Graph Neural Networks II

Neil Band

6.874 Deep Learning in the Life Sciences

Spring 2021

Sources / Further Reading

- Adapted from
  - Thomas Kipf's presentations (Cambridge CompBio, IPAM UCLA)
  - CS224W Machine Learning on Graphs by Jure Leskovec (Course @ Stanford)
  - Graph Neural Networks by Xavier Bresson (Guest lecture in Yann LeCun's NYU DL course)
  - Theoretical Foundations of Graph Neural Networks by Petar Veličković (@ Cambridge Computer Lab Seminar)
  - Junction Tree Variational Autoencoder (Wengong Jin, ICML 2018)
- Mining and Learning with Graphs at Scale (Google Graph Mining team @ NIPS 2020)
- Graph Representation Learning (Book by Will Hamilton, 2020)
- Thomas Kipf's thesis (Deep Learning with Graph Structured Representations, 2020)
- **Further reading**: Petar Veličković's thread of resources

Outline

1. **Refresher on graph neural nets (GNNs)**
2. **More problem domains**

Semi-supervised learning

Multi-relational data

Natural language processing

3. **Research frontiers**

Deep generative graph models

Latent graph inference

With applications in...

- Chemical synthesis
- Interacting systems (physical, multi-agent, biological)
- Causal inference
- Program induction

1 Refresher on GNNs

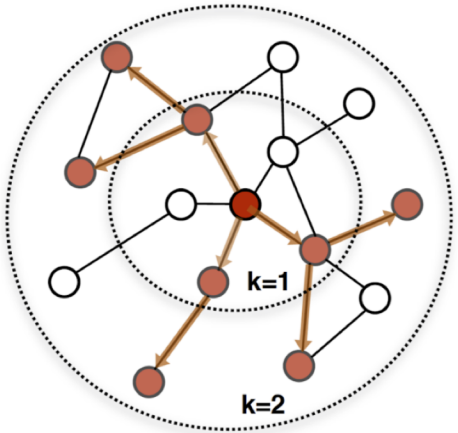
Main idea

Standard tasks

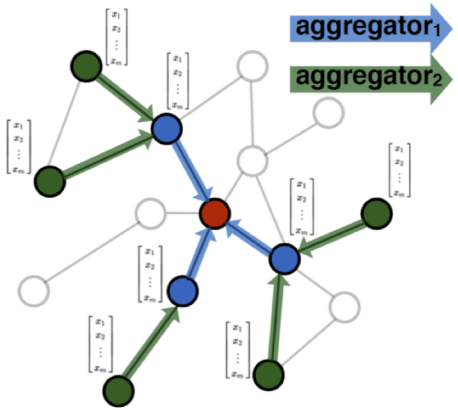
Core models

Aggregating neighbors

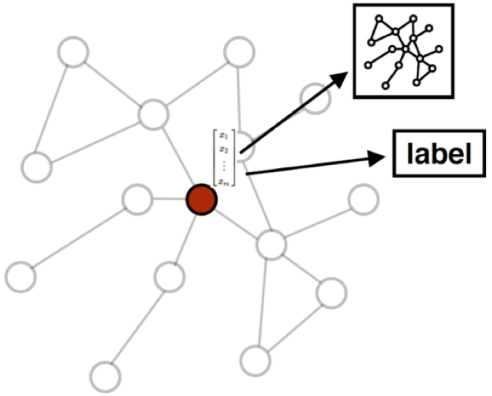
Idea: Node's neighborhood defines a computation graph



1. Sample neighborhood



2. Aggregate feature information from neighbors



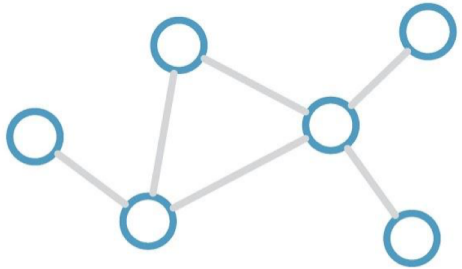
3. Predict graph context and label using aggregated information

