs

Spectral hypergraph embedding

- Make the embedding vectors of two nodes belonging to the same edge similar
- Use eigenvectors of Δ

$$\underset{f\in\mathbb{R}^{|V|}}{\operatorname{argmin}} \frac{1}{2}\sum_{e\in E}\sum_{\{u,v\}\subseteq e}\frac{w(e)}{\delta(e)}\left(\frac{f(u)}{\sqrt{d(u)}}-\frac{f(v)}{\sqrt{d(v)}}\right)^2 = f^T\Delta f$$

$$\text{subject to}\sum_{v\in V}f^2(v)=1,\ \sum_{v\in V}f(v)\sqrt{d(v)}=0.$$

Appendix

Generalize spectral clustering to hypergraphs

Transductive inference

- Try to assign the same label to all vertices contained in the same hyperedge

$$\underset{f \in \mathbb{R}^{|V|}}{\operatorname{argmin}} \{ R_{\mathrm{emp}}(f) + \mu \Omega(f) \}$$

Explicit regularization term!

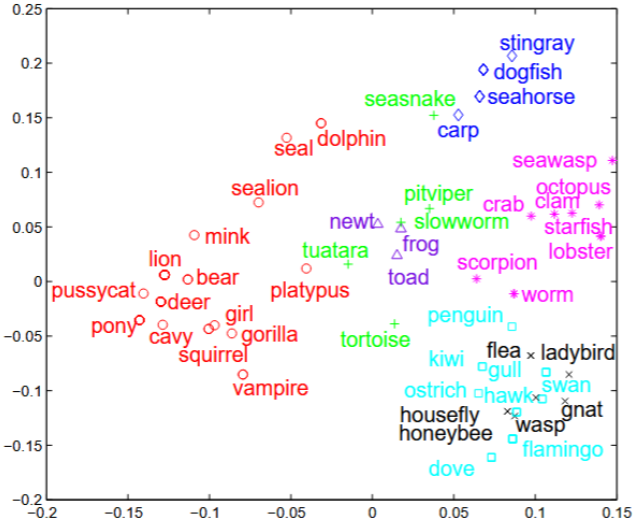
- $R_{emp}(f)$: a chosen empirical loss
- $\Omega(\cdot)$: an objective function from some clustering approach

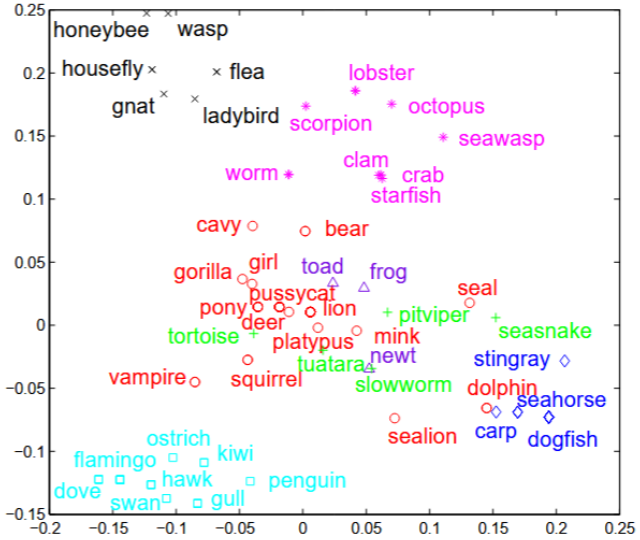
Explicit Laplacian regularization in the objective!

Generalize spectral clustering to hypergraphs

Experiments

- Generalization of the normalized cut approach from simple graphs to hypergraphs

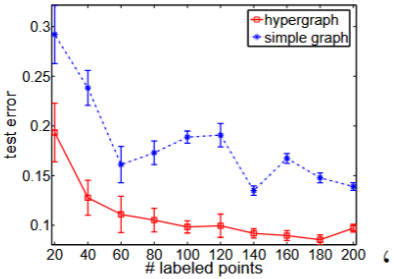




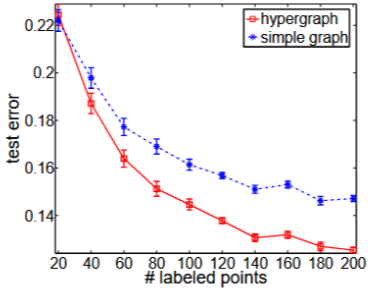
Generalize spectral clustering to hypergraphs

Experiments

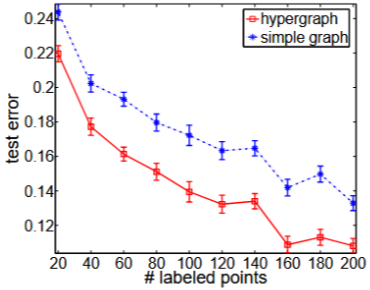
- Classification on complex relational data
  - Show the importance of hypergraphs



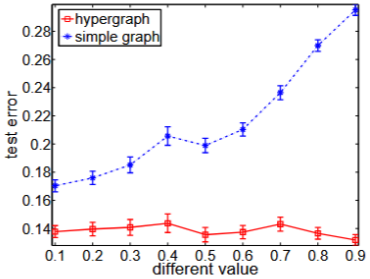
(a) mushroom



(b) 20-newsgroup



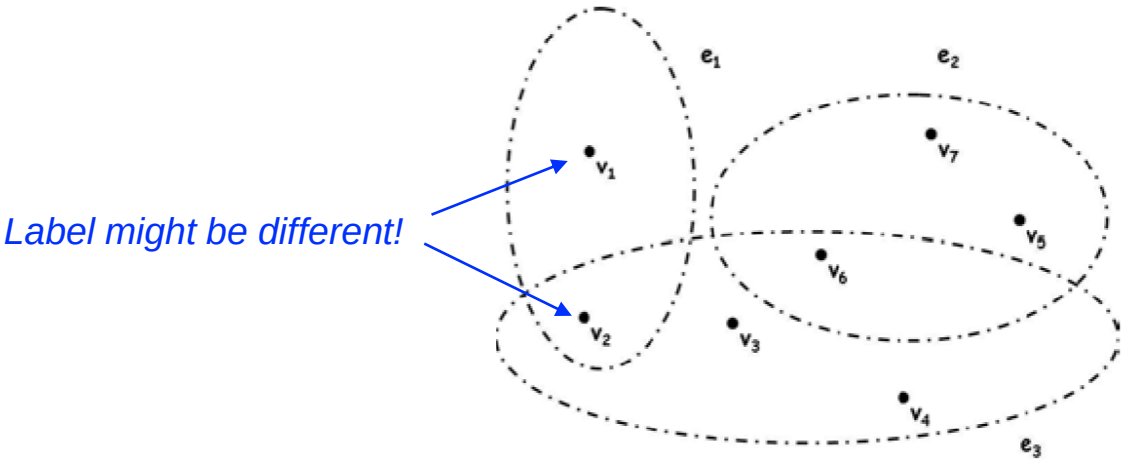
(c) letter



(d) α (letter)

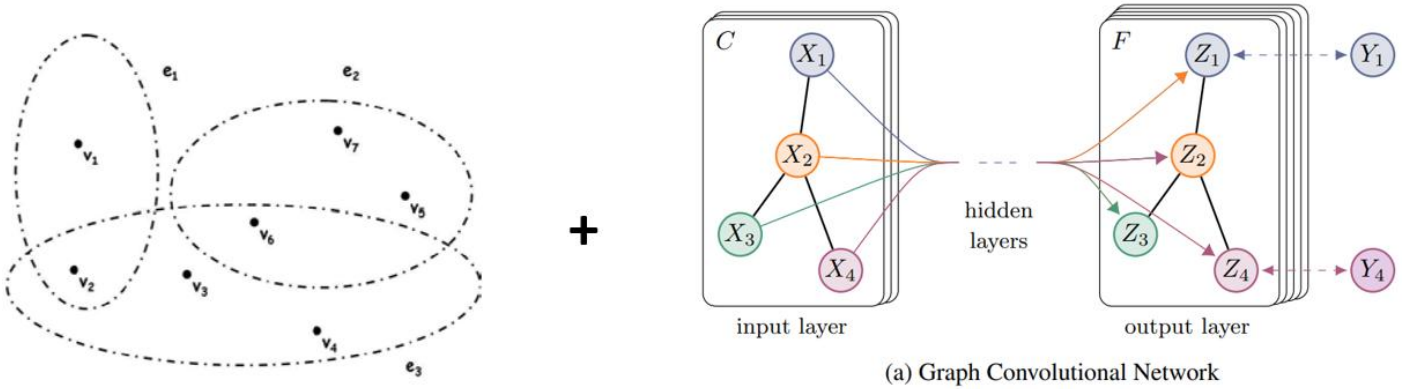
Apply Graph Convolution Network to hypergraphs

- **The problems of explicit Laplacian regularization**
  - Restrict modeling capacity
    - Hyperedges need not necessarily encode node similarity
    - Hyperedges can contain additional information



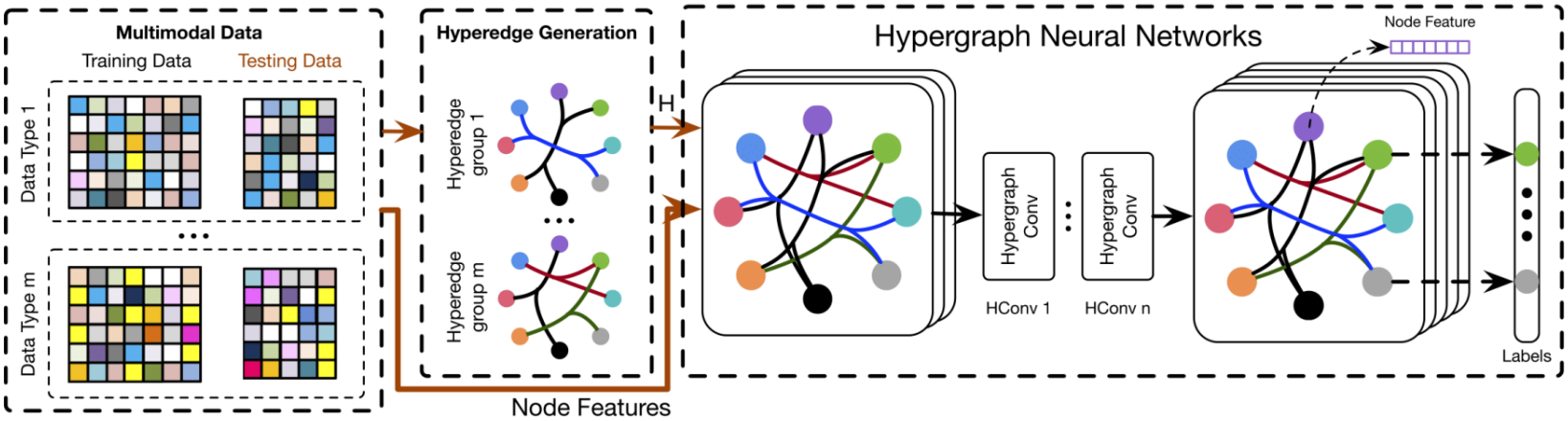
Apply Graph Convolution Network to hypergraphs

- **Implicit Laplacian regularization of GCNs**
  - Encode the hypergraph structure implicitly via a neural network $f(G, X)$



HGNN

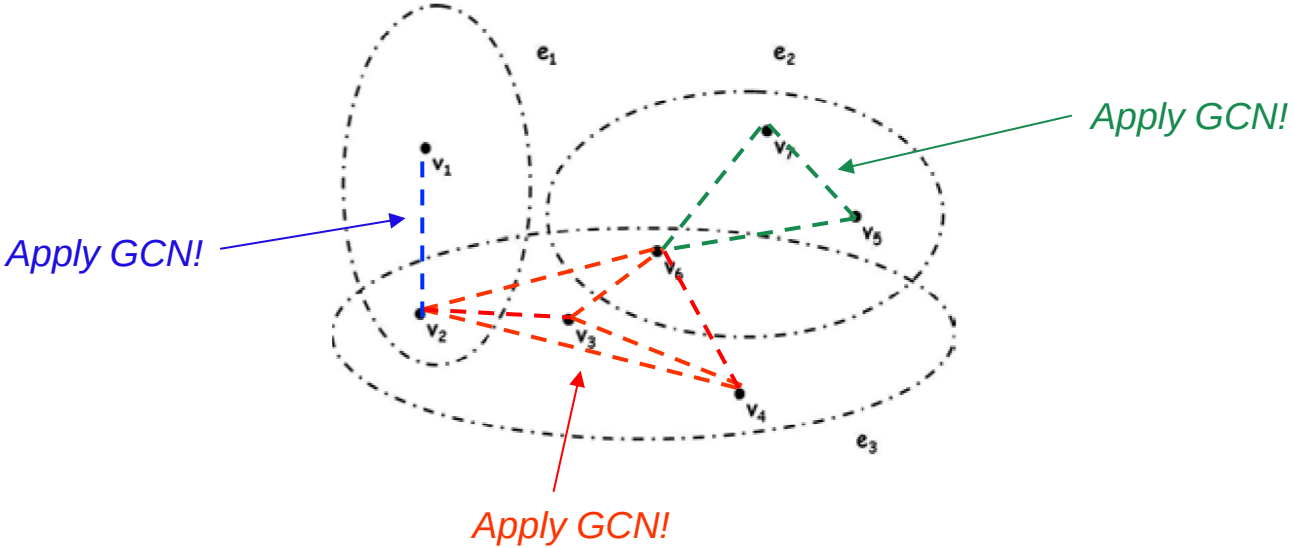
- A hypergraph neural network framework using hyperedge convolution operations



