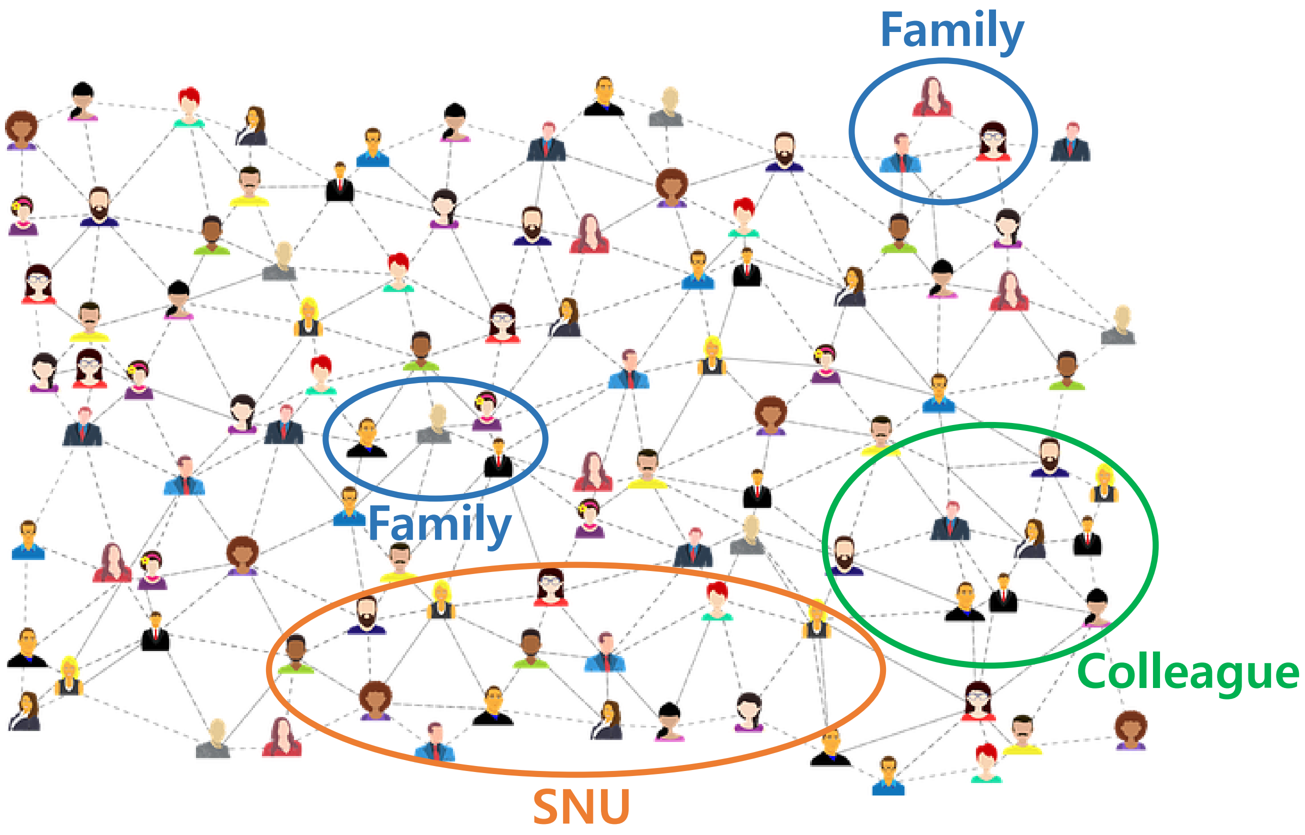
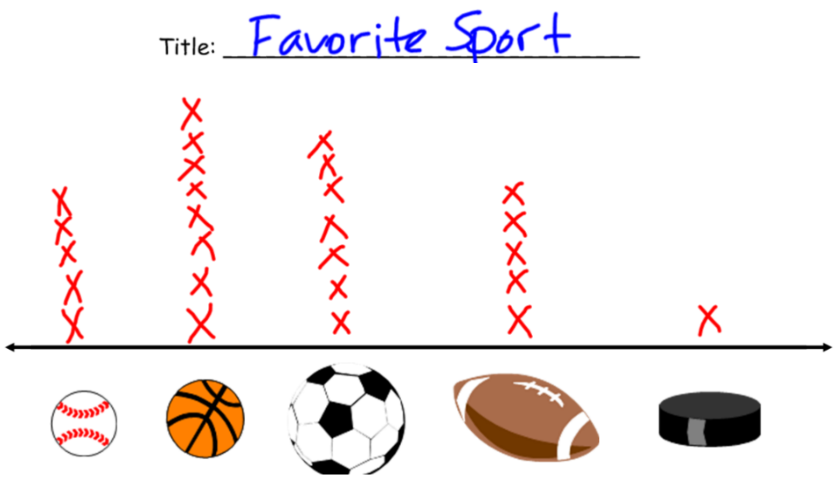
Hypergraph Neural Networks