Learn how to propagate information across the graph to compute node features

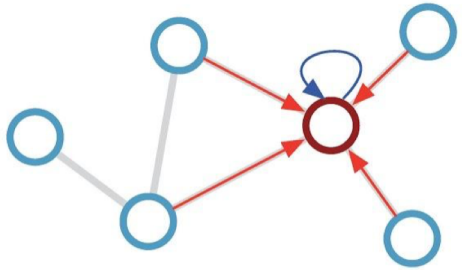
Graph convolutional networks (GCNs)

Kipf & Welling (ICLR 2017), related previous works: Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Consider this undirected graph:



Calculate update for node in red:



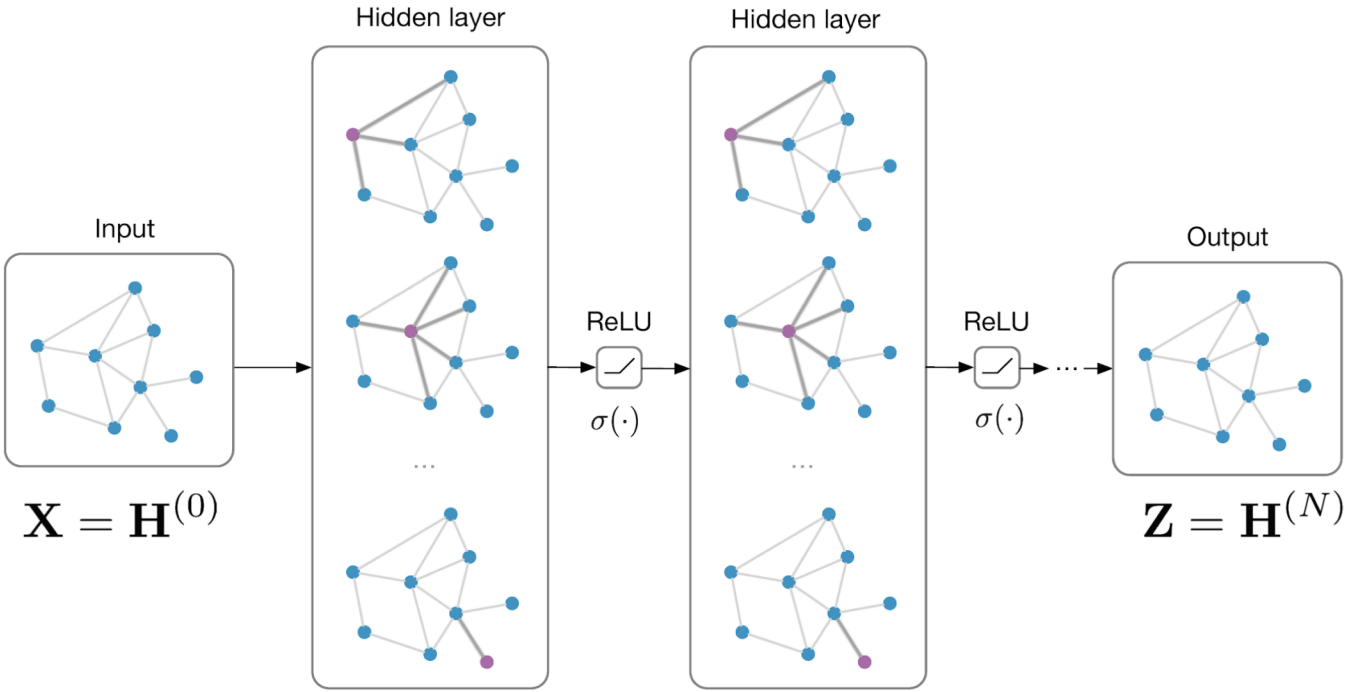
Update rule:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

$\mathcal{N}_i$: neighbor indices c_{ij} : norm. constant (fixed/trainable)