HGNN

Key idea

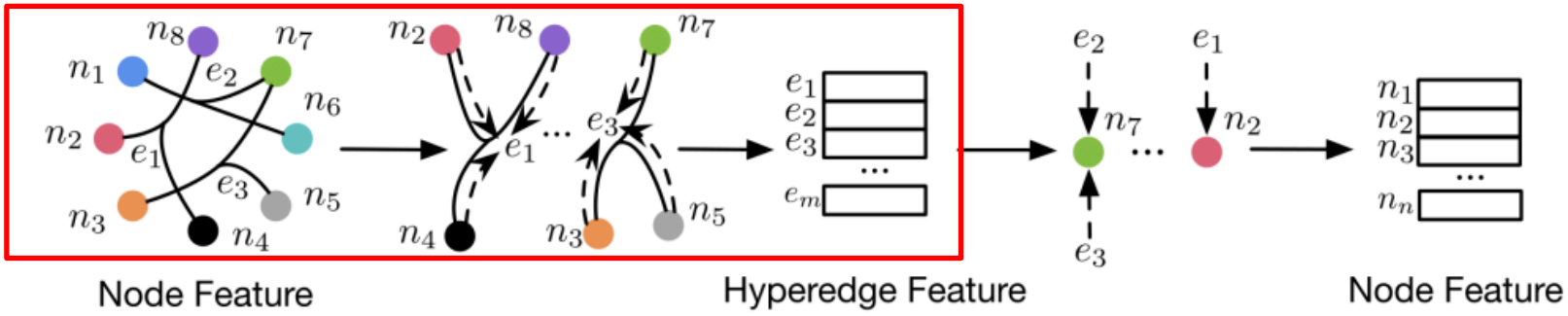
- Apply GCN for clique expansion of each hyperedge



HGNN

Hyperedge convolution operation

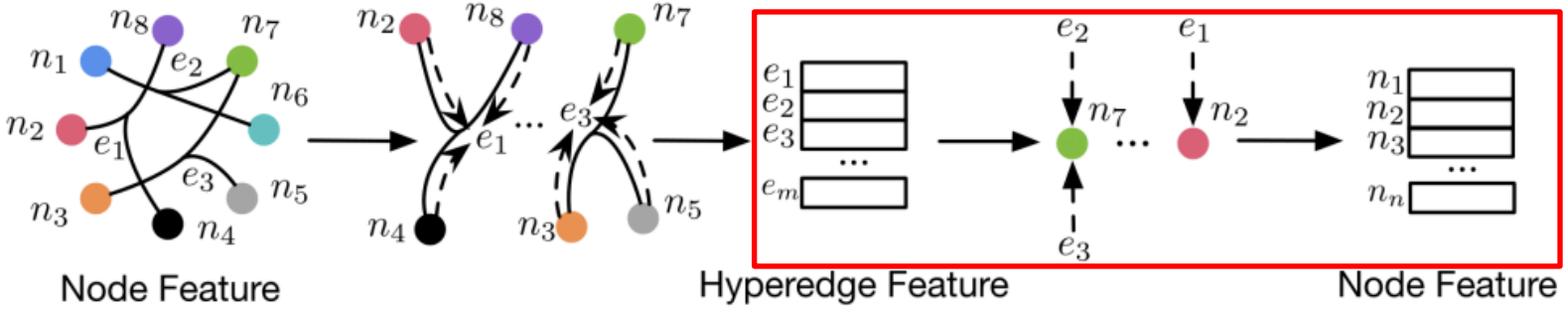
- Define hyperedge features through nodes that it contains



HGNN

Hyperedge convolution operation

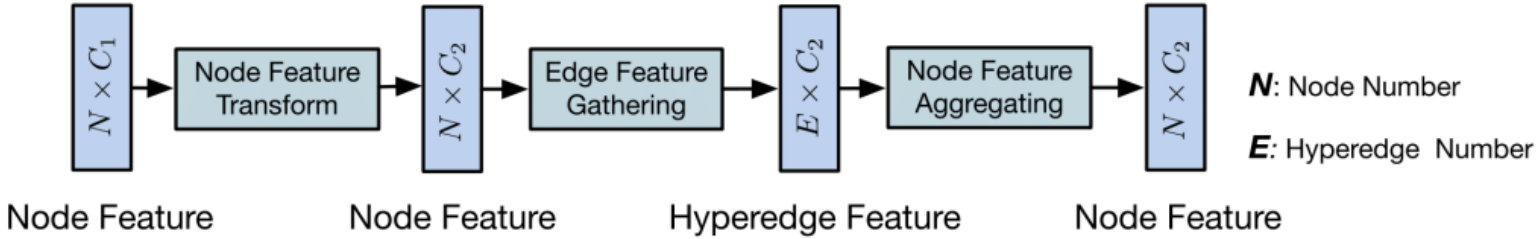
- Define nodes through hyperedges that contain itself



HGNN

Hyperedge convolution operation

- Node-edge-node transform



$$\boldsymbol{X^{(l+1)}} = \boldsymbol{\sigma}(D_v^{-1/2} H W D_e^{-1} D_v^{-1/2} H^T X^{(l)} \Theta^{(l)})$$

HGNN

Experiments

- Compared with GCN, HGNN achieve slight improvement

Dataset	Cora	Pumbed
Nodes	2708	19717
Edges	5429	44338
Feature	1433	500
Training node	140	60
Validation node	500	500
Testing node	1000	1000
Classes	7	3

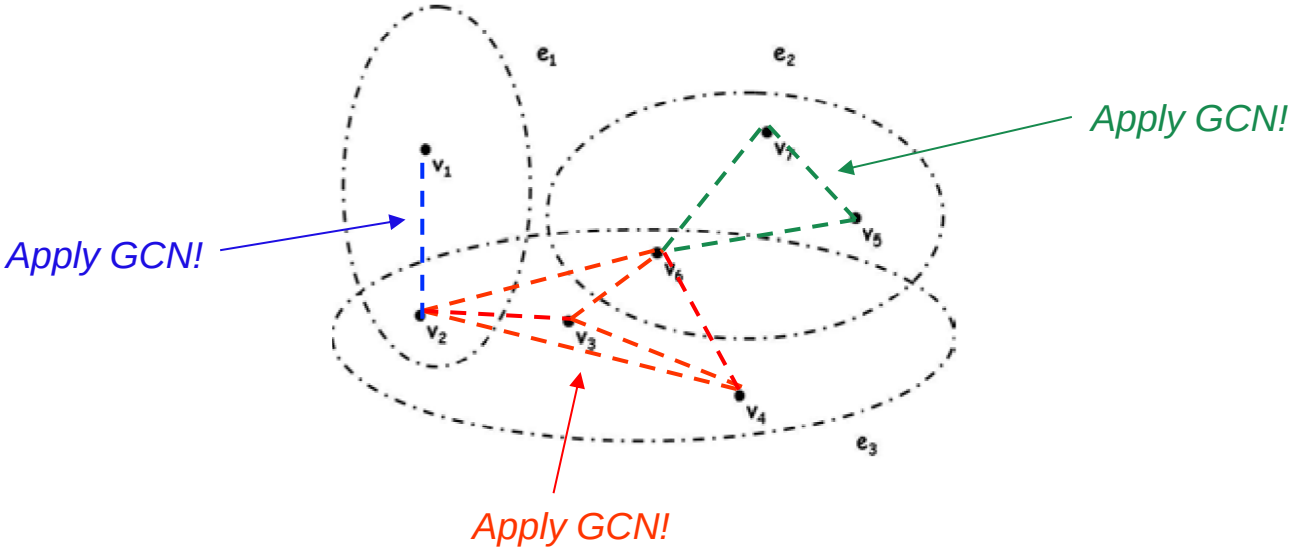
Method	Cora	Pubmed
DeepWalk (Perozzi, Al-Rfou, and Skiena 2014)	67.2%	65.3%
ICA (Lu and Getoor 2003)	75.1%	73.9%
Planetoid (Yang, Cohen, and Salakhutdinov 2016)	75.7%	77.2%
Chebyshev (Defferrard, Bresson, and Vandergheynst 2016)	81.2%	74.4%
GCN (Kipf and Welling 2017)	81.5%	79.0%
HGNN	**81.6%**	**80.1%**

HyperGCN

Motivation

- Clique-based GCNs have too much information loss!

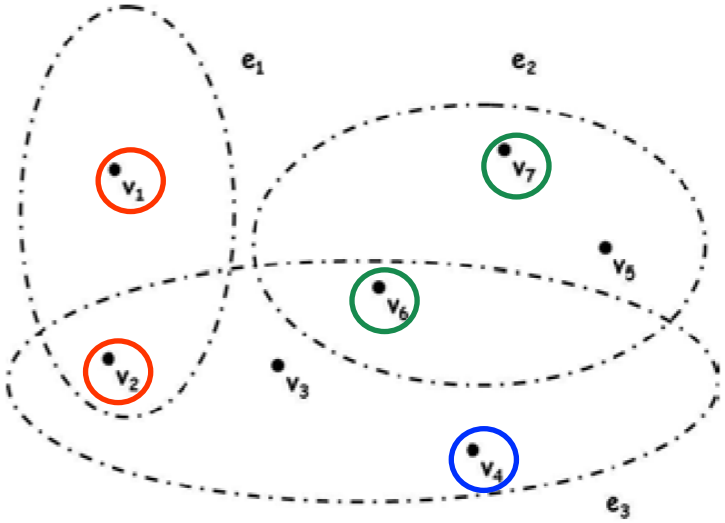
  → *Need to define a Laplacian for hypergraphs*



HyperGCN

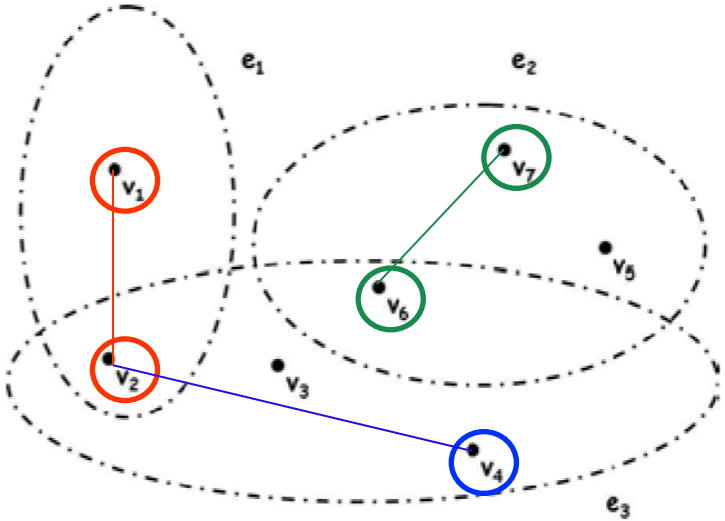
Define a Laplacian for hypergraphs

- 1. For each hyperedge $e \in E$, let $(i_e, j_e) := argmax_{i,j \in e} |S_i - S_j|$, breaking ties randomly



HyperGCN

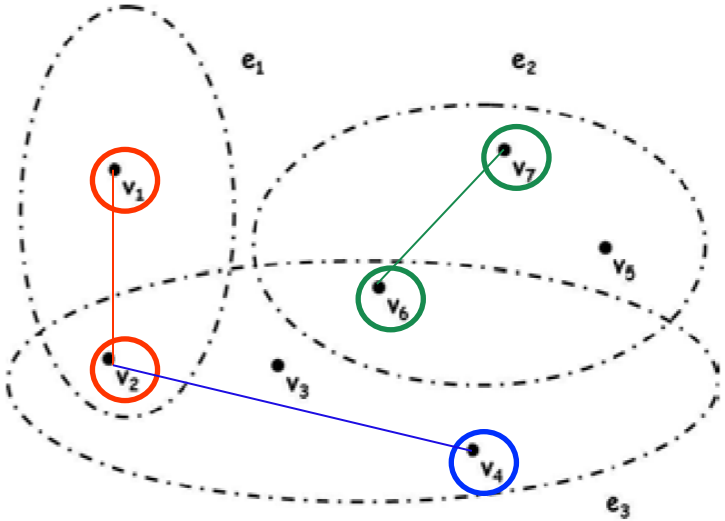
- **Define a Laplacian for hypergraphs**
  - 2. Construct a weighted graph G_S based on (i_e, j_e)



