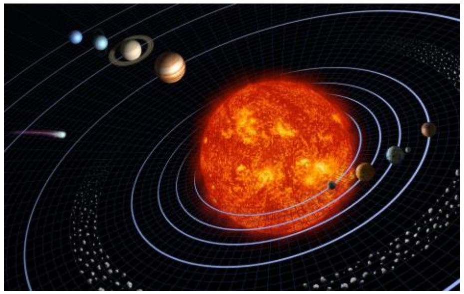
Graph Neural Networks

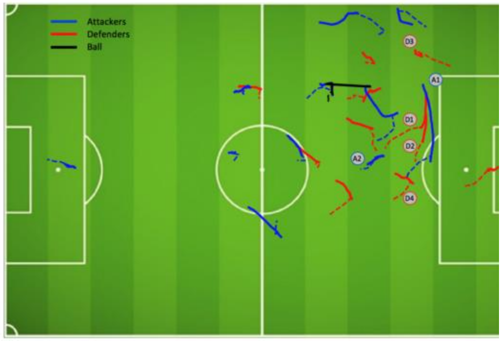
杨帆

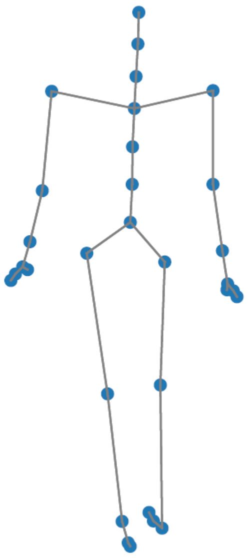
2018年7月10日

Motivations for This Talk (1)

- (My) ultimate goal: to build models that accurately **predict the future**
  - We have already learned some powerful sequential models
    - E.g., RNNs, stochastic RNNs, WaveNet, state space models
  - They are good at modeling a set of **independent** sequences
  - Real-world sequences are often **interacting** with each other
  - Leveraging such interactions can improve the predictive ability of models







Motivations for This Talk (2)

- **Simulators** are perfect tools for forecasting dynamical systems
  - However, building simulators for real environment is not easy
    - Requiring a lot of domain-specific knowledge, e.g., physics
- There is a latest trend toward **learning** simulators from observations
  - Neural Physics Engines, World Models
- Graph NNs are the key building block behind such learned simulators
  - Do well in modeling the relations between objects, while remaining **tractable**

- Besides, Graph NNs are also promising tools for:
  - Representation learning on graphs, i.e., node/graph embedding
  - Generalizing deep models to graph-structured data
  - Semi-supervised learning
  - Model-based reinforcement learning

Outline

- Introduction to Graph Neural Networks
- Hierarchical Pooling in GNN
- Modeling Interacting Systems with GNN
- Conclusions and Further Discussion

Outline

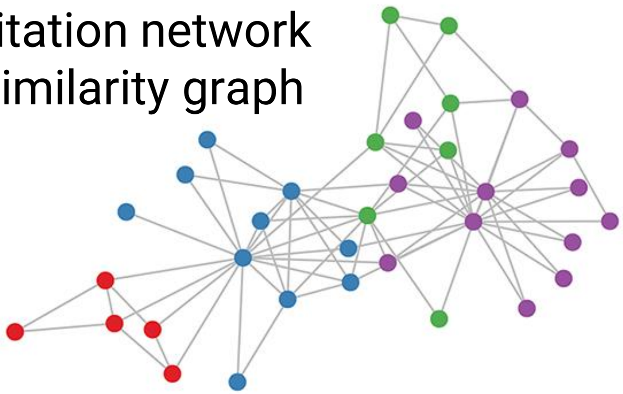
- **Introduction to Graph Neural Networks**
- Hierarchical Pooling in GNN
- Modeling Interacting Systems with GNN
- Conclusions and Further Discussion

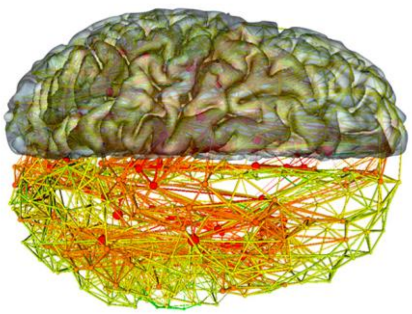
Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. M Defferrard, et al. NIPS 2016.
Semi-Supervised Classification with Graph Convolutional Networks. TN Kipf, et al. ICLR 2017.
Geometric Deep Learning: Going beyond Euclidean data. MM Bronstein, et al. IEEE Signal Processing Magazine, 2017.
Graph Signal Processing: Overview, Challenges, and Applications. A Ortega, et al. Proceedings of IEEE, 2018.
Representation Learning on Graphs: Methods and Applications. WL Hamilton, et al. Arxiv 2018.
Relational inductive biases, deep learning, and graph networks. PW Battaglia, et al. Arxiv 2018.