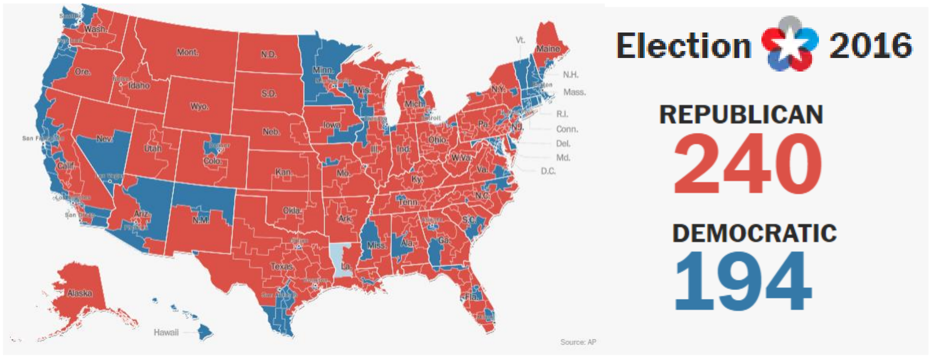
Yifan Feng et al. AAAI 2019

Presenter : Dae Ho Um

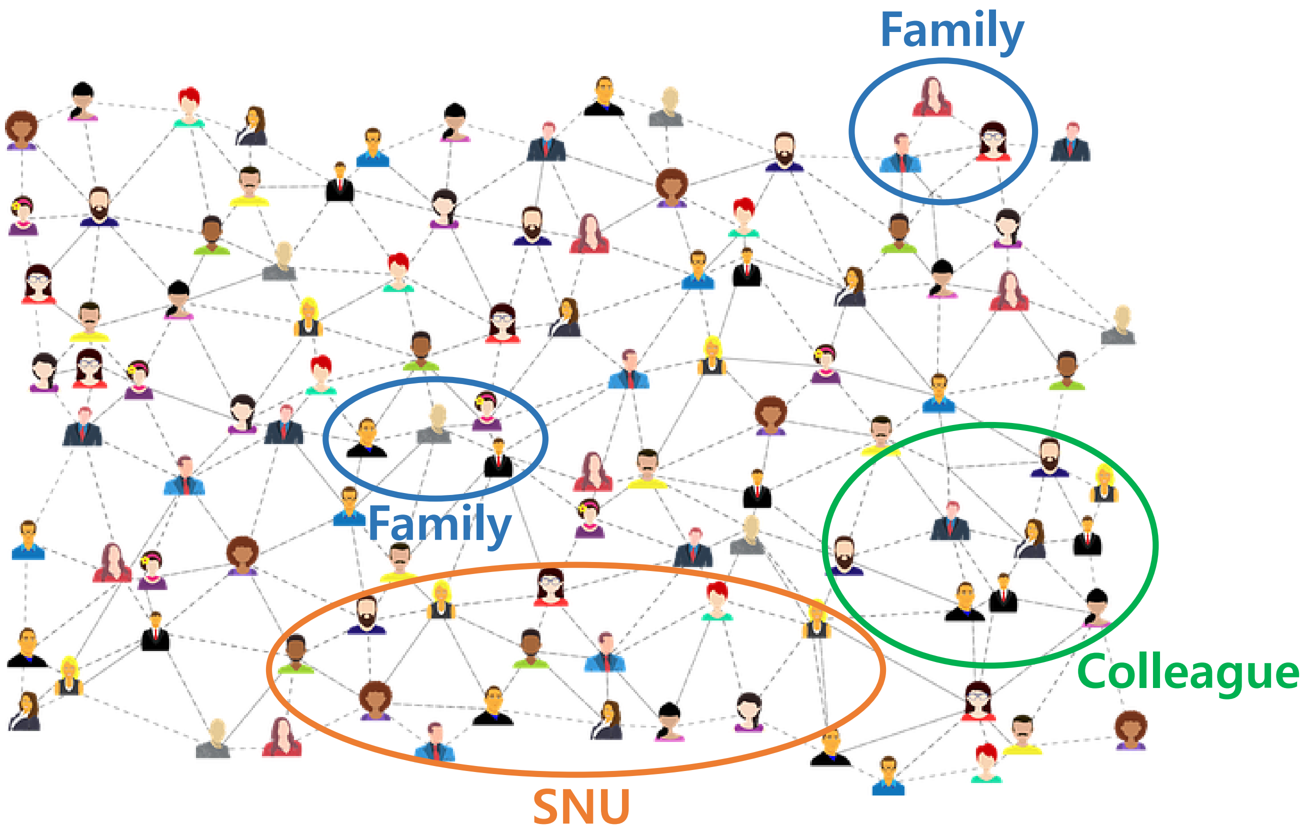
Data structure in real practice?