HyperGCN

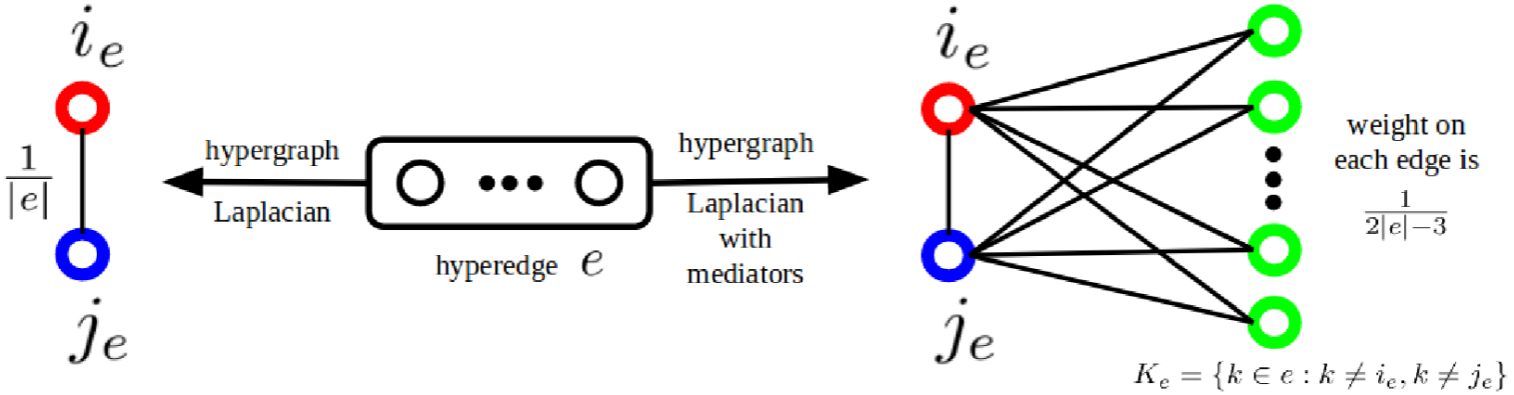
Define a Laplacian for hypergraphs

- 3. Define the symmetrically normalized hypergraph Laplacian
  - $L(S) := (I - D^{-\frac{1}{2}} A_S D^{-\frac{1}{2}})S$



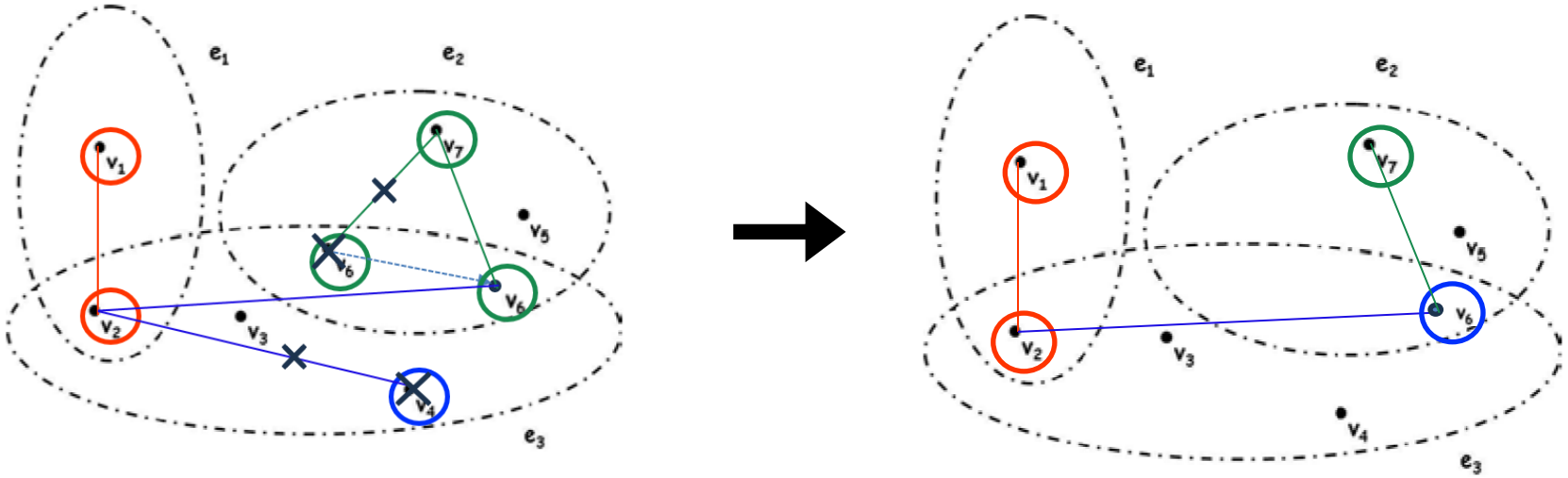
HyperGCN

- **Apply K_e as a mediators**
  - Reflect the hypernodes in $K_e := \{k \in e : k \neq i_e, k \neq j_e\}$ as mediators



HyperGCN

- **Perform GCN over simple graph G_S**
  - The adjacency matrix can be re-estimated
  - Apply the operation in each epoch τ of training until convergence

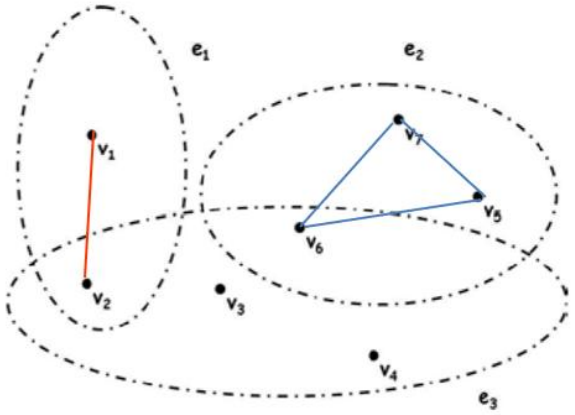


HyperGCN

Experiments

- Results of SSL experiments
  - Red : The mean hyperedge sizes are close to 3
  - Blue : Large noisy hyperedges in DBLP, Pubmed, Cora

Data	Method	DBLP co-authorship	Pubmed co-citation	Cora co-authorship	Cora co-citation	Citeseer co-citation
$\mathcal{H}$	CI	54.81 ± 0.9	52.96 ± 0.8	55.45 ± 0.6	64.40 ± 0.8	70.37 ± 0.3
$\mathbf{X}$	MLP	37.77 ± 2.0	30.70 ± 1.6	41.25 ± 1.9	42.14 ± 1.8	41.12 ± 1.7
$\mathcal{H}, \mathbf{X}$	MLP + HLR	30.42 ± 2.1	30.18 ± 1.5	34.87 ± 1.8	36.98 ± 1.8	37.75 ± 1.6
$\mathcal{H}, \mathbf{X}$	HGNN	25.65 ± 2.1	29.41 ± 1.5	31.90 ± 1.9	$\mathbf{32.41 \pm 1.8}$	$\mathbf{37.40 \pm 1.6}$
$\mathcal{H}, \mathbf{X}$	1-HyperGCN	33.87 ± 2.4	30.08 ± 1.5	36.22 ± 2.2	34.45 ± 2.1	38.87 ± 1.9
$\mathcal{H}, \mathbf{X}$	FastHyperGCN	27.34 ± 2.1	29.48 ± 1.6	32.54 ± 1.8	$\mathbf{32.43 \pm 1.8}$	$\mathbf{37.42 \pm 1.7}$
$\mathcal{H}, \mathbf{X}$	HyperGCN	$\mathbf{24.09 \pm 2.0}$	$\mathbf{25.56 \pm 1.6}$	$\mathbf{30.08 \pm 1.8}$	$\mathbf{32.37 \pm 1.7}$	$\mathbf{37.35 \pm 1.6}$



HyperGCN

Experiments

- Synthetic hypergraphs
  - HyperGCN, FastHyperGCN should be preferred to HGNN for hypergraphs with large noisy hyperedges

Method	$\eta = 0.75$	$\eta = 0.70$	$\eta = 0.65$	$\eta = 0.60$	$\eta = 0.55$	$\eta = 0.50$	sDBLP
HGNN	$\mathbf{15.92 \pm 2.4}$	$\mathbf{24.89 \pm 2.2}$	$\mathbf{31.32 \pm 1.9}$	39.13 ± 1.78	42.23 ± 1.9	44.25 ± 1.8	45.27 ± 2.4
FastHyperGCN	28.86 ± 2.6	31.56 ± 2.7	33.78 ± 2.1	33.89 ± 2.0	$\mathbf{34.56 \pm 2.2}$	$\mathbf{35.65 \pm 2.1}$	41.79 ± 2.8
HyperGCN	22.44 ± 2.0	29.33 ± 2.2	33.41 ± 1.9	$\mathbf{33.67 \pm 1.9}$	35.05 ± 2.0	37.89 ± 1.9	$\mathbf{41.64 \pm 2.6}$

HNHN

Motivation

- Prior methods focus on vertices, viewing hyperedges as connectors
  - *Used only to define a vertex*

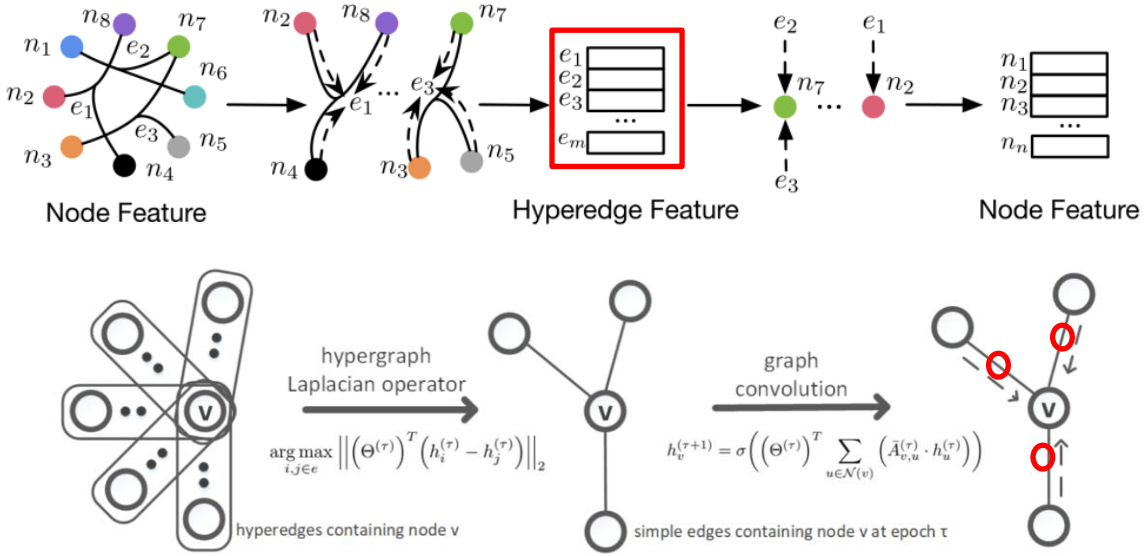
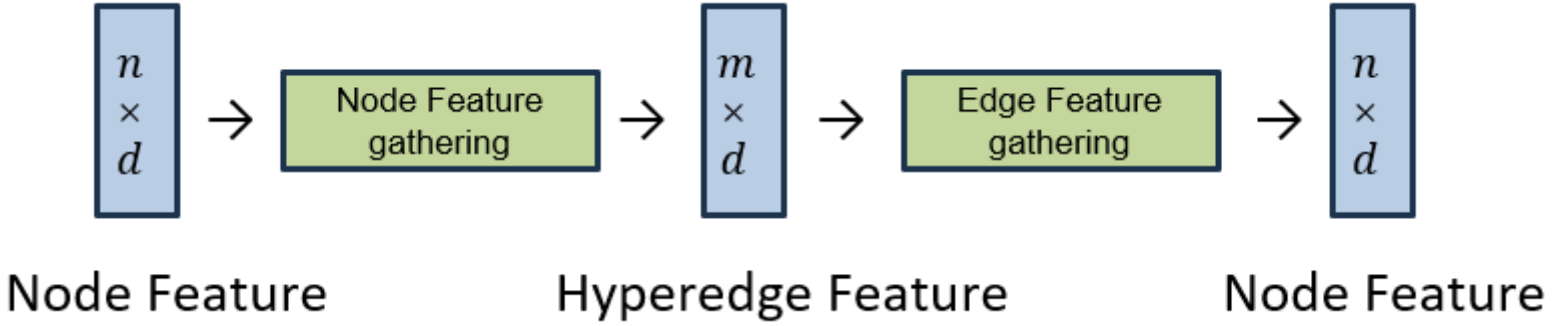


Figure 1: Graph convolution on a hypernode v using HyperGCN.