The Need for Graph Neural Networks

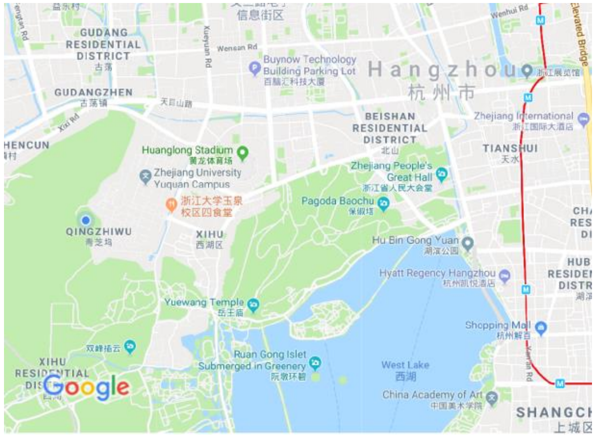
- Graph-structured data are ubiquitous

social network
citation network
similarity graph



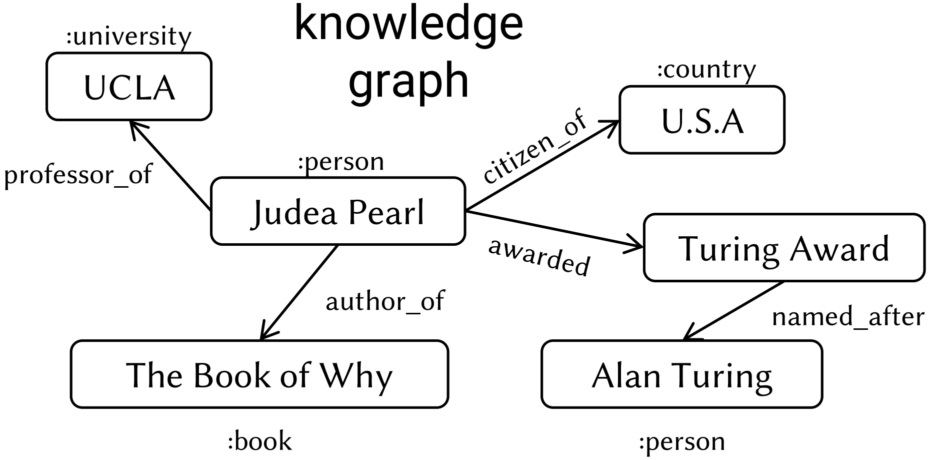


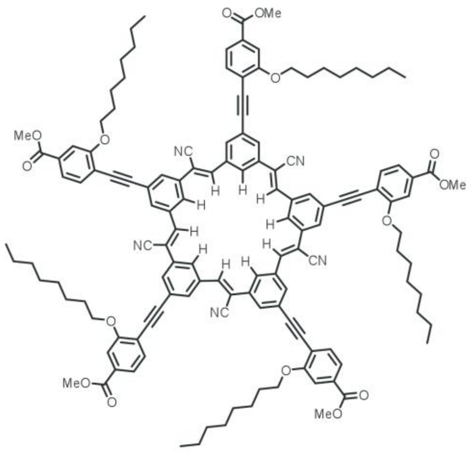
brain network



roadmaps

knowledge graph

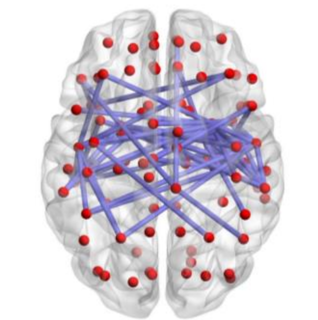




molecules

The Need for Graph Neural Networks

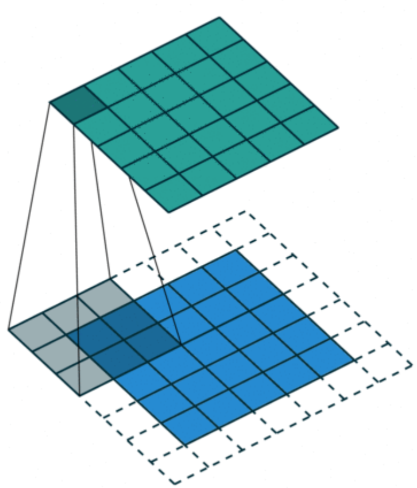
- Consider the basic graph classification task

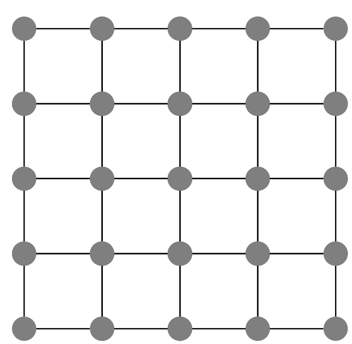


$f\colon \mathbb{R}^{n\times n} \to [0,1]$

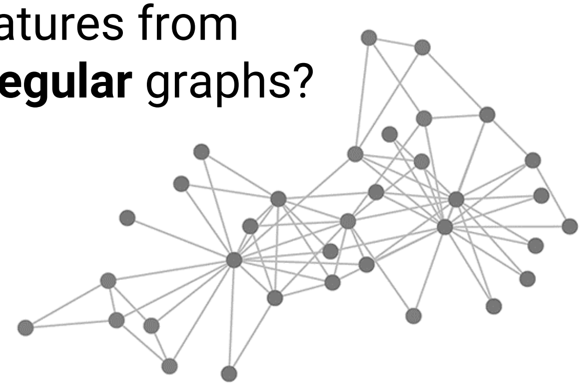
healthy?

- For image classification, we use **convolutions** to extract features





How to extract features from **irregular** graphs?