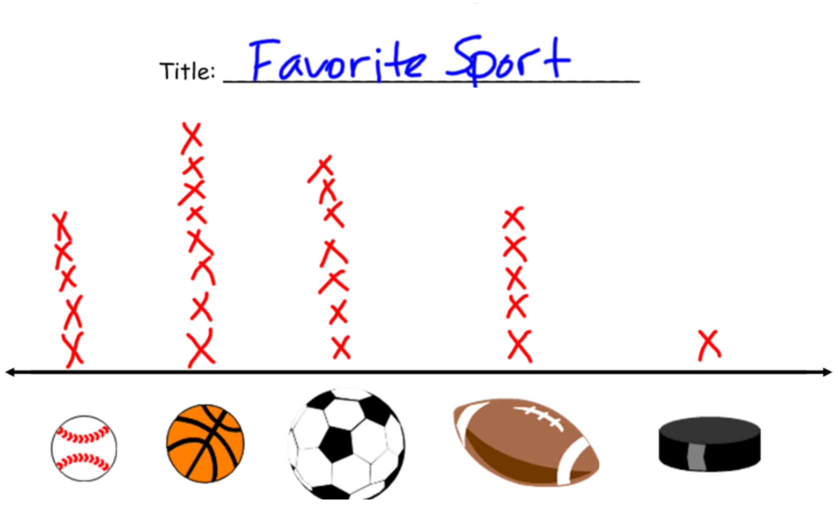


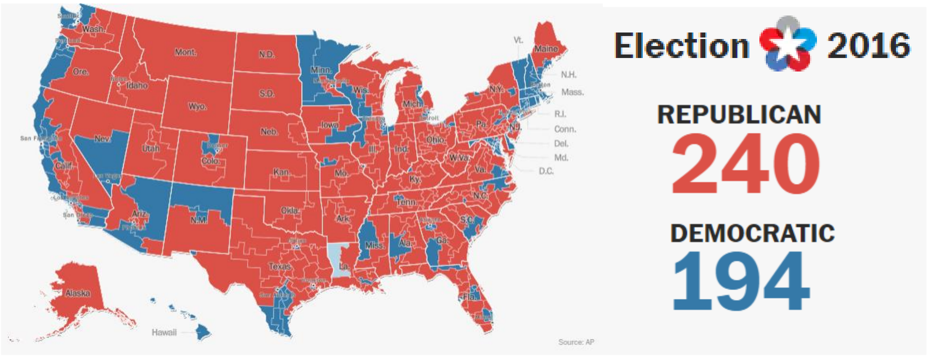




Data structure in real practice?