HNHN

Goal

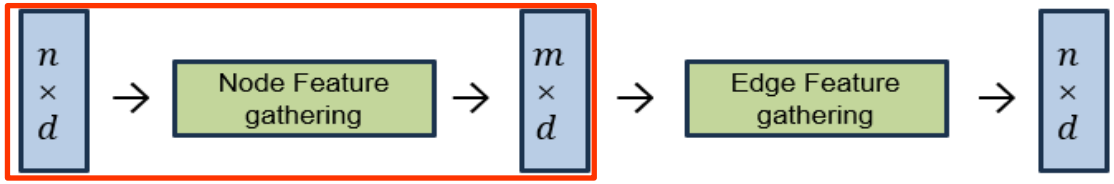
- Propose novel hypergraph convolution algorithm
  - Apply nonlinear activation functions to both hypernodes and hyperedges
  - Apply different weights to hypernodes and hyperedges
  - A more flexible approach to normalization

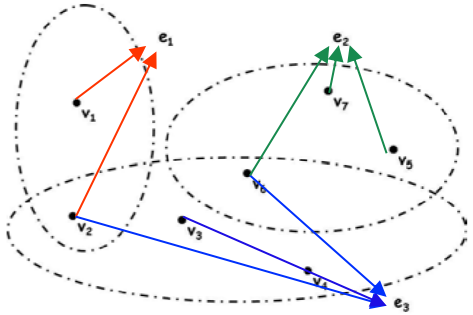


HNHN

Model architecture

- Define hyperedge features through nodes that it contains
- $X'_E = \sigma(D^{-1}_{E,l,\beta} A D_{V,r,\beta} X_V W_E + b_E)$

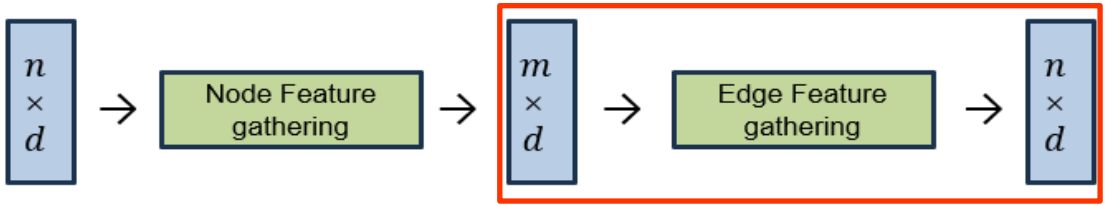


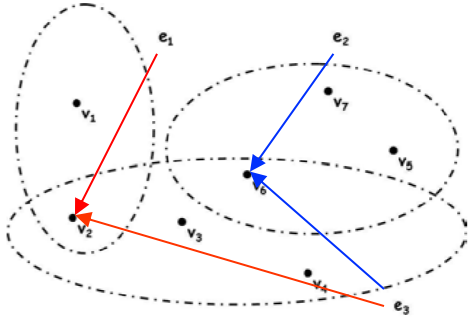


HNHN

Model architecture

- Define nodes through hyperedges that contain itself
- $X'_V = \sigma(D_{V,l,\alpha}^{-1} A D_{E,r,\alpha} X'_E W_V + b_V)$

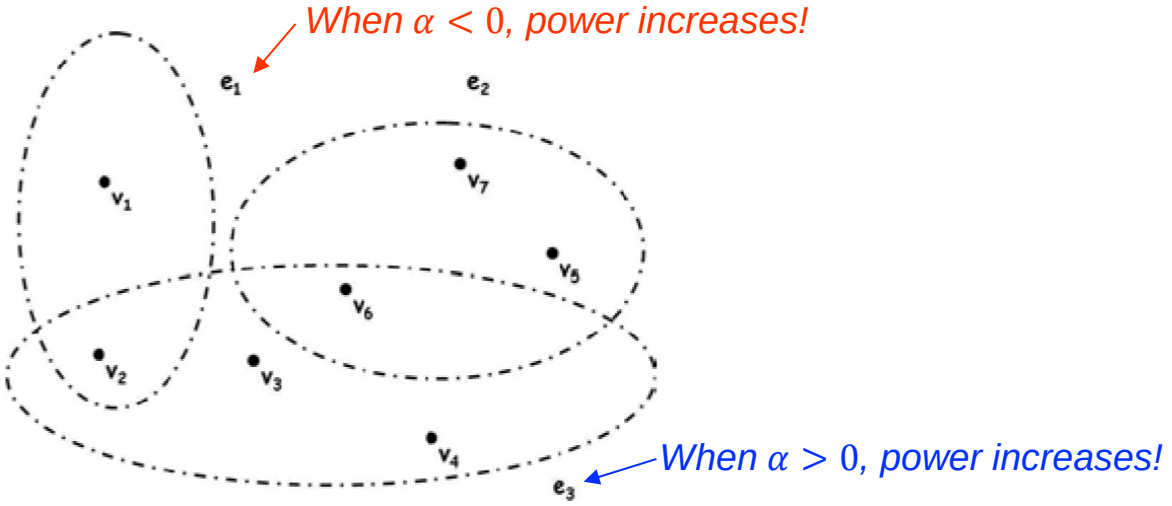




HNHN

Hypergraph normalization

- Weight the contribution of each hyperedge by a power of its degree, depending on a hyperparameter α
  - $\alpha > 0$: the contributions of hyperedges with large degrees are increased
  - $\alpha < 0$: their contributions are decreased

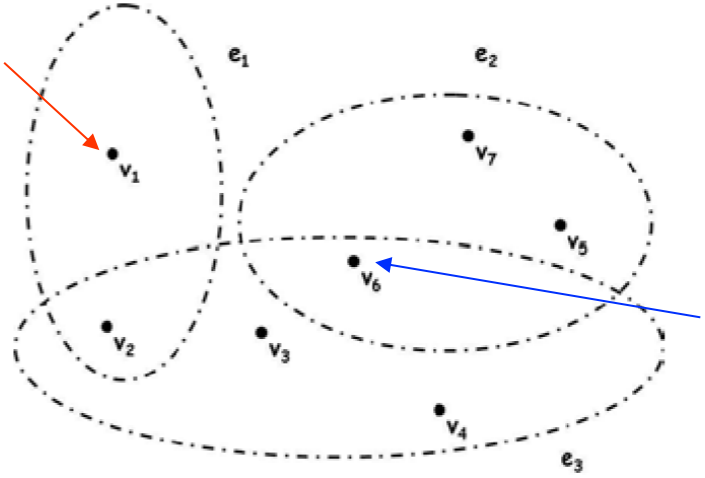


HNHN

Hypergraph normalization

- Weight the contribution of each hypernode by a power of its degree, depending on a hyperparameter β
  - $\beta > 0$: the contributions of hypernodes with large degrees are increased
  - $\beta < 0$: their contributions are decreased

When $\beta < 0$, power increases!



When $\beta > 0$, power increases!

HNHN

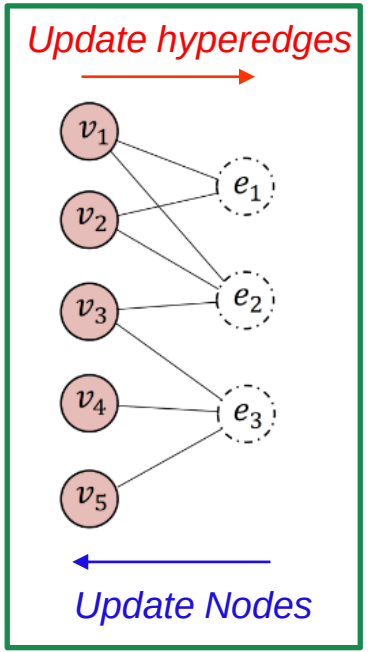
☐ Algorithm for node prediction

Repeat(n_layers) → get representations

Input: Hypergraph incidence matrix $A \in \mathbb{Z}^{n \times m}$, hypernode representations X_V
Input: Set of target labels $\{y\}_{k \in L}$
Compute: $D_{E,l,\alpha}$, $D_{V,r,\alpha}$, $D_{V,l,\alpha}$, and $D_{E,r,\alpha}$ as in § 2.3
for $i = 1$ **to** *n_epochs* **do**
  Initialize $X_E \leftarrow \tilde{0}$. Project X_V to hidden dimension
  for $j = 1$ **to** *n_layers* **do**
    Normalize hypernodes: $\widetilde{X_V} = D_{E,l,\beta}^{-1} A D_{V,r,\beta} X_V$
    Update hyperedges: $X_E = \sigma(\widetilde{X_V} W_{E,j} + b_{E,j})$
    Normalize hyperedges: $\widetilde{X_E} = D_{V,l,\alpha}^{-1} A D_{E,r,\alpha} X_E$
    Update hypernodes: $X_V = \sigma(\widetilde{X_E} W_{V,j} + b_{V,j})$
  Compute cross-entropy **loss** between $\{y\}_{k \in L}$ and predictions using $\{X_V\}_{k \in L}$
  Backpropagate on loss and **optimize** parameters e.g. $\{W_E\}_j$, $\{W_V\}_j$ $\{b_E\}_j$, $\{b_V\}_j$
Return: Learned representations X_V, X_E for prediction tasks

Update parameters

Repeat(n_epochs) → get Learned representations



HNHN

Experiments

- Hypernode prediction
  - Outperform other hypergraph convolution networks

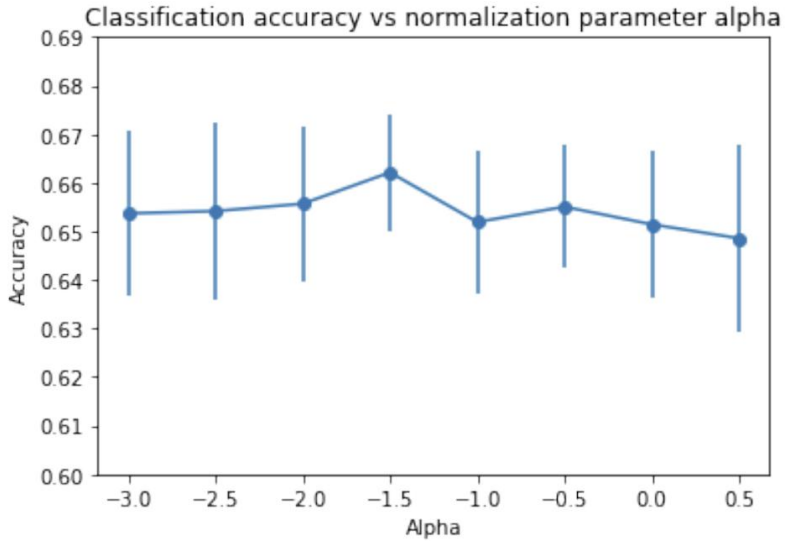
	Accuracy				Timing			
	DBLP	Cora	CiteSeer	PubMed	DBLP	Cora	CiteSeer	PubMed
HyperGCN	71.3±1.2	55.0±.9	54.7±9.8	60.0±10.7	563.4±27.8	183.4±2.7	15.6±.2	171.1±2.8
* Fast	70.5±14.3	45.2±12.9	56.1±11.2	54.4±10.0	**11.5±.1**	**2.9±.1**	**1.1±0.**	**2.5±.1**
HGNN	77.6±.4	58.2±.3	61.1±2.2	63.3±2.2	802.9±59.2	298.4±12.2	30.5±.8	270.1±10.5
HNHN	**85.1±.2**	**63.9±.8**	**64.8±1.6**	**75.9±1.5**	44.2±1.3	13.6±5.4	1.3±.1	26.6±.4

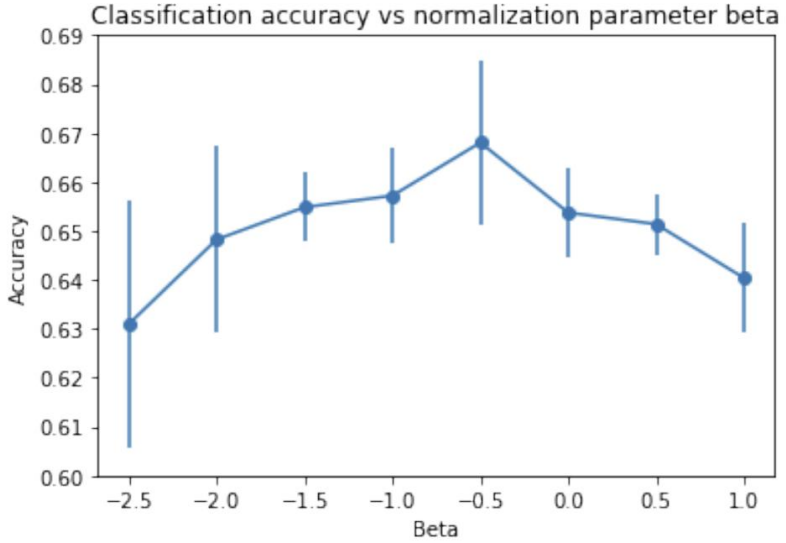
Table 2: Hypernode classification accuracy and timing results. Accuracies are in %, timings are measured in seconds. * Fast stands for HyperGCN Fast.