The Two Faces of Convolutions

- Convolution Theorem

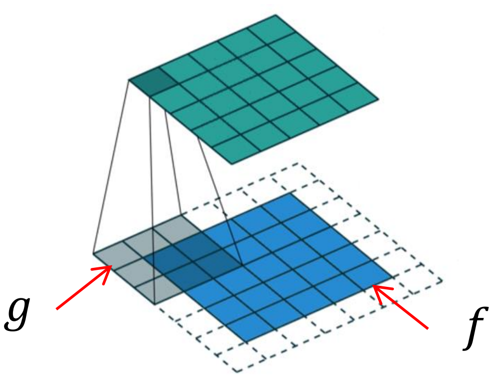
Fourier transform

$$f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\}$$

convolution

inverse Fourier transform

pointwise multiplication



spatial domain

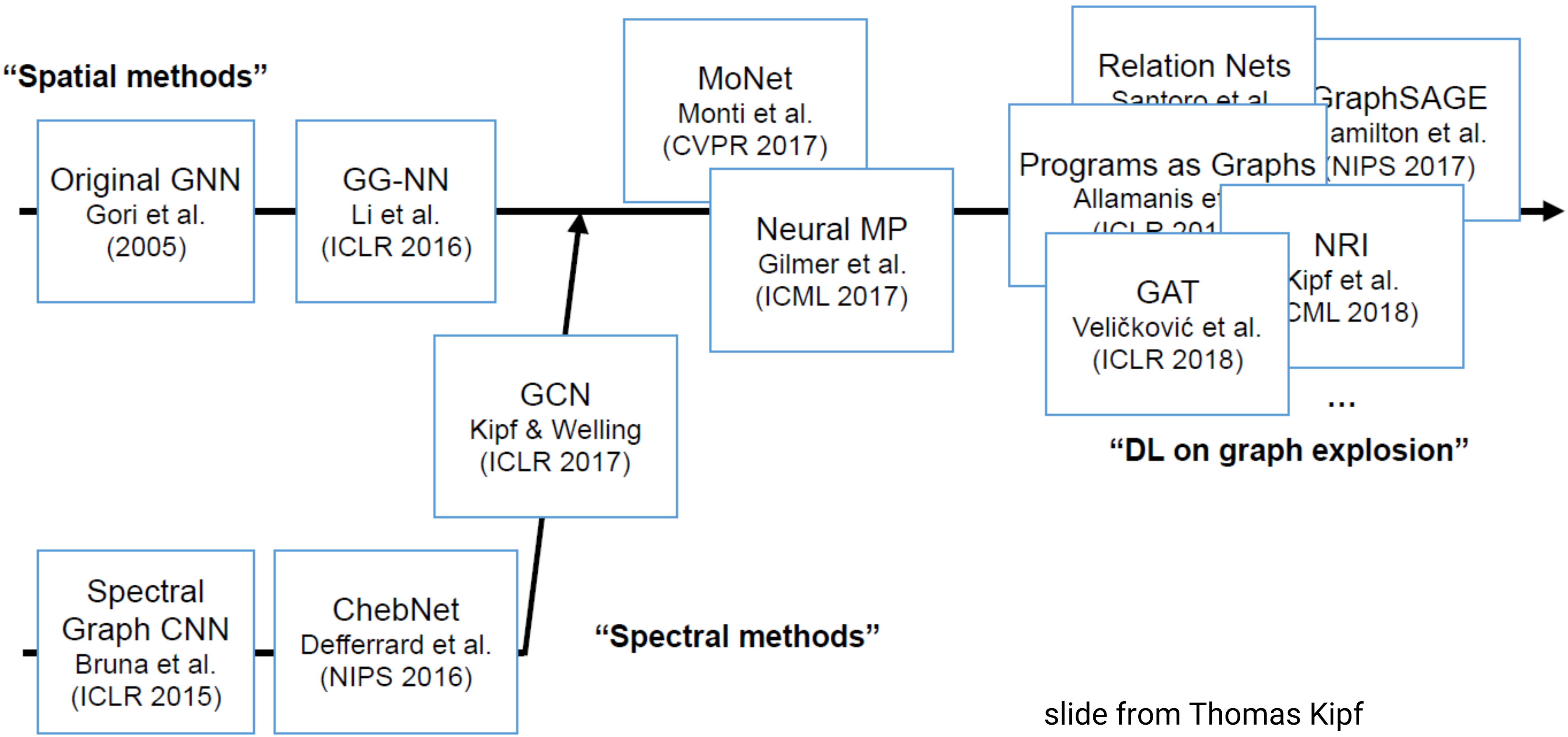
$$\begin{bmatrix} \hat{f}_1 \\ \hat{f}_2 \\ \vdots \\ \hat{f}_n \end{bmatrix} \begin{matrix} \times \\ \times \\ \vdots \\ \times \end{matrix} \begin{bmatrix} \hat{g}_1 \\ \hat{g}_2 \\ \vdots \\ \hat{g}_n \end{bmatrix}$$

spectral domain

- Convolving with a filter ⇔ cross-correlating with the reverse filter
  - i.e., *sliding inner-product*

History of Graph Neural Networks

- GNNs were also developed from two perspectives



slide from Thomas Kipf
http://tkipf.github.io/misc/SlidesCambridge.pdf

Spectral Graph Theory Basics

- Spectral graph convolution is built on top of **spectral graph theory**
  - Study the properties of a graph through the lens of its **Laplacian matrix**
    - Or adjacency matrix

- $G = (V, E)$ is a graph
  - undirected, unweighted, no self-loops
  - $N =$ #vertices
- **adjacency matrix** A is $N \times N$ matrix

$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

- **degree matrix** D is $N \times N$ matrix

$$D_{ij} = \begin{cases} d_i & \text{if } \ i = j \\ 0 & \text{otherwise} \end{cases}$$

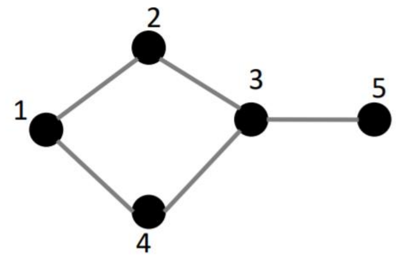
  - $d_i = \sum_j A_{ij}$ is the degree of vertex i
- **Laplacian matrix** $L = D - A$

A

	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	0	0
3	0	1	0	1	1
4	1	0	1	0	0
5	0	0	1	0	0

D

	1	2	3	4	5
1	2	0	0	0	0
2	0	2	0	0	0
3	0	0	3	0	0
4	0	0	0	2	0
5	0	0	0	0	1



L

	1	2	3	4	5
1	2	-1	0	-1	0
2	-1	2	-1	0	0
3	0	-1	3	-1	-1
4	-1	0	-1	2	0
5	0	0	-1	0	1

Spectral Graph Theory Basics

- The weighted and normalized definition

$$A_{ij} = \begin{cases} w_{ij} & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}, \qquad D_{ij} = \begin{cases} \sum_j A_{ij} & \text{if } \; i = j \\ 0 & \text{otherwise} \end{cases}$$

$$L = D^{-1/2}(D - A)D^{-1/2} = I - D^{-1/2}AD^{-1/2}$$

- Laplacian matrix is positive-semidefinite and can be diagonalized as:

$$L = U\Lambda U^T = \sum_{i=1}^{N} \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^T$$

where

$$U = [\boldsymbol{u}_1 \quad \boldsymbol{u}_2 \quad \dots \quad \boldsymbol{u}_N] \in \mathbb{R}^{N \times N}$$

$$\Lambda = \operatorname{diag}([\lambda_1, \lambda_2, \dots, \lambda_N]) \in \mathbb{R}^{N \times N}$$

Graph Signal Processing

- 1D signal: $x \in \mathbb{R}^N$
  - x_i is the value at time i
- Fourier transform

$$\hat{x} = Fx$$

- spectral components

$$\left\{F_k[n] = e^{-j\frac{2\pi}{N}kn}/\sqrt{N} : n = 0, \dots, N-1\right\}_{k=0}^{N-1}$$

- frequencies

$$2\pi k/N, k = 0,1, \dots, N-1$$

- convolution

$$h * x = F^{-1}\big((Fh) \cdot (Fx)\big)$$

- 1D graph signal: $x \in \mathbb{R}^N$
  - x_i is the value at node i
- graph Fourier transform

$$\hat{x} = U^T x$$

- spectral components

$$\{\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_N\}$$

- graph frequencies

$$\lambda_1, \lambda_2, \dots, \lambda_N$$

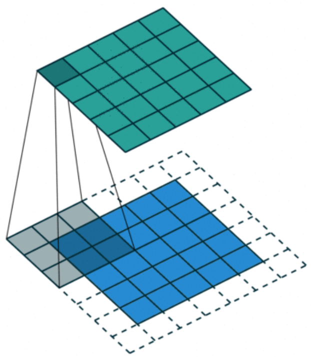
- graph convolution

$$h_\theta * x \triangleq U\big(\theta \cdot (U^T x)\big)$$

diagonalization: $O(N^3)$, matmul: $O(N^2)$ ☹

Spatial Graph NNs

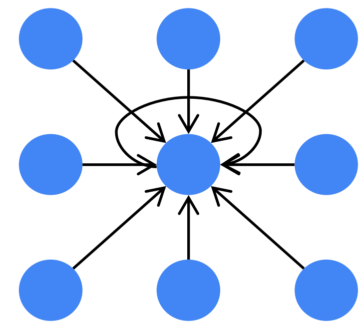
- Spatial Graph NNs are based on the idea of **message passing**
  - Some GNNs were inspired by the spatial interpretation of convolutions



$$h_{\mathrm{p}_i}^{l+1} = \sum_{\mathrm{p}_j \in \mathcal{N}(\mathrm{p}_i)} W_j^l h_{\mathrm{p}_j}^l$$