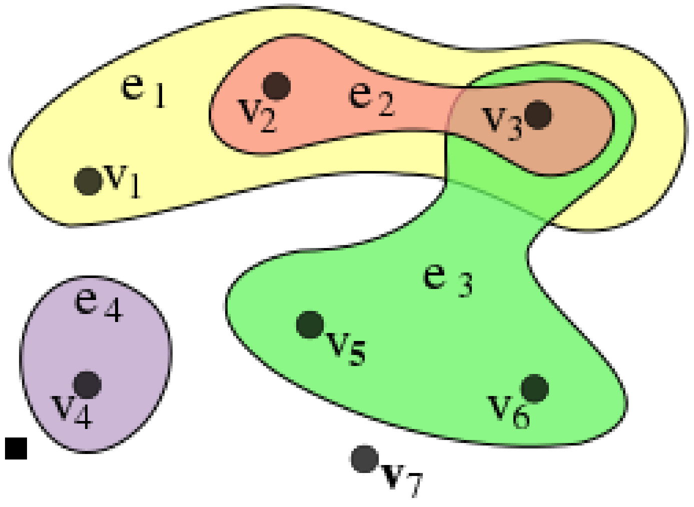






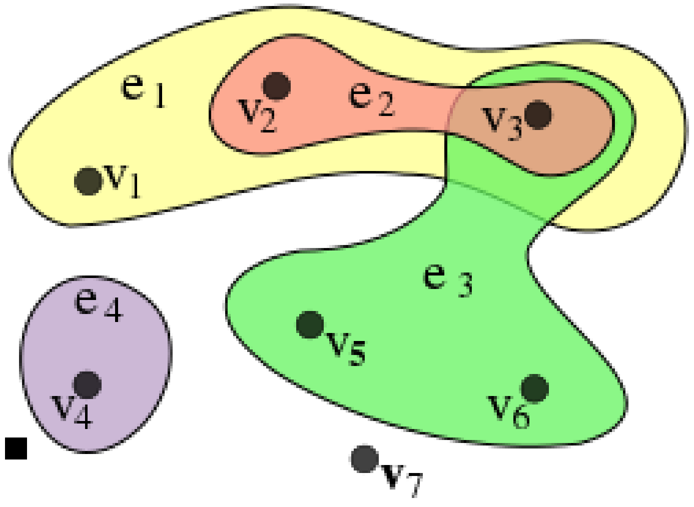
⇒ Often beyond pairwise connections!

Hypergraph



- Hypergraph is a generalization of a graph in which an edge can connect any number of vertices.
- $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$
  - $\mathcal{V}$: a vertex set
  - $\mathcal{E}$: a hyperedge set
  - $\mathbf{W}$: diagonal matrix of edge weights

Hypergraph



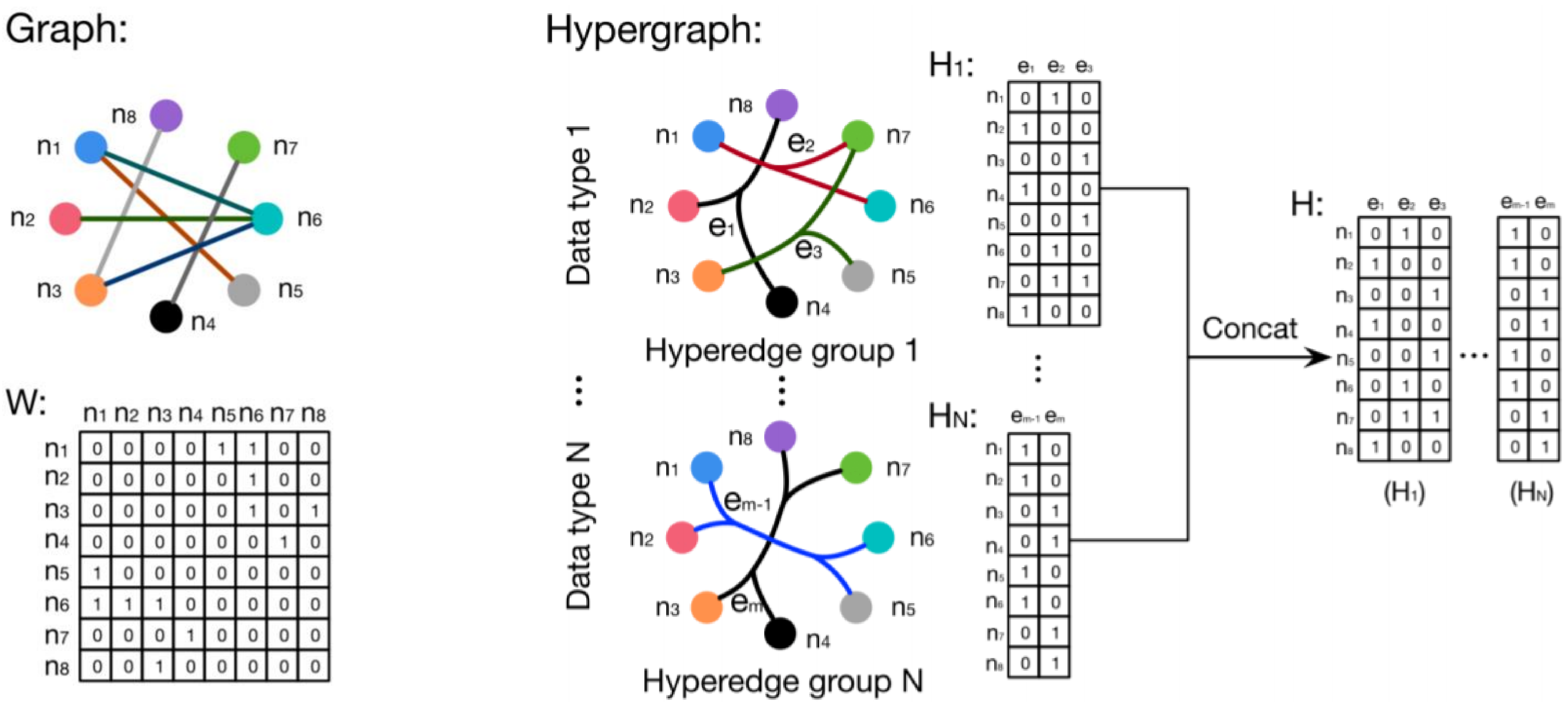
	e_1	e_2	e_3	e_4
v_1	1	0	0	0
v_2	1	1	0	0
v_3	1	1	1	0
v_4	0	0	0	1
v_5	0	0	1	0
v_6	0	0	1	0
v_7	0	0	0	0