HNHN

Experiments

- Effects of normalization parameters
  - Commonly used normalization corresponding to $\alpha = \beta = 0$ is not necessarily optimal

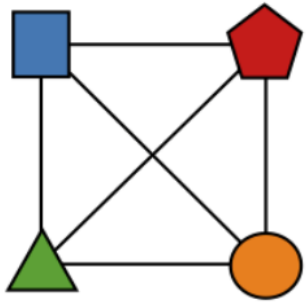


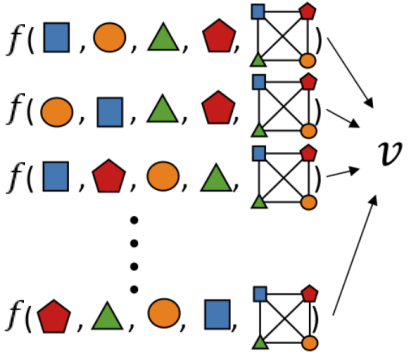


AllSet

Motivation

- A composition of two multiset functions can approximate all propagation rules
  - *A function f is permutation invariant*
    - $f(x_{\pi(1)}, \dots, x_{\pi(n)}) = f(x_1, \dots, x_n)$ *iff* $\forall \pi \in S_n$, *the symmetric group of order* $n!$
  - *A function f is a multiset function if it is permutation invariant*



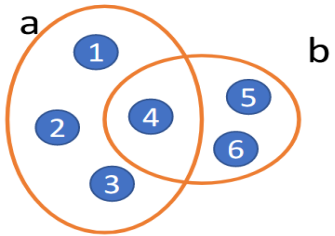


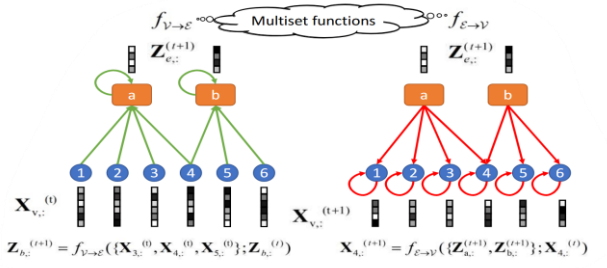
AllSet

Update rules

- $\mathbf{Z}_{e,:}^{(t+1)} = f_{\mathcal{V}\to\mathcal{E}}\left(V_{e,\mathbf{X}^{(t)}}; \mathbf{Z}_{e,:}^{(t)}\right), \mathbf{X}_{v,:}^{(t+1)} = f_{\mathcal{E}\to\mathcal{V}}(E_{v,\mathbf{Z}^{(t+1)}}; \mathbf{X}_{v,:}^{(t)})$
  - $V_{e,\mathbf{X}}$: the multiset of hidden node representations contained in the hyperedge e
  - $E_{v,\mathbf{Z}}$: the multiset of hidden representations of hyperedges that contain the node v

- Make the tacit assumption
  - Both functions $f_{\mathcal{V}\to\mathcal{E}}$ and $f_{\mathcal{E}\to\mathcal{V}}$ also include the hypergraph $\mathcal{G}$ as an input
    → Allow degree normalization to be part of framework

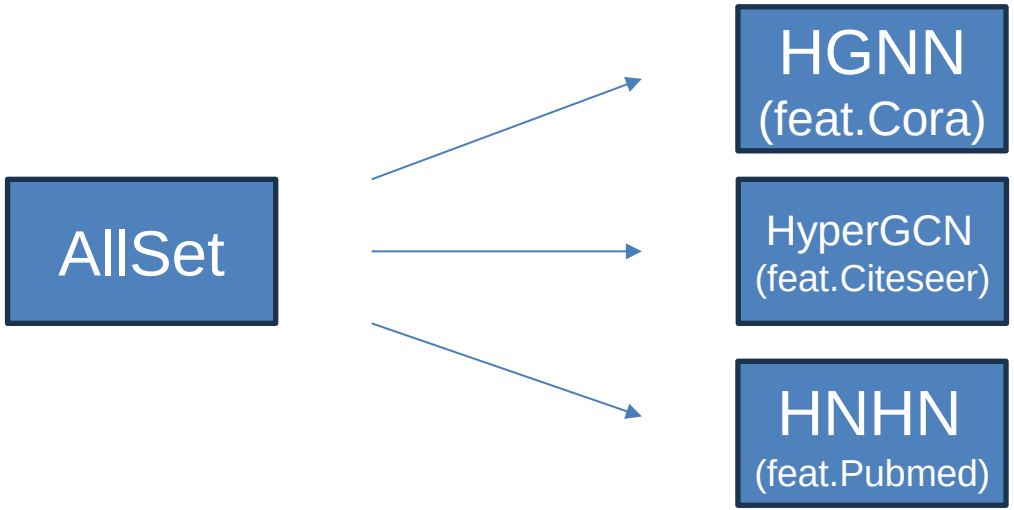




AllSet

Key idea behind AllSet

- To learn the multiset functions $f_{\mathcal{V}\to\mathcal{E}}$ and $f_{\mathcal{E}\to\mathcal{V}}$ on the fly for each dataset and task
  - We should properly parametrize the multiset functions
  - The parametrization should represent a universal approximator for a multiset function



AllSet

AllDeepSets

- Any multiset functions f can be parametrized
  - $f(S) = \rho(\sum_{s \in S} \phi(s))$, where ρ and ϕ are some bijective mappings
  - These mappings can be replaced by any universal approximator such as a MLP

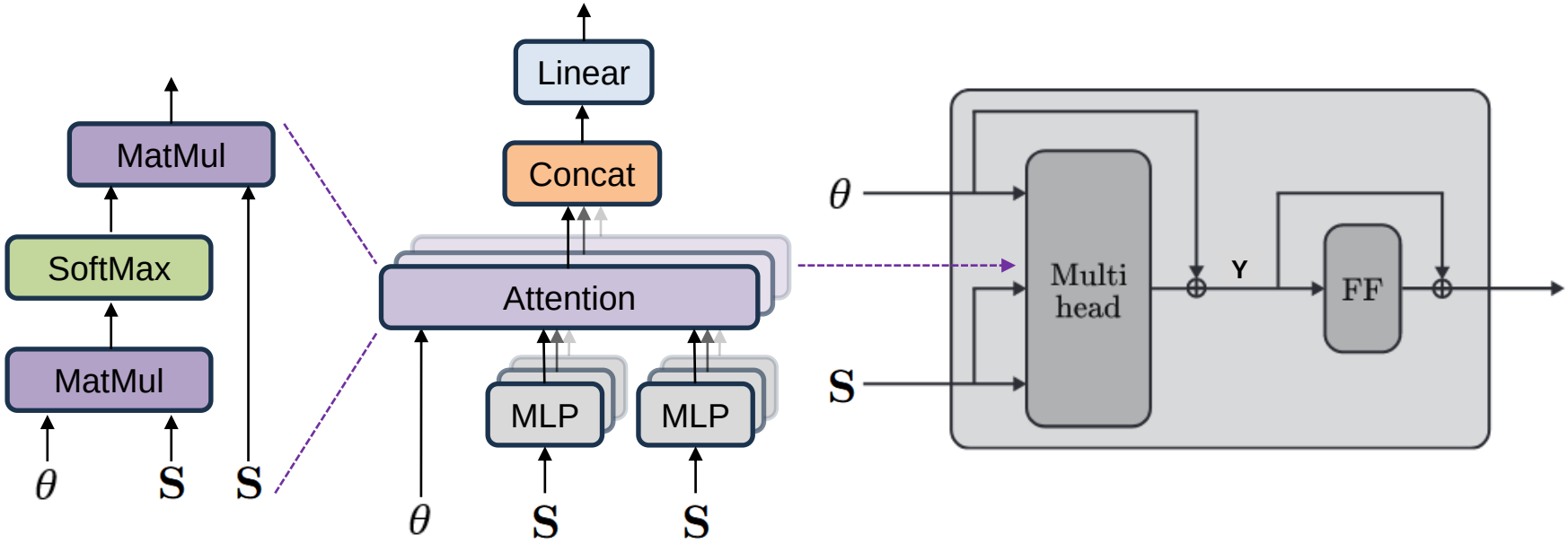
$$\text{AllDeepSets: } f_{\mathcal{V} \to \mathcal{E}}(S) = f_{\mathcal{E} \to \mathcal{V}}(S) = \boxed{\text{MLP}\left(\sum_{s \in S} \text{MLP}(s)\right)}$$

Hard to learn the importance of each individual contributing term!

AllSet

AllSetTransformer

- Attention-based AllSet layer for hypergraph neural networks
- Reflect similarity between two vertices in terms of θ



AllSet

Experiments

- Accuracy as the evaluation metric
  - Show the best overall performance compared to SOTA

	Cora	Citeseer	Pubmed	Cora-CA	DBLP-CA	Zoo	20Newsgroups	mushroom
AllSetTransformer	78.59 ± 1.47	73.08 ± 1.20	88.72 ± 0.37	83.63 ± 1.47	91.53 ± 0.23	**97.50 ± 3.59**	81.38 ± 0.58	**100.00 ± 0.00**
AllDeepSets	76.88 ± 1.80	70.83 ± 1.63	**88.75 ± 0.33**	81.97 ± 1.50	91.27 ± 0.27	95.39 ± 4.77	81.06 ± 0.54	99.99 ± 0.02
MLP	75.17 ± 1.21	72.67 ± 1.56	87.47 ± 0.51	74.31 ± 1.89	84.83 ± 0.22	87.18 ± 4.44	**81.42 ± 0.49**	**100.00 ± 0.00**
CECGN	76.17 ± 1.39	70.16 ± 1.31	86.45 ± 0.43	77.05 ± 1.26	88.00 ± 0.26	51.54 ± 11.19	OOM	95.27 ± 0.47
CEGAT	76.41 ± 1.53	70.63 ± 1.30	86.81 ± 0.42	76.16 ± 1.19	88.59 ± 0.29	47.88 ± 14.03	OOM	96.60 ± 1.67
HNHN	76.36 ± 1.92	72.64 ± 1.57	86.90 ± 0.30	77.19 ± 1.49	86.78 ± 0.29	93.59 ± 5.88	81.35 ± 0.61	**100.00 ± 0.01**
HGNN	79.39 ± 1.36	72.45 ± 1.16	86.44 ± 0.44	82.64 ± 1.65	91.03 ± 0.20	92.50 ± 4.58	80.33 ± 0.42	98.73 ± 0.32
HCHA	79.14 ± 1.02	72.42 ± 1.42	86.41 ± 0.36	82.55 ± 0.97	90.92 ± 0.22	93.65 ± 6.15	80.33 ± 0.80	98.70 ± 0.39
HyperGCN	78.45 ± 1.26	71.28 ± 0.82	82.84 ± 8.67	79.48 ± 2.08	89.38 ± 0.25	N/A	81.05 ± 0.59	47.90 ± 1.04
UniGCNII	78.81 ± 1.05	73.05 ± 2.21	88.25 ± 0.40	83.60 ± 1.14	**91.69 ± 0.19**	93.65 ± 4.37	81.12 ± 0.67	99.96 ± 0.05
HAN (full batch)*	**80.18 ± 1.15**	74.05 ± 1.43	86.21 ± 0.48	**84.04 ± 1.02**	90.89 ± 0.23	85.19 ± 8.18	OOM	90.86 ± 2.40
HAN (mini batch)*	79.70 ± 1.77	**74.12 ± 1.52**	85.32 ± 2.25	81.71 ± 1.73	90.17 ± 0.65	75.77 ± 7.10	79.72 ± 0.62	93.45 ± 1.31