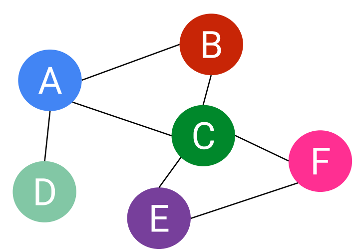
where

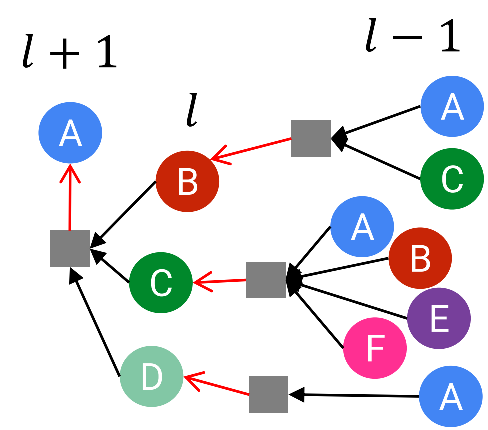
$$\mathcal{N}(\mathrm{p}) = \mathrm{neighbors}(\mathrm{p})$$



- Simple message passing on general undirected graphs

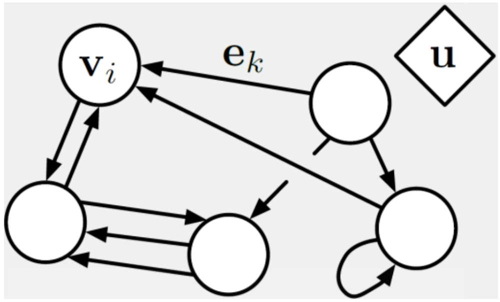
Input: Graph $G = (V, E)$, non-linearity σ,
differentiable AGGREGATE and COMBINE,
weights $\mathbf{W}$, node features $\{h_v^l : v \in V\}$
Output: new node features $\{h_v^{l+1} : v \in V\}$
for $v \in V$ **do**
$h_{\mathcal{N}(v)}^{l+1} \leftarrow \mathrm{AGGREGATE}\left(\{h_u^l : u \in \mathcal{N}(v)\}\right)$
$h_v^{l+1} \leftarrow \sigma\left(\mathbf{W} \cdot \mathrm{COMBINE}\left(h_v^l, h_{\mathcal{N}(v)}^{l+1}\right)\right)$
end





The Graph Network Framework

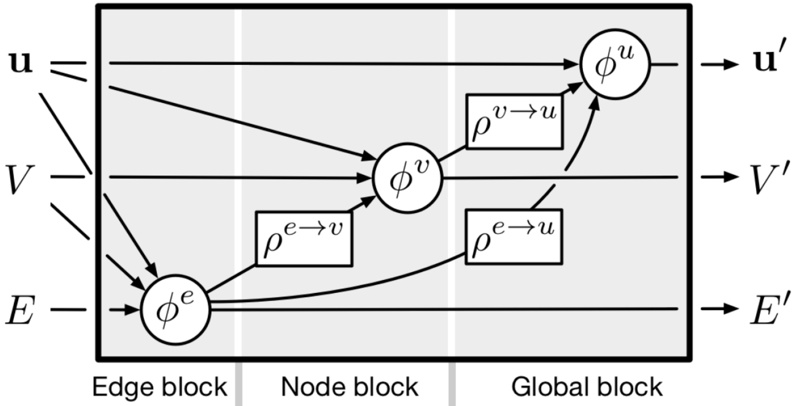
- **G**raph **N**etwork [Battaglia 2018] generalizes and extends various GNNs
  - They use a generalized definition of graph $G = (\mathbf{u}, V, E)$



- Directed: one-way edges $\mathbf{e}_k = (\mathbf{v}_{s_k}, \mathbf{v}_{r_k})$
- Attributed: vertices and edges have attributes
- Attribute: encoded as a vector or set
- Global attribute: a graph-level attribute $\mathbf{u}$
- Multi-graph: there can be more than one edge between vertices, including self-edges

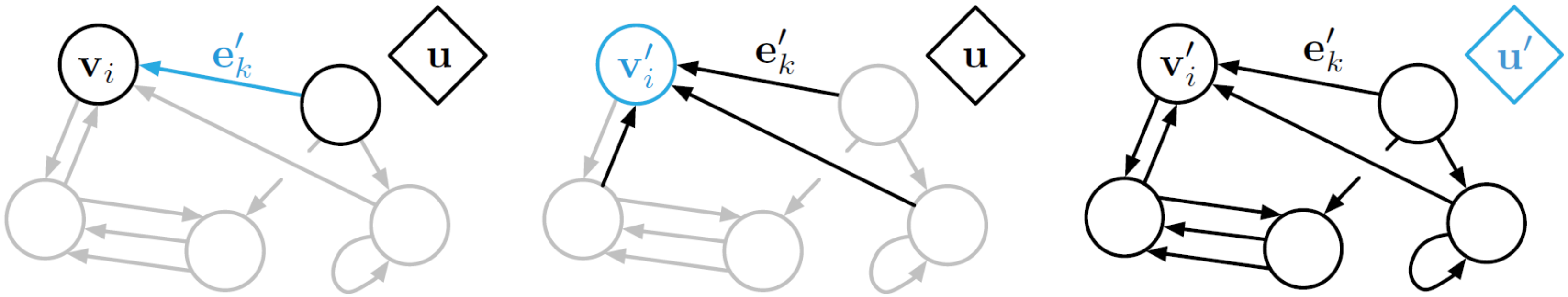
- A GN block takes as input a graph $G = (\boldsymbol{u}, V, E)$
  - And outputs a new graph $G' = (\boldsymbol{u}', V', E')$