- Hypergraph is a generalization of a graph in which an edge can connect any number of vertices.

- $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$
  - $\mathcal{V}$: a vertex set
  - $\mathcal{E}$: a hyperedge set
  - $\mathbf{W}$: diagonal matrix of edge weights

- $\mathbf{H}$: incidence matrix , $|\nu| \times |\varepsilon|$

Graph vs Hypergraph



HyperGraph Neural Networks

hypergraph Laplacian $\boldsymbol{\Delta}$

$$\mathbf{H} \quad : \quad h(v,e) = \begin{cases} 1, & \text{if } v \in e \\ 0, & \text{if } v \notin e, \end{cases}$$

$$\mathbf{D}_v \quad : \quad d(v) = \sum_{e \in \mathcal{E}} \omega(e) h(v,e)$$

$$\mathbf{D}_e \quad : \quad \delta(e) = \sum_{v \in \mathcal{V}} h(v,e)$$

$$\boldsymbol{\Theta} = \mathbf{D}_v^{-1/2} \mathbf{H} \mathbf{W} \mathbf{D}_e^{-1} \mathbf{H}^\top \mathbf{D}_v^{-1/2}$$

$$\Rightarrow \quad \boldsymbol{\Delta} = \mathbf{I} - \boldsymbol{\Theta}$$

HyperGraph Neural Networks

hypergraph Laplacian $\boldsymbol{\Delta}$

$$\mathbf{H} \quad : \quad h(v,e) = \begin{cases} 1, & \text{if } v \in e \\ 0, & \text{if } v \notin e, \end{cases}$$

$$\mathbf{D}_v \quad : \quad d(v) \ = \textstyle\sum_{e \in \mathcal{E}} \omega(e) h(v,e)$$

$$\mathbf{D}_e \quad : \quad \delta(e) = \textstyle\sum_{v \in \mathcal{V}} h(v,e)$$

$$\boldsymbol{\Theta} = \mathbf{D}_v^{-1/2} \mathbf{H} \mathbf{W} \mathbf{D}_e^{-1} \mathbf{H}^\top \mathbf{D}_v^{-1/2}$$

$$\Rightarrow \quad \boldsymbol{\Delta} = \mathbf{I} - \boldsymbol{\Theta}$$

$$f^\top \boldsymbol{\Delta} f \ = \frac{1}{2} \sum_{e \in \mathcal{E}} \sum_{\{u,v\} \in \mathcal{V}} \frac{w(e)h(u,e)h(v,e)}{\delta(e)} \left(\frac{f(u)}{\sqrt{d(u)}} - \frac{f(v)}{\sqrt{d(v)}} \right)^2$$

HyperGraph Neural Networks

spectral convolution using the truncated ChebyShev expansion

$$
\begin{aligned}
\mathbf{g} \star \mathbf{x} = \ & \mathbf{\Phi} g(\mathbf{\Lambda}) \mathbf{\Phi}^\top \mathbf{x} \\
& \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{\mathbf{\Delta}}) \mathbf{x} \quad (\tilde{\mathbf{\Delta}} = \frac{2}{\lambda_{max}} \mathbf{\Delta}) \\
& \approx \theta_0 \mathbf{x} - \theta_1 \mathbf{D}^{-1/2} \mathbf{H} \mathbf{W} \mathbf{D_e}^{-1} \mathbf{H}^\top \mathbf{D_v}^{-1/2} \mathbf{x} \\
& \approx \frac{1}{2} \theta \mathbf{D}_v^{-1/2} \mathbf{H} (\mathbf{W} + \mathbf{I}) \mathbf{D}_e^{-1} \mathbf{H}^\top \mathbf{D}_v^{-1/2} \mathbf{x} \\
& \approx \theta \mathbf{D}_v^{-1/2} \mathbf{H} \mathbf{W} \mathbf{D}_e^{-1} \mathbf{H}^\top \mathbf{D}_v^{-1/2} \mathbf{x},
\end{aligned}
$$