	NTU2012	ModelNet40	Yelp	House(1)	Walmart(1)	House(0.6)	Walmart(0.6)	avg. ranking (↑)
AllSetTransformer	88.69 ± 1.24	**98.20 ± 0.20**	**36.89 ± 0.51**	69.33 ± 2.20	**65.46 ± 0.25**	83.14 ± 1.92	**78.46 ± 0.40**	**2.00**
AllDeepSets	88.09 ± 1.52	96.98 ± 0.26	30.36 ± 1.57	67.82 ± 2.40	64.55 ± 0.33	80.70 ± 1.59	**78.46 ± 0.26**	4.47
MLP	85.52 ± 1.49	96.14 ± 0.36	31.96 ± 0.44	67.93 ± 2.33	45.51 ± 0.24	81.53 ± 2.26	63.28 ± 0.37	6.27
CECGN	81.52 ± 1.43	89.92 ± 0.46	OOM	62.80 ± 2.61	54.44 ± 0.24	64.36 ± 2.41	59.78 ± 0.32	9.66
CEGAT	82.21 ± 1.23	92.52 ± 0.39	OOM	69.09 ± 3.00	51.14 ± 0.56	77.25 ± 2.53	59.47 ± 1.05	8.80
HNHN	89.11 ± 1.44	97.84 ± 0.25	31.65 ± 0.44	67.80 ± 2.59	47.18 ± 0.35	78.78 ± 1.88	65.80 ± 0.39	5.87
HGNN	87.72 ± 1.35	95.44 ± 0.33	33.04 ± 0.62	61.39 ± 2.96	62.00 ± 0.24	66.16 ± 1.80	77.72 ± 0.21	5.73
HCHA	87.48 ± 1.87	94.48 ± 0.28	30.99 ± 0.72	61.36 ± 2.53	62.45 ± 0.26	67.91 ± 2.26	77.12 ± 0.26	6.40
HyperGCN	56.36 ± 4.86	75.89 ± 5.26	29.42 ± 1.54	48.31 ± 2.93	44.74 ± 2.81	78.22 ± 2.46	55.31 ± 0.30	9.87
UniGCNII	**89.30 ± 1.33**	98.07 ± 0.23	31.70 ± 0.52	67.25 ± 2.57	54.45 ± 0.37	80.65 ± 1.96	72.08 ± 0.28	3.87
HAN (full batch)*	83.58 ± 1.46	94.04 ± 0.41	OOM	**71.05 ± 2.26**	OOM	**83.27 ± 1.62**	OOM	6.73
HAN (mini batch)*	80.77 ± 2.36	91.52 ± 0.96	26.05 ± 1.37	62.00 ± 9.06	48.57 ± 1.04	82.04 ± 2.68	63.10 ± 0.96	7.60

AllSet

Experiments

- Average running times over all different choices of hyperparameters

	Cora	Citeseer	Pubmed	Cora-CA	DBLP-CA	Zoo
AllSetTransformer	8.18s ± 3.59s	8.41s ± 4.39s	22.99s ± 16.54s	7.21s ± 2.94s	53.14s ± 43.49s	5.97s ± 2.23s
AllDeepSets	5.61s ± 4.47s	6.83s ± 5.63s	27.05s ± 23.94s	5.62s ± 4.38s	57.26s ± 55.65s	4.14s ± 3.18s
MLP	1.00s ± 0.46s	1.03s ± 0.48s	1.53s ± 0.95s	1.00s ± 0.45s	2.64s ± 1.25s	0.99s ± 0.50s
CEGCN	1.63s ± 0.76s	2.24s ± 1.38s	5.93s ± 5.01s	1.79s ± 0.99s	18.91s ± 17.08s	1.41s ± 0.62s
CEGAT	6.47s ± 3.99s	9.25s ± 7.60s	34.52s ± 35.16s	7.93s ± 5.94s	53.84s ± 37.83s	5.28s ± 5.92s
HNHN	1.65s ± 1.01s	2.43s ± 1.62s	5.03s ± 3.14s	1.89s ± 0.80s	14.80s ± 9.55s	1.59s ± 0.63s
HGNN	3.42s ± 1.49s	4.44s ± 2.03s	8.44s ± 5.08s	3.37s ± 1.47s	21.02s ± 13.61s	2.75s ± 1.06s
HCHA	3.32s ± 1.42s	4.30s ± 2.02s	8.22s ± 4.97s	3.24s ± 1.42s	20.66s ± 13.37s	2.65s ± 1.03s
HyperGCN	2.13s ± 0.75s	2.48s ± 0.76s	3.23s ± 1.00s	2.27s ± 0.67s	6.13s ± 1.93s	1.91s ± 0.58s
UniGCNII	13.13s ± 13.64s	16.81s ± 17.63s	61.81s ± 75.63s	11.61s ± 12.46s	94.97s ± 81.68s	3.56s ± 2.58s
HAN (full batch)	3.47s ± 0.52s	2.58s ± 0.48s	7.17s ± 0.11s	3.09s ± 0.07s	12.39s ± 0.22s	2.78s ± 0.09s
HAN (mini batch)	99.87s ± 17.05s	69.06s ± 11.69s	502.44s ± 161.40s	143.80s ± 15.48s	0.45h ± 0.12h	102.98s ± 2.39s

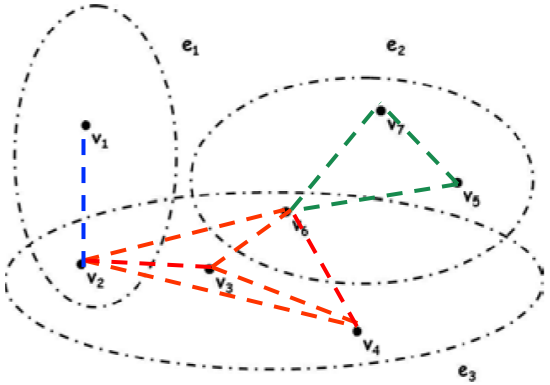
	20Newsgroups	Mushroom	NTU2012	ModelNet40	Yelp	House(1)	Walmart(1)
AllSetTransformer	21.97s ± 17.96s	14.35s ± 11.10s	6.96s ± 2.65s	19.99s ± 14.46s	244.61s ± 97.44s	6.87s ± 3.11s	123.51s ± 102.91s
AllDeepSets	20.88s ± 21.73s	12.76s ± 13.89s	5.15s ± 4.03s	19.90s ± 19.15s	196.53s ± 254.83s	4.52s ± 3.01s	107.54s ± 119.02s
MLP	1.22s ± 0.54s	1.11s ± 0.53s	0.97s ± 0.43s	1.14s ± 0.50s	3.88s ± 1.93s	0.98s ± 0.47s	4.44s ± 2.72s
CEGCN	OOM	67.66s ± 47.17s	1.51s ± 0.69s	3.68s ± 2.72s	OOM	4.16s ± 3.65s	63.42s ± 62.30s
CEGAT	OOM	121.39s ± 7.43s	5.45s ± 2.97s	19.54s ± 19.68s	OOM	21.94s ± 23.44s	85.82s ± 45.10s
HNHN	5.21s ± 3.18s	3.28s ± 1.80s	1.68s ± 0.68s	4.92s ± 2.99s	56.15s ± 77.57s	1.70s ± 0.53s	23.62s ± 18.74s
HGNN	9.69s ± 5.93s	6.51s ± 3.58s	3.30s ± 1.40s	8.86s ± 5.36s	343.16s ± 262.93s	3.29s ± 1.21s	44.19s ± 36.85s
HCHA	9.61s ± 5.98s	6.48s ± 3.57s	3.23s ± 1.37s	8.82s ± 5.47s	135.47s ± 187.18s	3.20s ± 1.18s	43.62s ± 36.37s
HyperGCN	4.34s ± 1.54s	3.56s ± 1.25s	2.29s ± 0.72s	3.41s ± 1.17s	125.67s ± 56.93s	2.39s ± 0.76s	20.11s ± 7.88s
UniGCNII	58.01s ± 66.97s	35.47s ± 44.72s	9.22s ± 9.68s	46.30s ± 54.77s	309.35s ± 129.68s	8.83s ± 7.97s	119.66s ± 73.89s
HAN (full batch)	OOM	30.28s ± 3.29s	3.10s ± 0.11s	4.56s ± 0.21s	OOM	3.32s ± 0.12s	OOM
HAN (mini batch)	313.28s ± 93.42s	111.53s ± 30.05s	165.35s ± 1.96s	414.50s ± 87.25s	5.50h ± 1.48h	72.79s ± 5.59s	1.96h ± 0.44h

Propose a novel expansion

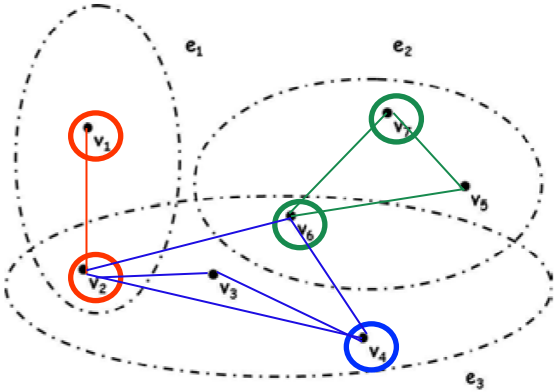
☐ Motivation

- Prior works are based on CE's variant

  → *Inevitably occur information loss!*



HGNN, HNHN



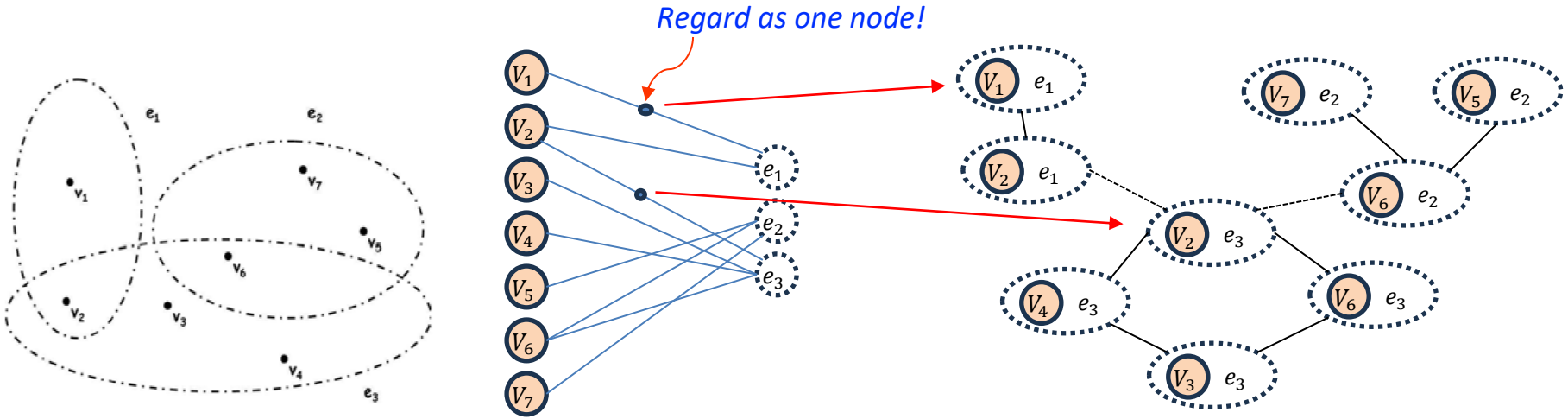
HyperGCN

Line Expansion (LE)

Key idea

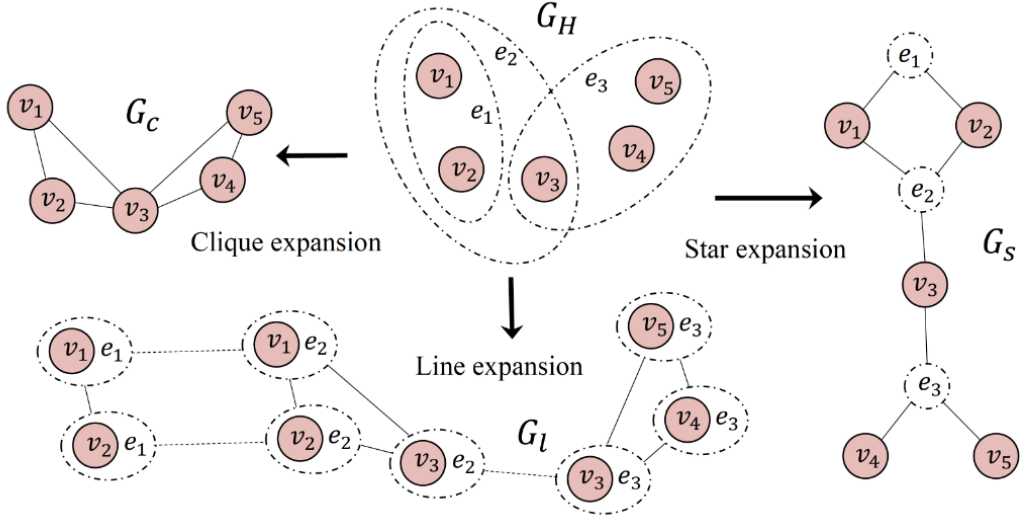
- Hypergraph has a special symmetric structure
  - Vertices are connected to multiple edges
  - Edges are conversely connected to multiple vertices