- Differentiable and composable

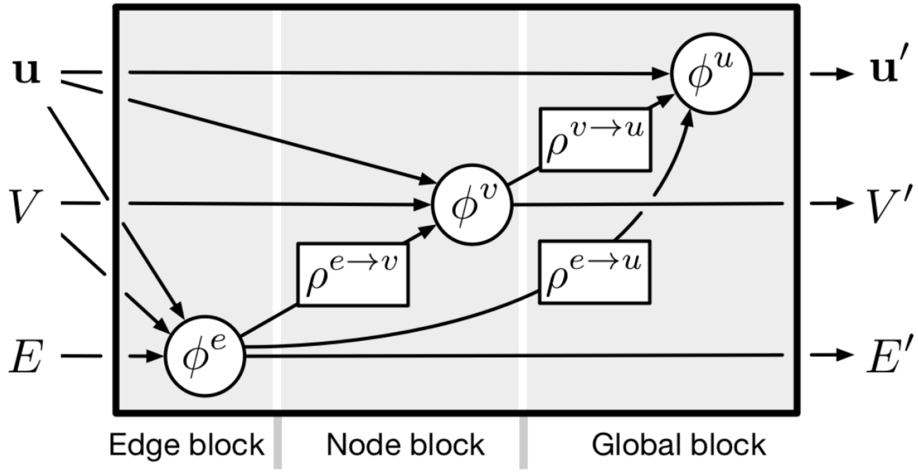


Insides of a Graph Network Block

- A GN block: 3 update functions ϕ and 3 aggregation functions ρ
  - Edge update -> node update -> global update



(a) Edge update (b) Node update (c) Global update



$$\mathbf{e}'_k = \phi^e(\mathbf{e}_k, \mathbf{v}_{s_k}, \mathbf{v}_{r_k}, \mathbf{u})$$
$$\bar{\mathbf{e}}'_i = \rho^{e\to v}(E'_i)$$
$$\mathbf{v}'_i = \phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u})$$
$$\bar{\mathbf{e}}' = \rho^{e\to u}(E')$$
$$\bar{\mathbf{v}}' = \rho^{v\to u}(V')$$
$$\mathbf{u}' = \phi^u(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u})$$

Configuring Within-Block Structure

- Note that a GN block is a computation flow abstraction
- Functions ϕ and ρ must be instantiated with concrete implementation
- Example:

$$\phi^e\left(\mathbf{e}_k, \mathbf{v}_{s_k}, \mathbf{v}_{r_k}, \mathbf{u}\right) := \mathrm{NN}_e\left(\left[\mathbf{e}_k, \mathbf{v}_{s_k}, \mathbf{v}_{r_k}, \mathbf{u}\right]\right)$$

$$\phi^v(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}) := \mathrm{NN}_v([\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}])$$

$$\phi^u(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}) := \mathrm{NN}_u([\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}])$$

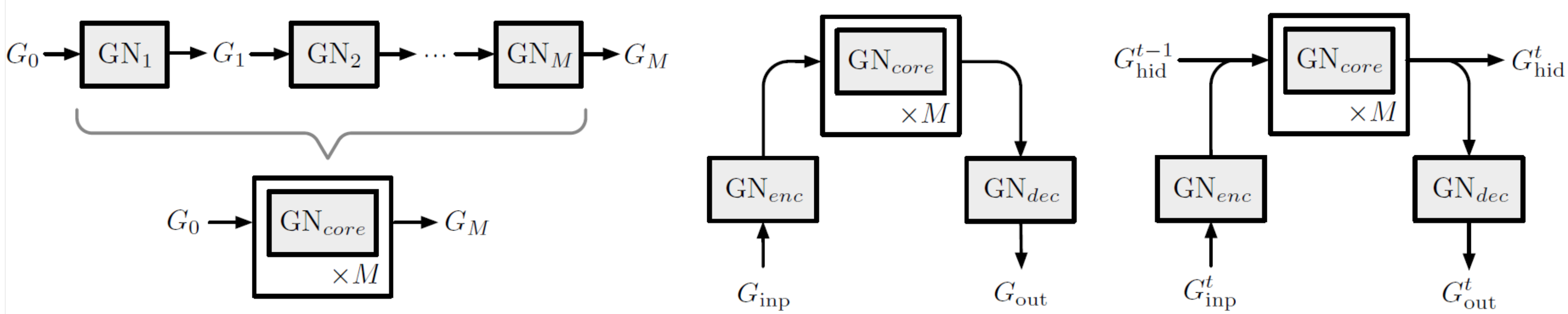
$$\rho^{e \to v}(E'_i) := \sum_k \mathbb{I}(r_k = i)\mathbf{e}'_k$$

$$\rho^{v \to u}(V') := \sum_i \mathbf{v}'_i, \qquad \rho^{e \to u}(E') := \sum_k \mathbf{e}'_k$$

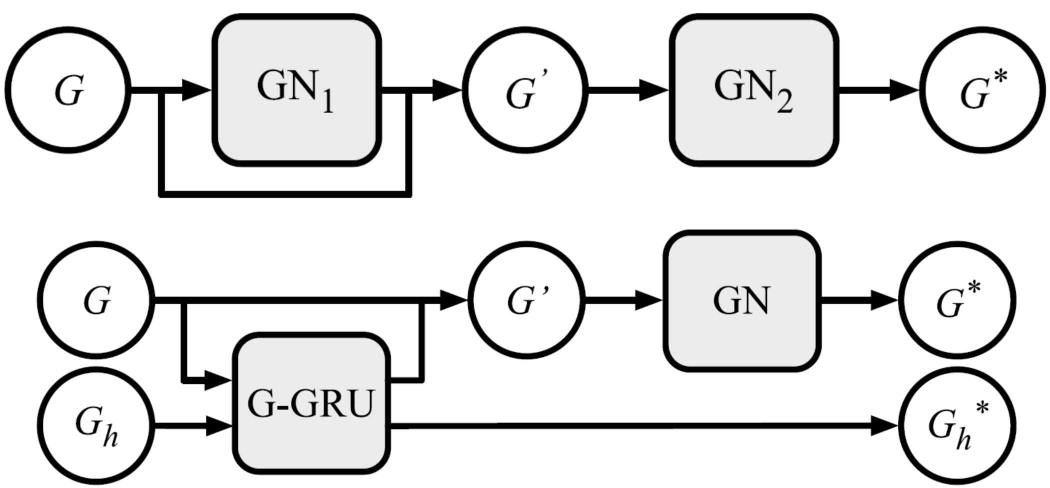
- Other aggregation functions can be used, e.g., average/max

Compositions of Graph Network Blocks

- GN blocks can be stacked together to build more power models



(a) Composition of GN blocks (b) Encode-process-decode (c) Recurrent GN architecture