Credits to Thomas Kipf

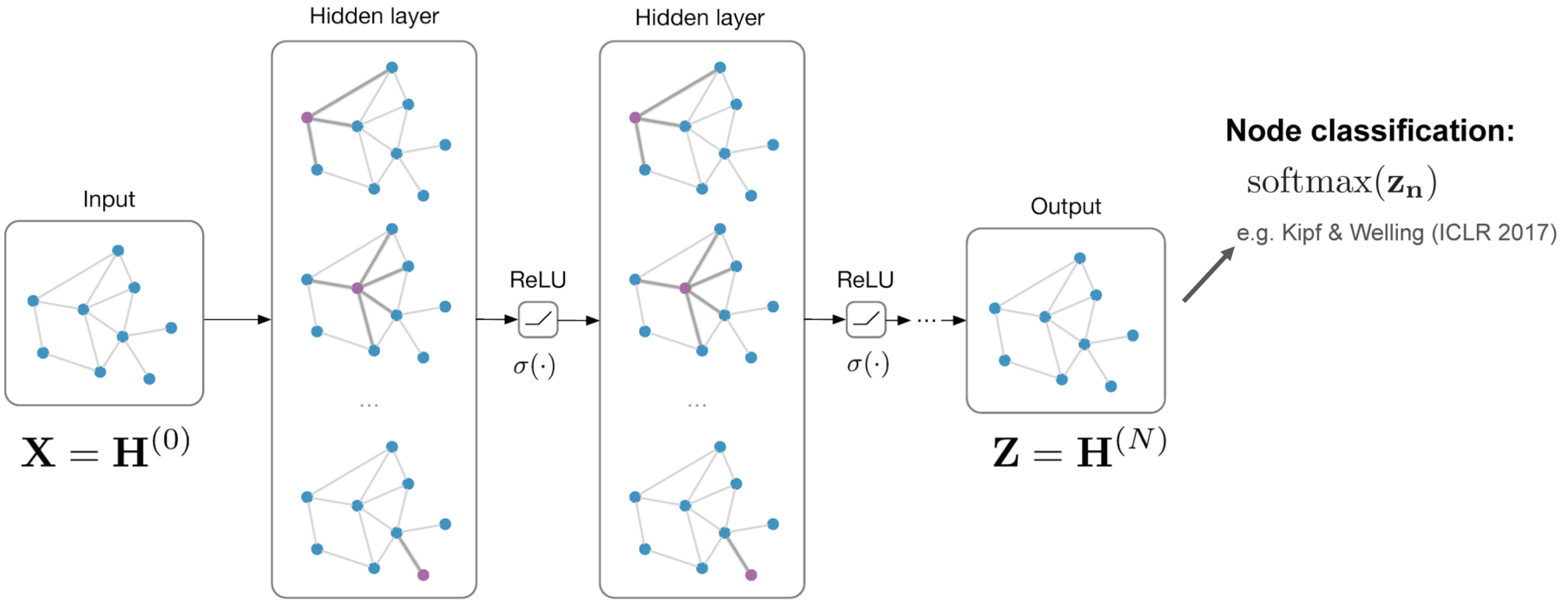
One fits all: Classification and link prediction with GNNs/GCNs