  → *Treat vertices and edges equally!*



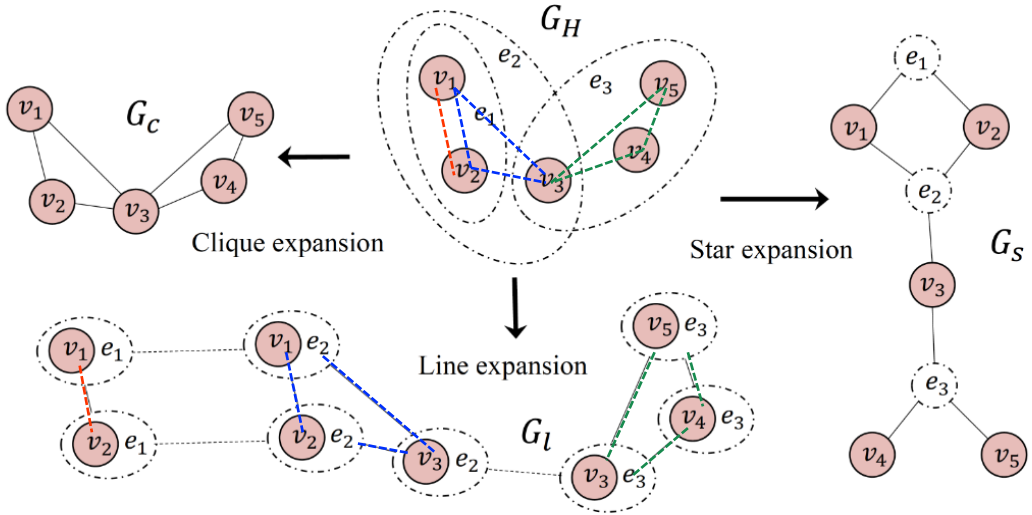
Line Expansion (LE)

- $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$: **Line Expansion of hypergraph** $\mathcal{G}_H = (\mathcal{V}, \mathcal{E})$
  - $\mathcal{V}_l$: vertex-hyperedge pair $\{(v, e) | v \in e, v \in \mathcal{V}, e \in \mathcal{E}\}$
  - Two nodes are "neighborhood" when *they share the same vertex or hyperedge*



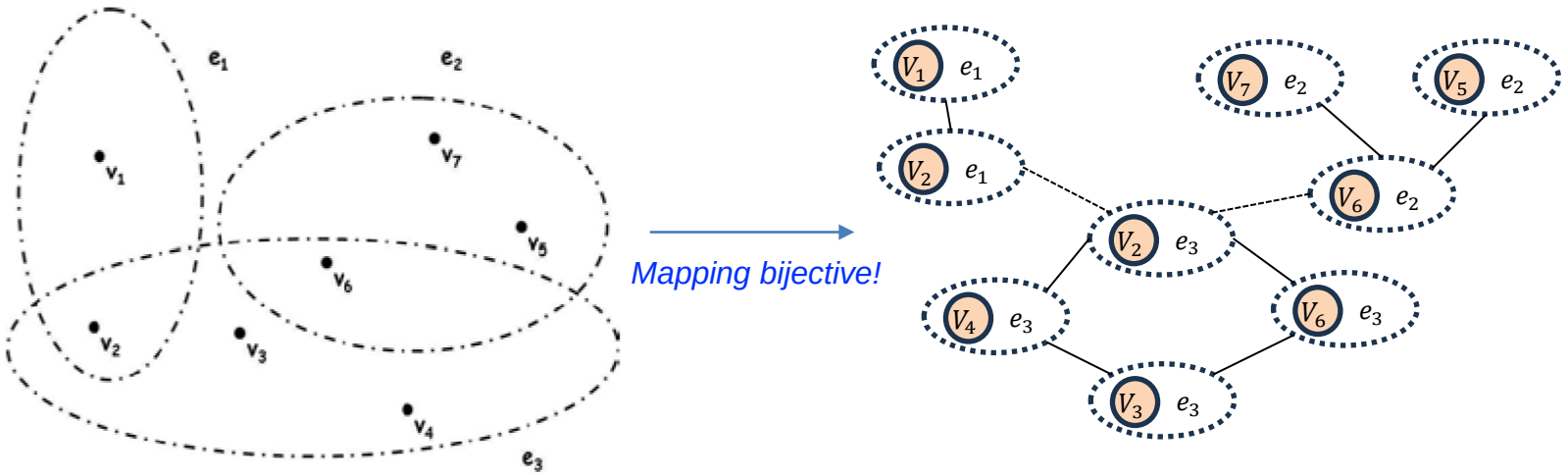
Line Expansion (LE)

- $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$: **Line Expansion of hypergraph** $\mathcal{G}_H = (\mathcal{V}, \mathcal{E})$
  - Unify clique and star expansion
  - View each line node as a vertex with hyperedge context or a hyperedge with vertex context



Line Expansion (LE)

- $\mathcal{G}_l = (\mathcal{V}_l, \mathcal{E}_l)$: **Line Expansion of hypergraph** $\mathcal{G}_H = (\mathcal{V}, \mathcal{E})$
  - Bijective, i.e., can uniquely recover from the line expansion graph to the original hypergraph
    → All higher-order information is preserved during the transformation

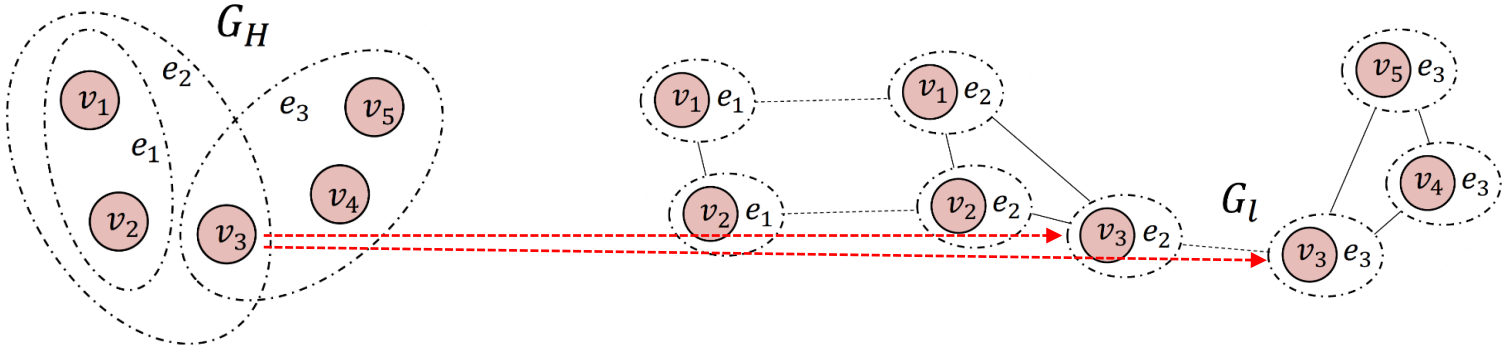


Line Expansion (LE)

☐ Hypergraph Learning with LE

- ■ Step 1 : Feature Projection
  - ☐ Map hypernodes to line nodes using P_{vertex}
  - ☐ Scatter features from vertex of $\mathcal{G}_H$ to feature vectors of line nodes in $\mathcal{G}_l$

$$H^{(0)} = P_{vertex} X \in \mathbb{R}^{|\mathcal{V}_l| \times d_i}$$

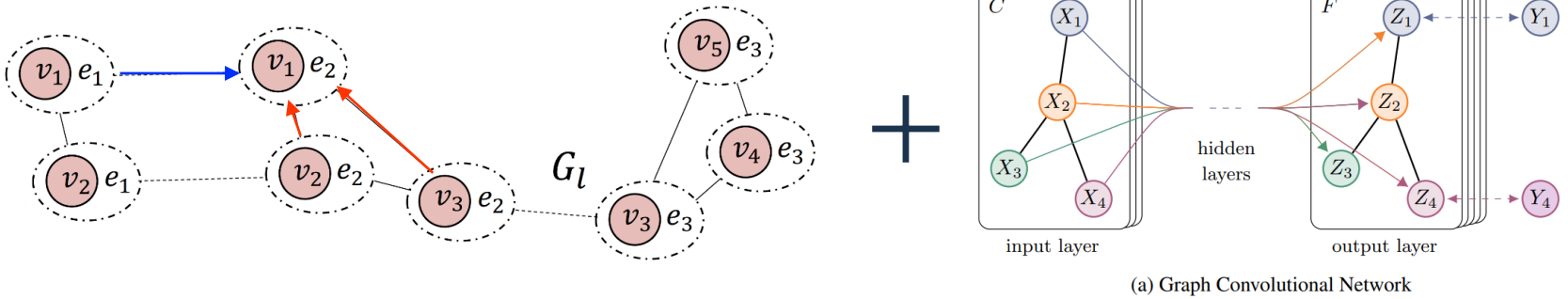


Line Expansion (LE)

Hypergraph Learning with LE

- Step 2 : Convolution Layer
  - Apply neighborhood feature aggregation by graph convolution
    - Convolve information from neighbors who share the same hyperedges (Red) / vertices (Blue)

$$h_{(v,e)}^{(k+1)} = \sigma\Big(\sum_{e'} w_e h_{(v,e')}^{(k)} \Theta^{(k)} + \sum_{v'} w_v h_{(v',e)}^{(k)} \Theta^{(k)}\Big)$$



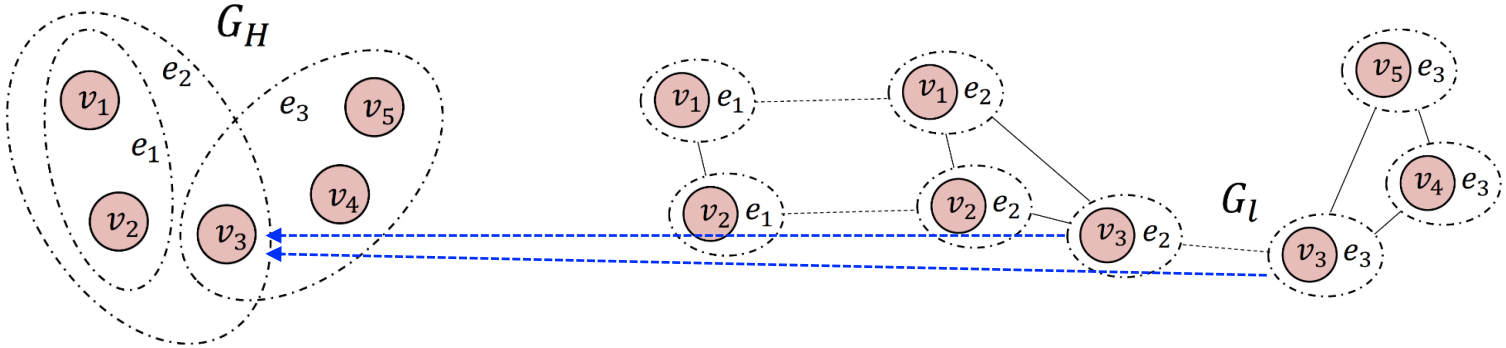
(a) Graph Convolutional Network

Line Expansion (LE)

Hypergraph Learning with LE

- Step 3 : Representation Back-projection
  - Fuse the learned representation using P'_{vertex} in an inverse edge degree manner
  - Vertex labels are predicted on the fused representation

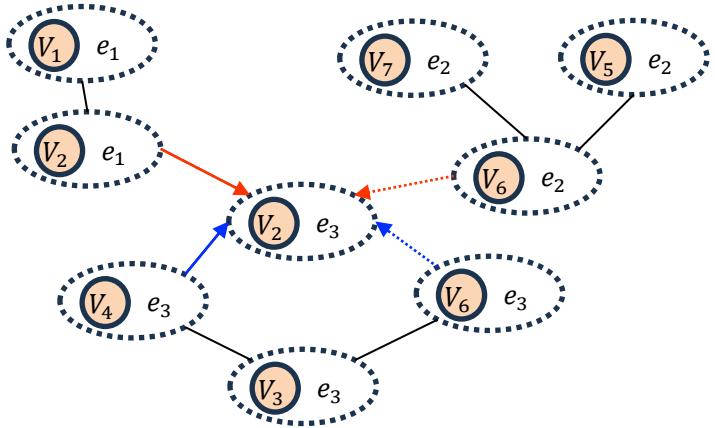
$$Y = P'_{vertex} H^{(K)} \in \mathbb{R}^{|\mathcal{V}| \times d_o}$$



Line Expansion (LE)

Hypergraph Learning with LE

- Acceleration : Neighborhood Sampling
  - Approximate the overall hyperedge / vertex neighboring information
    - $|\mathcal{N}_E(v,e)| > \delta_e$, randomly sample δ_e elements from $\mathcal{N}_E(v,e)$
    - $|\mathcal{N}_V(v,e)| > \delta_v$, randomly sample δ_v elements from $\mathcal{N}_V(v,e)$



Line Expansion (LE)

Experiments

- Hypergraph Node Classification
  - More effective in terms of learning representation