$\lambda_{max} \approx 2$, $K = 1$

$$
\begin{cases} \theta_1 = -\frac{1}{2}\theta \\ \theta_0 = \frac{1}{2}\theta \mathbf{D}_v^{-1/2} \mathbf{H} \mathbf{D}_e^{-1} \mathbf{H}^\top \mathbf{D}_v^{-1/2} \end{cases}
$$

hyperedge convolution

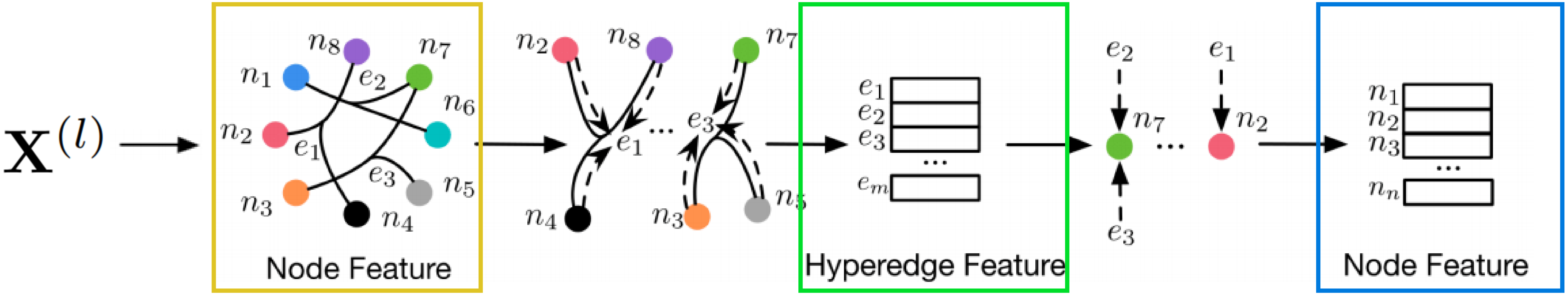
$$
\Rightarrow \quad \mathbf{Y} = \mathbf{D_v}^{-1/2} \mathbf{H} \mathbf{W} \mathbf{D_e}^{-1} \mathbf{H}^\top \mathbf{D_v}^{-1/2} \mathbf{X} \mathbf{\Theta}
$$

HyperGraph Neural Networks

- hyperedge convolutional layer

$$\mathbf{X}^{(l+1)} = \sigma\left(\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}\mathbf{X}^{(l)}\mathbf{\Theta}^{(l)}\right)$$

N by E $\quad$ E by N



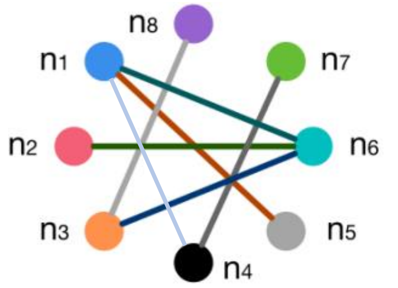
$\mathbb{R}^{N \times C_1}$ $\quad$ $\mathbb{R}^{N \times C_2}$ $\quad$ $\mathbb{R}^{E \times C_2}$ $\quad$ $\mathbb{R}^{N \times C_2}$

Experiments

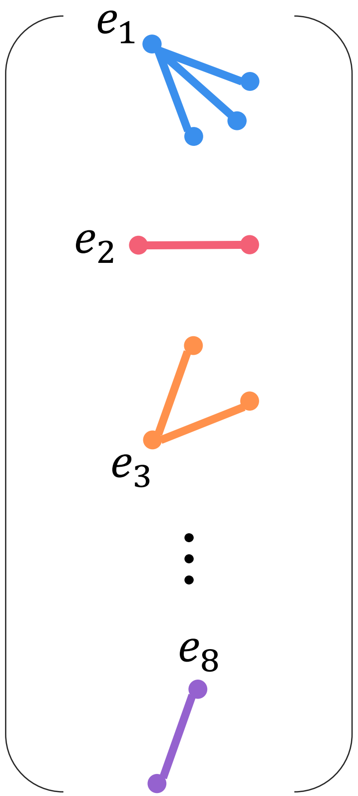
- Citation network classification

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

- Hypergraph generation



pairwise graph, A



Hypergraph, H

Experiments

- Citation network classification

  - Results

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%**

Table 2: Classification results on the Cora and Pubmed datasets.