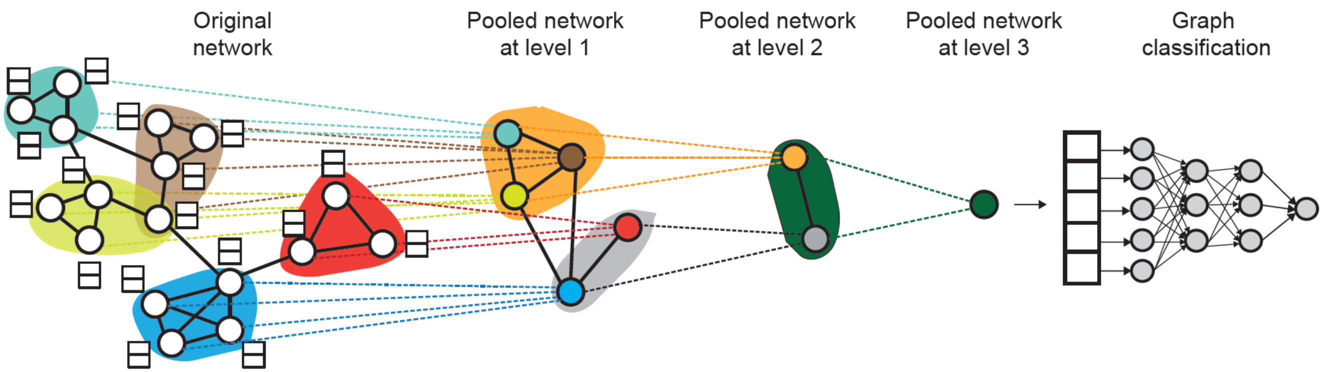
Outline

- Introduction to Graph Neural Networks
- **Hierarchical Pooling in GNN**
- Modeling Interacting Systems with GNN
- Conclusions and Further Discussion

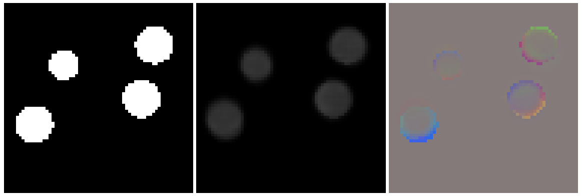
Neural Expectation Maximization. K Greff, et al. NIPS 2017
Relational Neural Expectation Maximization: Unsupervised Discovery of Objects and their Interactions. S Steenkiste, et al. ICLR 2018
Hierarchical Graph Representation Learning with Differentiable Pooling. R Ying, et al. Arxiv 2018
Flexible Neural Representation for Physics Prediction. D Mrowca, et al. Arxiv 2018

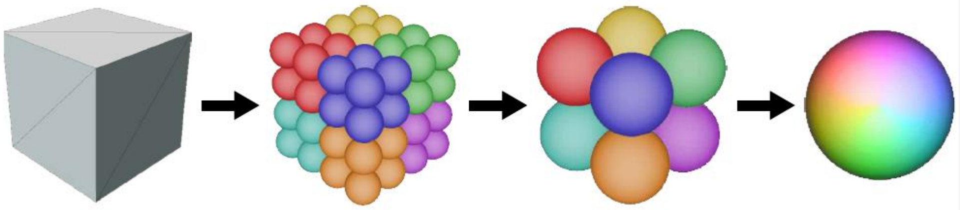
The Need for Hierarchical Pooling/Clustering

- Graph Networks are flat: the input and output have the same #vertices
- We would like to introduce some hierarchical structure to:
  - Aggregate node features in a hierarchical way



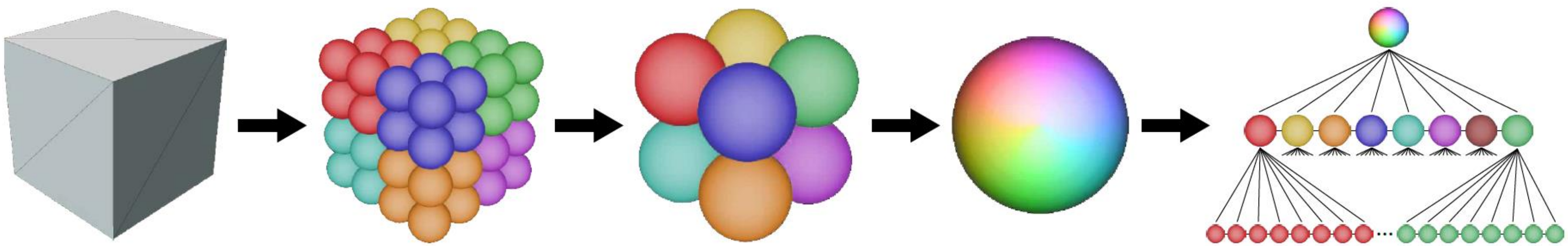
  - Model the interactions at a coarser level
  - Make the message passing more efficient



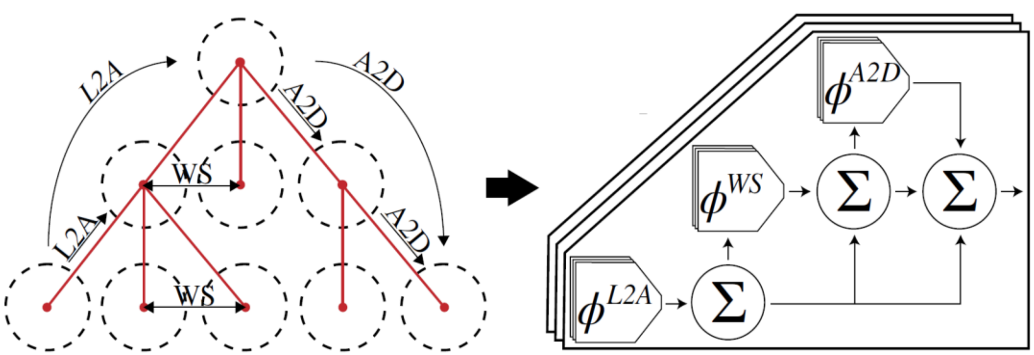


Hard Pooling

- Sometimes the hierarchical structure is fixed or relatively stable
- We can use hierarchical (graph) clustering algorithms to build a tree



- Hierarchical Graph Convolution

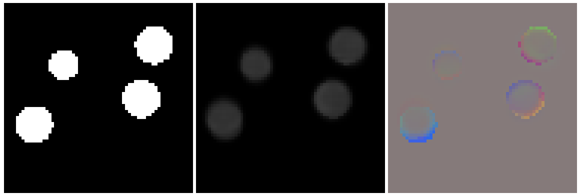


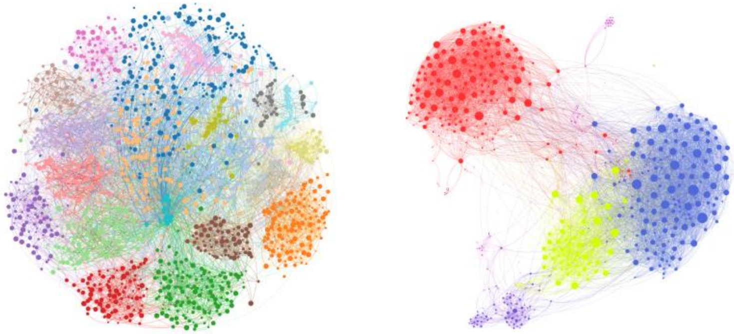
message passing:

(1) ϕ^{L2A}: leave $L \rightarrow$ ancestors A
(2) ϕ^{WS}: between siblings
(3) ϕ^{A2D}: $A \rightarrow$ descendants D

Soft Pooling

- In many cases, the hierarchical structure is not fixed
  - Evolving over time
  - Different hierarchies across graphs





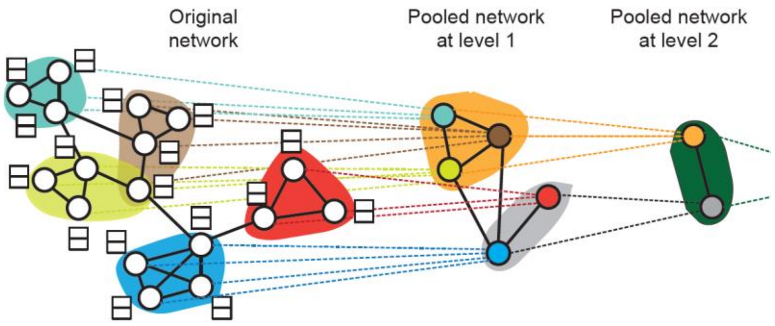
- Possible solution: learn the **assignment matrices**
  - Interestingly, an assignment matrix itself can be computed using a GNN

$$Z^{(l)} = \mathrm{GNN}_{l,\mathrm{embed}}\left(A^{(l)}, X^{(l)}\right) \in \mathbb{R}^{n_l \times d}$$

$$S^{(l)} = \mathrm{softmax}\left(\mathrm{GNN}_{l,\mathrm{pool}}\left(A^{(l)}, X^{(l)}\right)\right) \in \mathbb{R}^{n_l \times n_{l+1}}$$

$$X^{(l+1)} = {S^{(l)}}^T Z^{(l)} \in \mathbb{R}^{n_{l+1} \times d}$$

$$A^{(l+1)} = {S^{(l)}}^T A^{(l)} S^{(l)} \in \mathbb{R}^{n_{l+1} \times n_{l+1}}$$



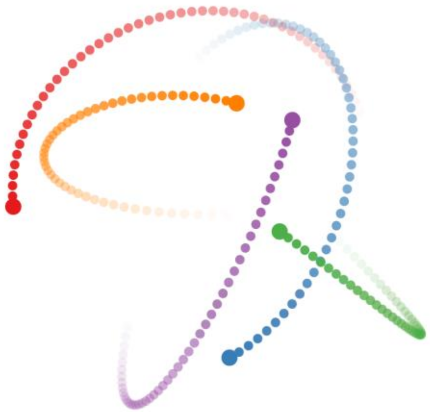
Outline

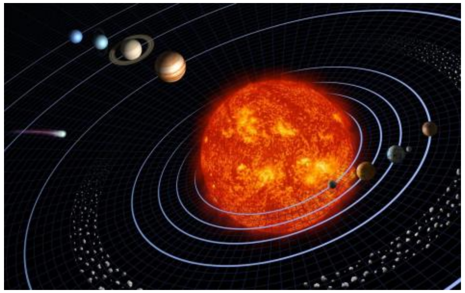
- Introduction to Graph Neural Networks
- Hierarchical Pooling in GNN
- **Modeling Interacting Systems with GNN**
- Conclusions and Further Discussion

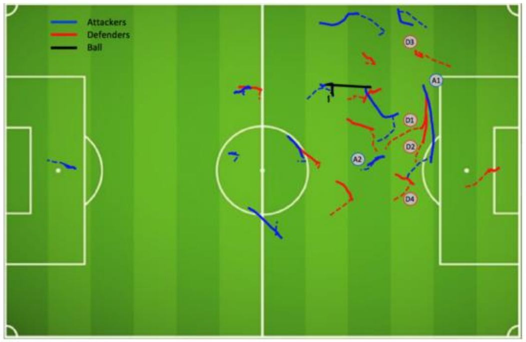
Relational Neural Expectation Maximization: Unsupervised Discovery of Objects and their Interactions. S Steenkiste, et al. ICLR 2018
Neural Relational Inference for Interacting Systems. T Kipf, M Welling. ICML 2018
Graph Networks as Learnable Physics Engines for Inference and Control. A Sanchez-Gonzalez, et al. ICML 2018

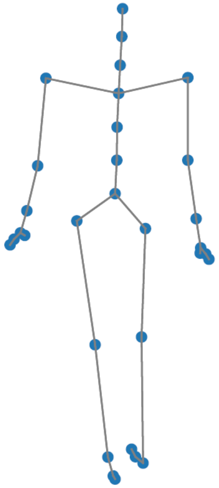
Interacting Dynamical Systems

- Most real-world dynamical systems can be decomposed into smaller dynamics that interact with each other









- Given sequential observations of individual objects, we would like to:
  - Infer the latent interaction structure
  - Learn the dynamical model of the interacting system
- So we could understand what happens and better predict the future

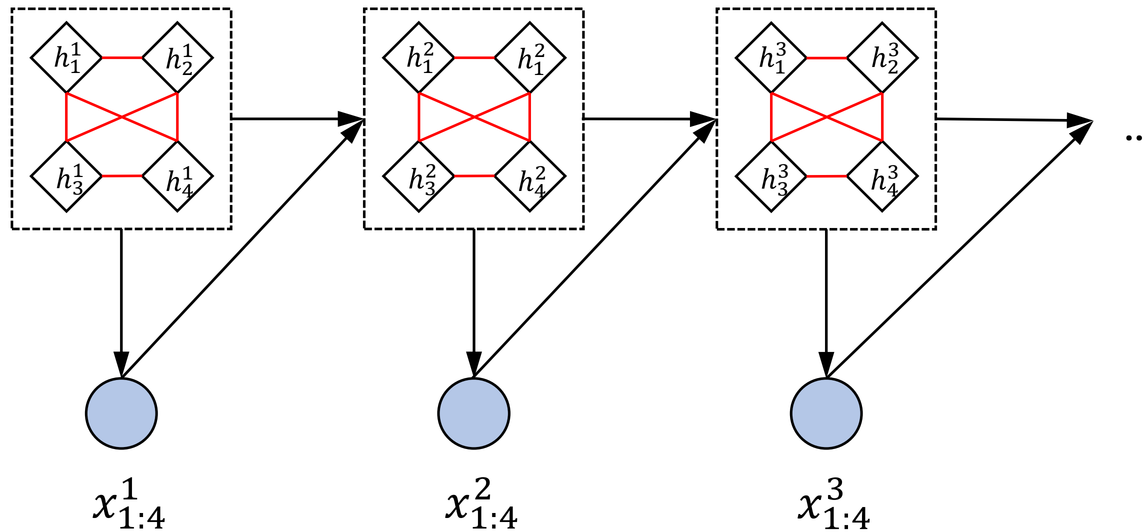
Modeling Interacting Systems with GNN

- Assume there are K objects and their corresponding sequential observations $\{x_k^{1:T}\}_{k=1}^{K}$