$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

Credits to Thomas Kipf

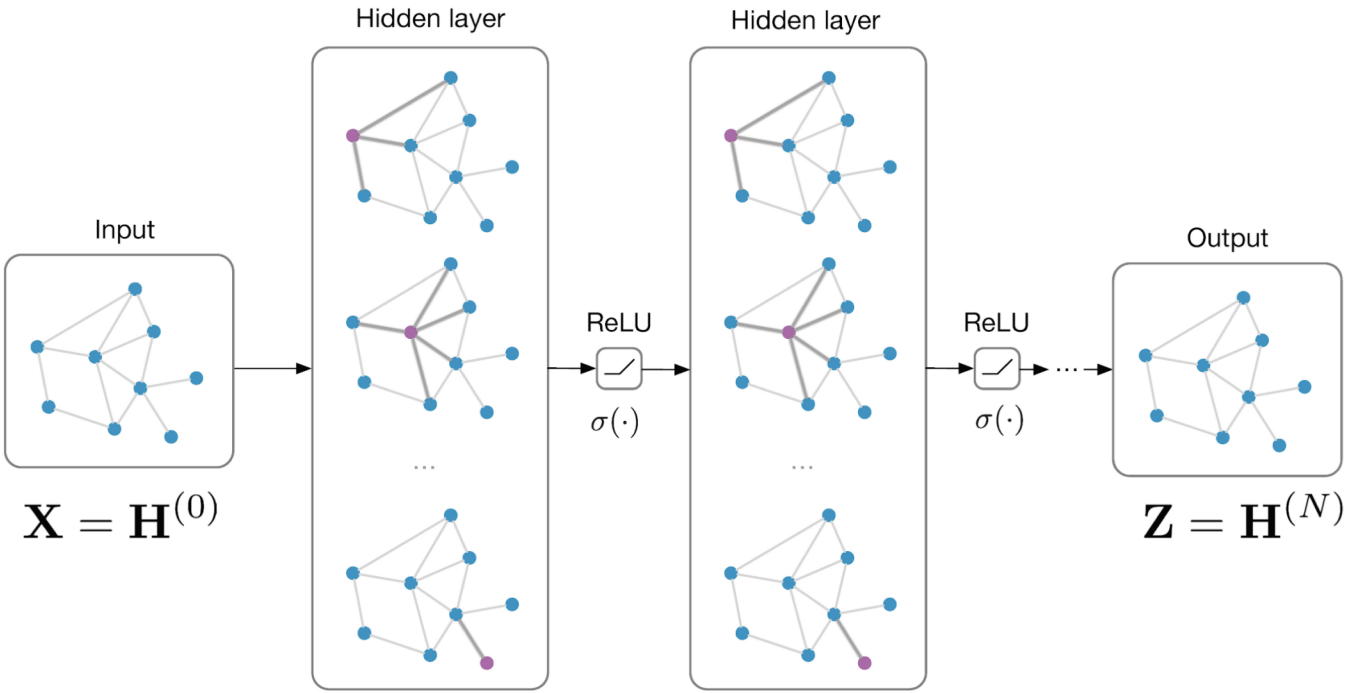
One fits all: Classification and link prediction with GNNs/GCNs



$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

Credits to Thomas Kipf

One fits all: Classification and link prediction with GNNs/GCNs



Node classification:

$\mathrm{softmax}(\mathbf{z_n})$

e.g. Kipf & Welling (ICLR 2017)

Graph classification:

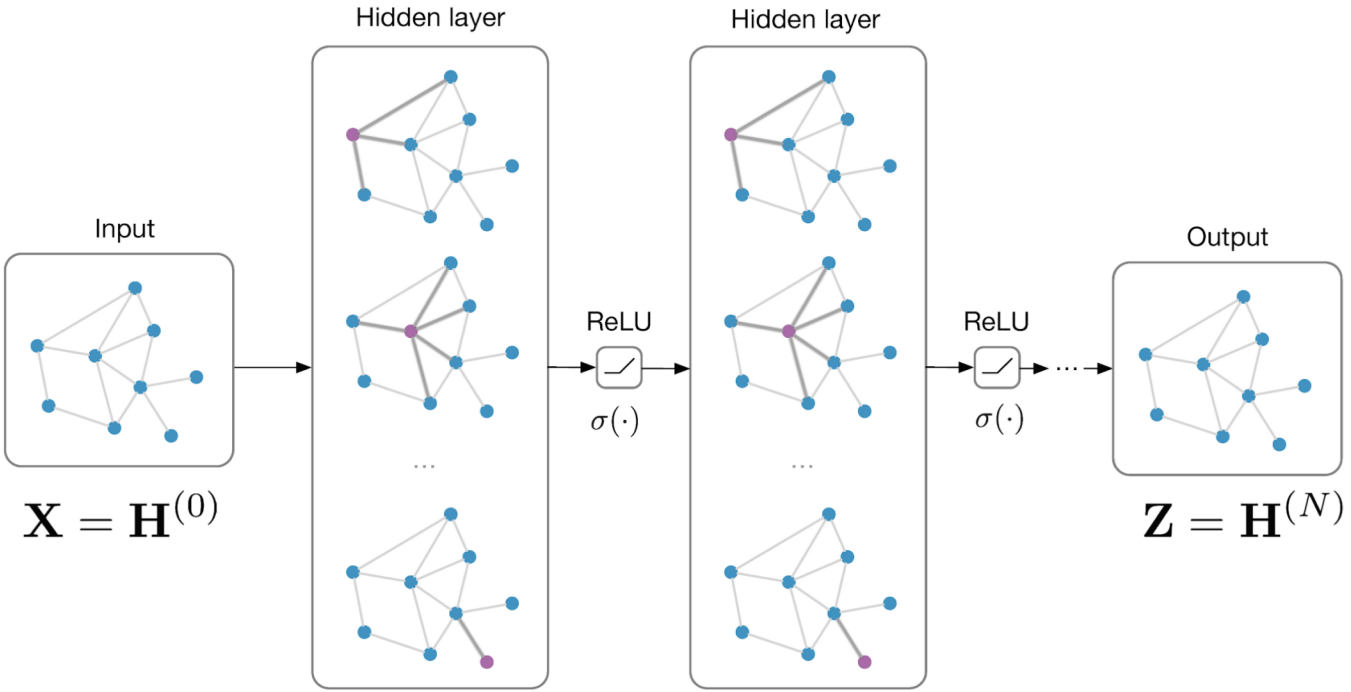
$\mathrm{softmax}(\sum_n \mathbf{z_n})$

e.g. Duvenaud et al. (NIPS 2015)

$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

Credits to Thomas Kipf

One fits all: Classification and link prediction with GNNs/GCNs



$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

Node classification:

$\mathrm{softmax}(\mathbf{z_n})$

e.g. Kipf & Welling (ICLR 2017)

Graph classification:

$\mathrm{softmax}(\sum_n \mathbf{z_n})$

e.g. Duvenaud et al. (NIPS 2015)

Link prediction:

$p(A_{ij}) = \sigma(\mathbf{z_i^T z_j})$

Kipf & Welling (NIPS BDL 2016)

"Graph Auto-Encoders"

Credits to Thomas Kipf

GCN classification on citation networks

Kipf & Welling, Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017

Input: Citation networks (nodes are papers, edges are citation links, optionally bag-of-words features on nodes)

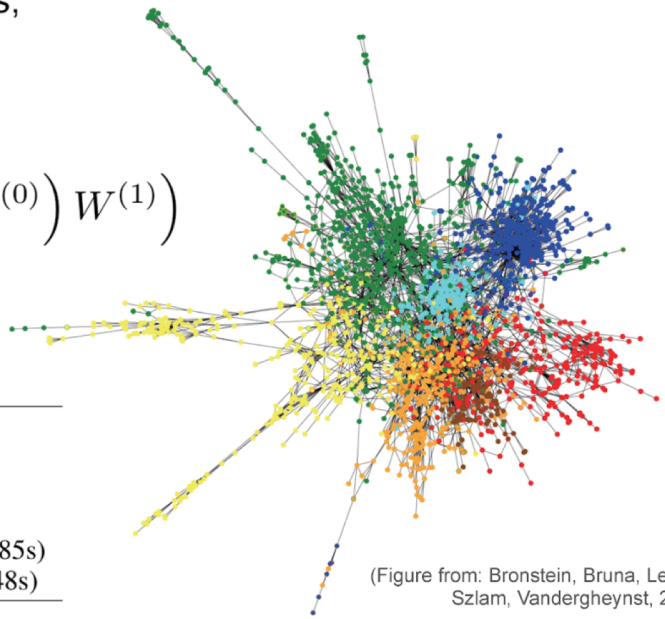
Target: Paper category (e.g. stat.ML, cs.LG, …)