Experiments

- Visual object classification

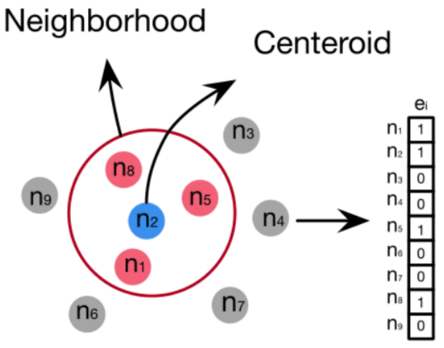
Dataset	ModelNet40	NTU
Objects	12311	2012
MVCNN Feature	4096	4096
GVCNN Feature	2048	2048
Training node	9843	1639
Testing node	2468	373
Classes	40	67

- A, Affinity Matrix (for GCN)

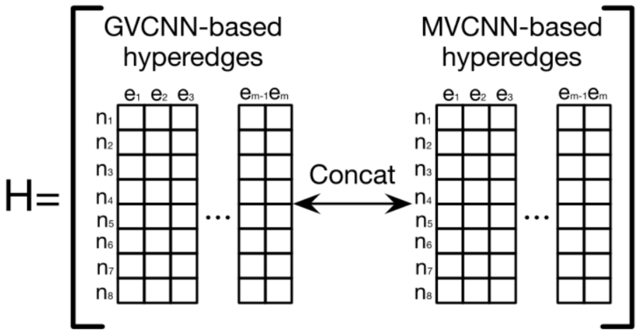
$$A_{ij} = \exp\left(-\frac{2Dij^2}{\Delta}\right)$$

(D : Euclidean distance, Δ : Average Euclidean distance)

- H, Incidence Matrix (for HGNN)



Hyperedge Generation



Incidence Matrix Generation

Experiments

- Visual object classification

- Results

Feature	Features for Structure					
	GVCNN		MVCNN		GVCNN+MVCNN	
	GCN	HGNN	GCN	HGNN	GCN	HGNN
GVCNN (Feng et al. 2018)	91.8%	**92.6%**	91.5%	**91.8%**	92.8%	**96.6%**
MVCNN (Su et al. 2015)	92.5%	**92.9%**	86.7%	**91.0%**	92.3%	**96.6%**
GVCNN+MVCNN	-	-	-	-	94.4%	**96.7%**

Table 4: Comparison between GCN and HGNN on the ModelNet40 dataset.

Feature	Features for Structure					
	GVCNN		MVCNN		GVCNN+MVCNN	
	GCN	HGNN	GCN	HGNN	GCN	HGNN
GVCNN ((Feng et al. 2018))	78.8%	**82.5%**	78.8%	**79.1%**	75.9%	**84.2%**
MVCNN ((Su et al. 2015))	74.0%	**77.2%**	71.3%	**75.6%**	73.2%	**83.6%**
GVCNN+MVCNN	–	–	–	–	76.1%	**84.2%**

Table 5: Comparison between GCN and HGNN on the NTU dataset.

Method	Classification Accuracy
PointNet (Qi et al. 2017a)	89.2%
PointNet++ (Qi et al. 2017b)	90.7%
PointCNN (Li et al. 2018)	91.8%
SO-Net (Li, Chen, and Lee 2018)	93.4%
HGNN	**96.7%**

Table 6: Experimental comparison among recent classification methods on ModelNet40 dataset.

Conclusion

- HGNN is a more general framework which is able to handle the complex and high-order correlations through the hypergraph structure for representation learning compared with traditional graph.

- HGNN generalizes the convolution operation to the hypergraph learning process.

감사합니다

Reproduce

visual object classfication

- code at GitHub Link

Installation

- install Pytorch 0.4.0 and yaml.
- The code has been tested with Python 3.6, Pytorch 0.4.0 and CUDA 10.1 on Ubuntu 16.04.

Usage

- configure the "data_root" and "result_root" path in config/config.yaml.
- Download ModelNet40_mvcnn_gvcnn_feature, NTU2012_mvcnn_gvcnn_feature and move them to data folder.
- python train.py

visual object classfication

- A brief explaination about the code

(1) load feature

(2) **construct hypergraph adjacency matrix H**

(3) **generate G(= $\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}$) for hyperedge convolutional layer**