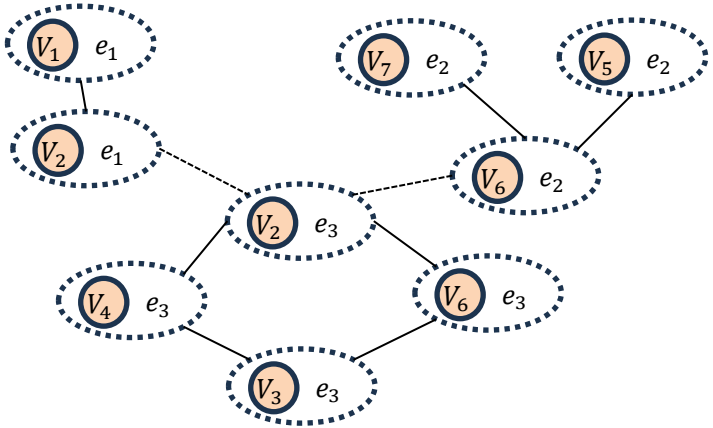
Table 3: Accuracy and Running Time Comparison on Real-world Hypergraphs (%)

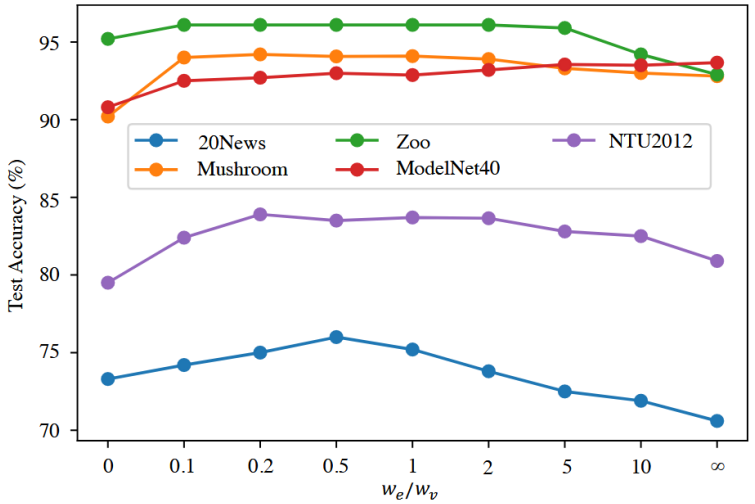
Model	20News	Mushroom	Zoo	ModelNet40	NTU2012
LR	72.9 ± 0.7	81.6 ± 0.1	74.3 ± 0.0	59.0 ± 2.8	37.5 ± 2.1
Star_{GCN}	68.8 ± 0.4	91.8 ± 0.3	95.2 ± 0.0	90.0 ± 0.0	79.1 ± 0.0
Clique_{GCN}	69.0 ± 0.3	90.0 ± 0.6	94.8 ± 0.3	89.7 ± 0.4	78.9 ± 0.8
H-NCut [49]	72.8 ± 0.5	87.7 ± 0.2	87.3 ± 0.5	91.4 ± 1.1	74.8 ± 0.9
LHCN [4]	69.1 ± 0.4 (37.6s)	90.2 ± 0.3 (18.4s)	55.8 ± 0.1 (1.8s)	90.2 ± 0.2 (145.1s)	79.9 ± 0.5 (27.4s)
Hyper-Conv [3]	73.1 ± 0.7 (72.6s)	93.7 ± 0.6 (10.6s)	93.1 ± 2.3 (0.8s)	91.1 ± 0.8 (63.1s)	79.4 ± 1.3 (6.3s)
HGNN [17]	74.3 ± 0.2 (74.2s)	93.1 ± 0.5 (16.1s)	92.0 ± 2.8 (0.8s)	91.7 ± 0.4 (61.0s)	80.0 ± 0.7 (5.6s)
HyperGCN [46]	73.6 ± 0.3 (147.8s)	92.3 ± 0.3 (30.23s)	93.1 ± 2.3 (1.1s)	91.4 ± 0.9 (86.5s)	80.4 ± 0.7 (9.7s)
LE_{GCN}	**75.6 ± 0.2** (38.6s)	**95.2 ± 0.1** (18.9s)	**97.0 ± 0.0** (2.8s)	**94.1 ± 0.3** (85.9s)	**83.2 ± 0.2** (9.9s)

Line Expansion (LE)

Experiments

- Ablation Study on w_e and w_v
  - beneficial to aggregate information from both edge-similar and vertex-similar neighbors
  - For hypergraphs with fewer / sufficient hyperedges, the peak appears before / after $\frac{w_e}{w_v} = 1$





Conclusion

Introduction

- Hypergraphs provide a natural representation for many world datasets
- Hypergraphs are irreversible

Several methods

- Generalize spectral clustering to hypergraphs
  - Based on the graph cut theory
- Apply Graph Convolution Network to hypergraphs
  - Convert hypergraph into simple graph based on the CE's variant
  - HGNN, HyperGCN, HNHN, AllSet
- Propose a novel expansion
  - Line Expansion

Appendix

Normalized hypergraph cut

- Hypergraph boundary ∂S of S

$$\text{vol}\,\partial S \coloneqq \sum_{e \in \partial S} w(e) \frac{|e \cap S||e \cap S^c|}{\delta(e)}$$

$$\underset{\emptyset \neq S \subset V}{\text{argmin}}\, c(S) := \text{vol}\,\partial S \left(\frac{1}{\text{vol}\, S} + \frac{1}{\text{vol}\, S^c} \right)$$

Appendix

Random walk explanation

- The stationary distribution $\boldsymbol{\pi}$ of the random walk
  - $\pi(v) = \frac{d(v)}{vol\ V}$
- Probability of moving from vertex u to vertex v
  - $p(u,v) = \sum_{e \in E} w(e) \frac{h(u,e)}{d(u)} \frac{h(v,e)}{\delta(e)}$
- Rewrite $c(S)$
  - $c(S) = \frac{vol \partial S}{vol\ V} \left(\frac{1}{vol\ S / vol\ V} + \frac{1}{vol\ S^c / vol\ V} \right)$
  - Minimize red one, maximize blue ones

Appendix

Random walk explanation

- The probability with which the random walk occupies some vertex in S
  - $\frac{vol\ S}{vol\ V} = \sum_{v \in S} \frac{d(v)}{vol\ V} = \sum_{v \in S} \pi(v)$
- The probability with which one sees a jump from S to S^c
  - $\frac{vol\ \partial S}{vol\ V} = \sum_{u \in S} \sum_{v \in S^c} \pi(u) p(u, v)$

Appendix

Spectral hypergraph embedding

- The solution to the optimization problem
  - Eigenvector Φ of Δ associated with its smallest nonzero eigenvalue
  - $S = \{v \in V \,|\Phi(v) \geq 0\}, S^c = \{v \in V \,|\Phi(v) < 0\}$

- Extend to k-way partition by $(V_1, \cdots, V_k)$
  - Minimize $c(V_1, \cdots, V_k) = \sum_{i=1}^{k} \frac{vol\, \partial V_i}{vol\, V_i}$ over all k-way partitions
  - Utilize the eigenvectors of Δ associated with the k smallest eigenvalues

- How to utilize multiple eigenvectors?
  - Form a matrix $X = [\Phi_1 \cdots \Phi_k]$
  - The row vectors of X : the representations of the graph vertices in k-dimensional Euclidian space

Appendix

HyperGCN

- Neural message-passing framework

$$h_v^{(\tau+1)} = \sigma\left(\left(\Theta^{(\tau)}\right)^T \sum_{u \in N(v)} \left(\left[\bar{A}_S^{(\tau)}\right]_{v,u} \cdot h_u^{(\tau)}\right)\right)$$

Appendix

HNHN

Weight simplification

For this purpose, let $B \in \mathbb{R}^{(n+m)\times(n+m)}$ be the adjacency matrix of G_*. Then

$$B = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}$$

If we set $W_E = W_V = W$ and $b_E = b_V = b$ in the update rule, we have for all i:

$$X_E^{i+1} = \sigma(A^T X_V^i W + b) \text{ and } X_V^{i+1} = \sigma(A X_E^{i+1} W + b)$$

and for all i we define X^{2i} to be the concatenation of X_V^i with the $m \times d$ matrix of zeroes, and X^{2i+1} to be the concatenation of the $n \times d$ matrix of zeroes with X_E^{i+1}, then for all i we have

$$X^{i+1} = \sigma(B X^i W + b) \tag{2}$$

Appendix

HNHN

- Linear simplification

Lemma 2.2. *If we define*

$$X'_E = A^T X_V W_E + b_E \qquad \text{and} \qquad X'_V = A X'_E W_V + b_V$$

then

$$X'_V = C X_V W_c + b_c \tag{3}$$

where C is the adjacency matrix of G_c, $W_c \in \mathbb{R}^{d\times d}$, and $b_c \in \mathbb{R}^d$.

Proof. We have

$$X'_V = A X'_E W_V + b_V = A(A^T X_V W_E + b_E) W_V + b_V = A A^T X_V W_E W_V + A b_E W_V + b_V.$$

Setting $W_c = W_E W_V$ and $b_c = A b_E W_V + b_V$, and using $C = AA^T$, we obtain (3). □

Appendix

HNHN

- Hypergraph normalization
  - $\left(D_{E,r,\alpha}\right)_{j_1 j_2} = \begin{cases} \left|N_{j_1}\right|^{\alpha} & j_1 = j_2 \\ 0 & j_1 \neq j_2 \end{cases}$
  - $\left(D_{V,l,\alpha}\right)_{i_1 i_2} = \begin{cases} \sum_{j \in N_{i_1}} \left|N_j\right|^{\alpha} & i_1 = i_2 \\ 0 & i_1 \neq i_2 \end{cases}$
  - $(X'_V)_{it} = \frac{\sum_{j \in N_i} |N_j|^{\alpha} \sum_{s=1}^{d} (X'_E)_{is} (W_V)_{st}}{\sum_{j \in N_i} |N_j|^{\alpha}} + (b_V)_t$

Appendix

HNHN

- Hypergraph normalization
  - $\left(D_{V,r,\beta}\right)_{i_1 i_2} = \begin{cases} \left|N_{i_1}\right|^{\beta} & i_1 = i_2 \\ 0 & i_1 \neq i_2 \end{cases}$
  - $\left(D_{E,l,\beta}\right)_{j_1 j_2} = \begin{cases} \sum_{i \in N_{j_1}} |N_i|^{\beta} & j_1 = j_2 \\ 0 & j_1 \neq j_2 \end{cases}$

Appendix

AllSetTransformer

- LN : the layer normalization, || : concatenation, $\theta \in \mathbb{R}^{1\times hF_h}$: a learnable weight
- $\mathrm{MH}_{h,\omega}$: a multihead attention mechanism with h heads and activation function ω

$$\text{AllSetTransformer: } f_{\mathcal{V}\to\mathcal{E}}(S) = f_{\mathcal{E}\to\mathcal{V}}(S) = \mathrm{LN}\left(\mathbf{Y} + \mathrm{MLP}(\mathbf{Y})\right),$$
$$\text{where } \mathbf{Y} = \mathrm{LN}\left(\theta + \mathrm{MH}_{h,\omega}(\theta, \mathbf{S}, \mathbf{S})\right), \mathrm{MH}_{h,\omega}(\theta, \mathbf{S}, \mathbf{S}) = \|_{i=1}^{h} \mathbf{O}^{(i)},$$
$$\mathbf{O}^{(i)} = \omega\left(\theta^{(i)}(\mathbf{K}^{(i)})^T\right)\mathbf{V}^{(i)}, \theta \triangleq \|_{i=1}^{h}\theta^{(i)},\ \mathbf{K}^{(i)} = \mathrm{MLP}^{K,i}(\mathbf{S}),\ \mathbf{V}^{(i)} = \mathrm{MLP}^{V,i}(\mathbf{S})$$

Appendix

Hypergraph Learning with LE

- $A_l(u_l, v_l) = \begin{cases} w_e & u_l = (v, e), v_l = (v', e'), v = v', \\ w_v & u_l = (v, e), v_l = (v', e'), e = e', \\ 0 & otherwise \end{cases}$
- $P_{vertex}(v_l, v) = \begin{cases} 1 \text{ if the vertex part of } v_l \text{is } v, \\ 0 \quad otherwise \end{cases}$
- $P'_{vertex}(v, v_l) = \begin{cases} \frac{\frac{1}{\delta(e)}}{\sum_{(v,e') \in \mathcal{V}_l} \frac{1}{\delta(e')}} \text{ if } v \text{ is the vertex part of } v_l, \\ 0 \quad otherwise \end{cases}$

Appendix

☐ Acceleration : Neighborhood sampling

$$\sum_{e'} w_e h^{(k)}_{(v,e')} \approx \frac{|\mathcal{N}_E(v,e)|}{\delta_e} \sum_{i=1:e'_i \sim \mathcal{N}_E(v,e)}^{i=\delta_e} w_e h^{(k)}_{(v,e'_i)}$$

$$\sum_{v'} w_v h^{(k)}_{(v',e)} \approx \frac{|\mathcal{N}_V(v,e)|}{\delta_v} \sum_{i=1:v'_i \sim \mathcal{N}_V(v,e)}^{i=\delta_v} w_v h^{(k)}_{(v^i_i,e)}$$