$$\tilde{h}_{1:K}^{t} = \mathrm{GNN}(h_{1:K}^{t})$$

$$h_k^{t+1} = \mathrm{RNN}(x_{1:K}^{t}, \tilde{h}_k^{t})$$

$$x_{1:K}^{t+1} \big| x_{1:K}^{t} \sim p(\cdot \,|f(h_{1:K}^{t+1}))$$



---

where $\tilde{h}_{1:K}^{t} = \mathrm{GNN}(h_{1:K}^{t})$ is implemented as follows:

$$\tilde{h}_k^t = [\hat{h}_k^t, E_k^t], \qquad \hat{h}_k^t = \mathrm{MLP}^{enc}(h_k^t), \qquad E_k^t = \sum\nolimits_{i \neq k} \alpha_{k,i}^t \cdot e_{k,i}^t$$

$$\alpha_{k,i}^t = \mathrm{MLP}^{att}(\xi_{k,i}^t), \qquad e_{k,i}^t = \mathrm{MLP}^{eff}(\xi_{k,i}^t), \qquad \xi_{k,i}^t = \mathrm{MLP}^{emb}([\hat{h}_k^t, \hat{h}_i^t])$$

Relational Neural Expectation Maximization: Unsupervised Discovery of Objects and their Interactions. S Steenkiste, et al. ICLR 2018

Building Latent Variable Models with GNN

- **NRI** introduces discrete random variables z_{ij} to represent edge types

$$\boldsymbol{z} \sim p(\boldsymbol{z})$$

VAE is used for model learning and inference

$$h_{1:K}^{t+1} = \mathrm{GNN}(h_{1:K}^t, x_{1:K}^t, \boldsymbol{z})$$

$$\mu_k^{t+1} = x_k^t + f_{\mathrm{out}}\left(h_k^{t+1}\right)$$

$$x_{1:K}^{t+1} | x_{1:K}^t, \boldsymbol{z} \sim \mathcal{N}(\mu_{1:K}^{t+1}, \sigma^2 \mathbf{I})$$

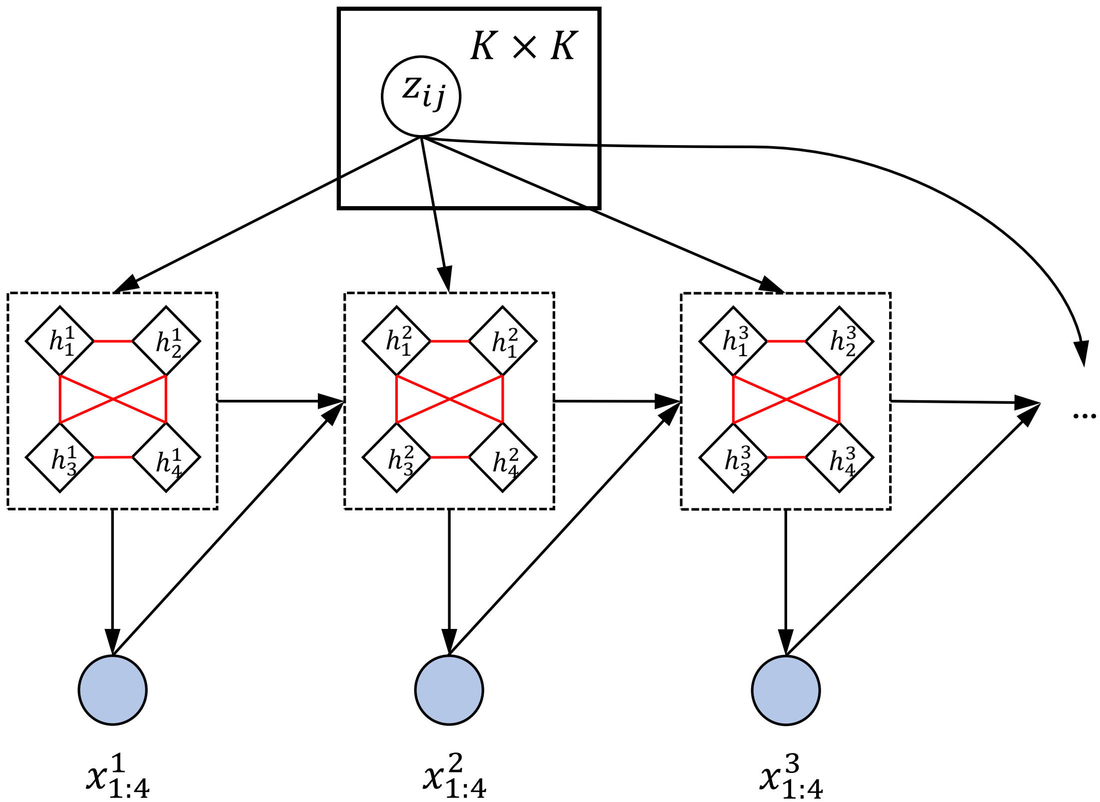
---

where $h_{1:K}^{t+1} = \mathrm{GNN}(h_{1:K}^t, x_{1:K}^t, \boldsymbol{z})$ is:

$$h_{(i,j)}^t = \sum_{m=1}^{M} z_{ij,m} f_e^m\left(\left[h_i^t, h_j^t\right]\right)$$

$$\mathrm{MSG}_k^t = \sum_{i \neq k} h_{(i,k)}^t$$

$$h_k^{t+1} = \mathrm{GRU}([\mathrm{MSG}_k^t, x_k^t], h_k^t)$$



Neural Relational Inference for Interacting Systems. T Kipf, M Welling. ICML 2018

Outline

- Introduction to Graph Neural Networks
- Hierarchical Pooling in GNN
- Modeling Interacting Systems with GNN
- **Conclusions and Further Discussion**

Conclusions and Further Discussion

- Graph Neural Networks are very likely to be ubiquitous building blocks for deep models
- The theoretical understanding of GNNs is still lacking
  - e.g., spectral graph theory for directed graphs, convergence of massage passing, effects of auxiliary edges/channels
- GNNs are useful for modeling interacting dynamical systems
  - There is still room for improvement, e.g., introducing hierarchies, using better models for individual objects

Questions?