(4) build a two-layer model

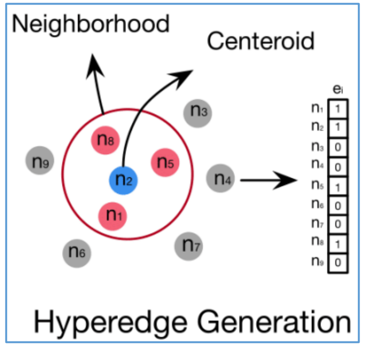
(5) train

visual object classfication

(2) **construct H by KNN**

```python
construct_H_with_KNN_from_distance(dis_mat, k_neig, is_probH=True, m_prob=1):
    """
    construct hypregraph incidence matrix from hypergraph node distance matrix
    :param dis_mat: node distance matrix
    :param k_neig: K nearest neighbor
    :param is_probH: prob Vertex-Edge matrix or binary
    :param m_prob: prob
    :return: N_object X N_hyperedge
    """
    n_obj = dis_mat.shape[0]
    # construct hyperedge from the central feature space of each node
    n_edge = n_obj
    H = np.zeros((n_obj, n_edge))
    for center_idx in range(n_obj):
        dis_mat[center_idx, center_idx] = 0
        dis_vec = dis_mat[center_idx]
        nearest_idx = np.array(np.argsort(dis_vec)).squeeze()
        avg_dis = np.average(dis_vec)
        if not np.any(nearest_idx[:k_neig] == center_idx):
            nearest_idx[k_neig - 1] = center_idx

        for node_idx in nearest_idx[:k_neig]:
            if is_probH:
                H[node_idx, center_idx] = np.exp(-dis_vec[0, node_idx] ** 2 / (m_prob * avg_dis) ** 2)
            else:
                H[node_idx, center_idx] = 1.0
    return H
```



Each time one object is selected as the centroid, and KKN in the selected feature space are used to generate a hyperedge.

$$A_{ij} = \exp(-\frac{2Dij^2}{\Delta})$$

Δ is the average pairwise distance between nodes.

visual object classfication

(3) **generate G(= $\mathbf{D_v^{-1/2} H W D_e^{-1} H^\top D_v^{-1/2}}$) for hyperedge convolutional layer**

```python
_generate_G_from_H(H, variable_weight=False):
    """
    calculate G from hypgraph incidence matrix H
    :param H: hypergraph incidence matrix H
    :param variable_weight: whether the weight of hyperedge is variable
    :return: G
    """
    H = np.array(H)
    n_edge = H.shape[1]
    # the weight of the hyperedge
    W = np.ones(n_edge)
    # the degree of the node
    DV = np.sum(H * W, axis=1)
    # the degree of the hyperedge
    DE = np.sum(H, axis=0)

    invDE = np.mat(np.diag(np.power(DE, -1)))
    DV2 = np.mat(np.diag(np.power(DV, -0.5)))
    W = np.mat(np.diag(W))
    H = np.mat(H)
    HT = H.T

    if variable_weight:
        DV2_H = DV2 * H
        invDE_HT_DV2 = invDE * HT * DV2
        return DV2_H, W, invDE_HT_DV2
    else:
        G = DV2 * H * W * invDE * HT * DV2
        return G
```

$\mathbf{D}_v$

sum along axis=1

$\mathbf{D}_e$

sum along axis=0

	e_1	e_2	e_3	e_4
v_1	1	0	0	0
v_2	1	1	0	0
v_3	1	1	1	0
v_4	0	0	0	1
v_5	0	0	1	0
v_6	0	0	1	0
v_7	0	0	0	0

$$\mathbf{D_v^{-1/2} H W D_e^{-1} H^\top D_v^{-1/2}}$$

visual object classfication

- Results

Paper

Feature	Features for Structure					
	GVCNN		MVCNN		GVCNN+MVCNN	
	GCN	HGNN	GCN	HGNN	GCN	HGNN
GVCNN (Feng et al. 2018)	91.8%	**92.6%**	91.5%	**91.8%**	92.8%	**96.6%**
MVCNN (Su et al. 2015)	92.5%	**92.9%**	86.7%	**91.0%**	92.3%	**96.6%**
GVCNN+MVCNN	-	-	-	-	94.4%	**96.7%**

Table 4: Comparison between GCN and HGNN on the ModelNet40 dataset.

Feature	Features for Structure					
	GVCNN		MVCNN		GVCNN+MVCNN	
	GCN	HGNN	GCN	HGNN	GCN	HGNN
GVCNN ((Feng et al. 2018))	78.8%	**82.5%**	78.8%	**79.1%**	75.9%	**84.2%**
MVCNN ((Su et al. 2015))	74.0%	**77.2%**	71.3%	**75.6%**	73.2%	**83.6%**
GVCNN+MVCNN	–	–	–	–	76.1%	**84.2%**

Table 5: Comparison between GCN and HGNN on the NTU dataset.

Reproduced

```
Training complete in 0m 34s
Best val Acc: 0.968801
```

```
Training complete in 0m 3s
Best val Acc: 0.833780
```