Model: 2-layer GCN $Z = f(X, A) = \mathrm{softmax}\Big(\hat{A}\ \mathrm{ReLU}\Big(\hat{A}XW^{(0)}\Big)\ W^{(1)}\Big)$

Classification results (accuracy)

Method	Citeseer	Cora	Pubmed	NELL
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [24]	59.6	59.0	71.1	26.7
LP [27]	45.3	68.0	63.0	26.5
DeepWalk [18]	43.2	67.2	65.3	58.1
Planetoid* [25]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)

no input features

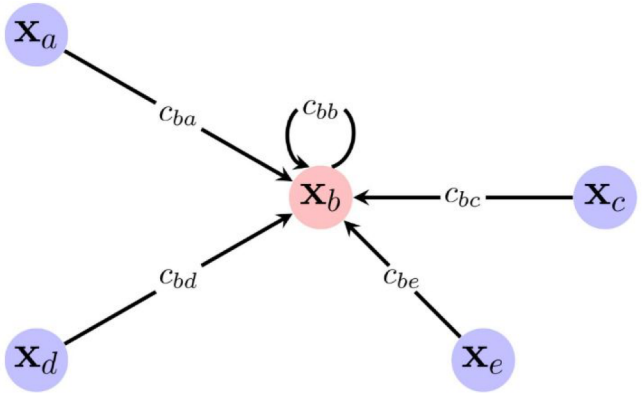


(Figure from: Bronstein, Bruna, LeCun, Szlam, Vandergheynst, 2016)

Credits to Thomas Kipf

Core models

Kipf & Welling (ICLR 2017); Kipf et al. (ICML 2018); Veličković et al. (ICLR 2018)



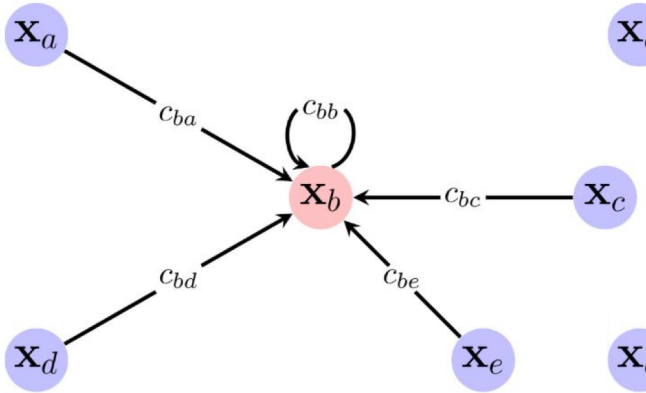
Convolutional

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij}\psi(\mathbf{x}_j)\right)$$

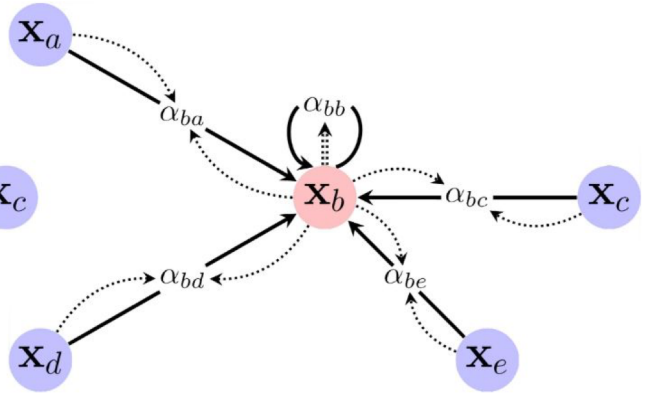
Credits to Petar Veličković

Core models

Kipf & Welling (ICLR 2017); Kipf et al. (ICML 2018); Veličković et al. (ICLR 2018)



Convolutional



Attentional

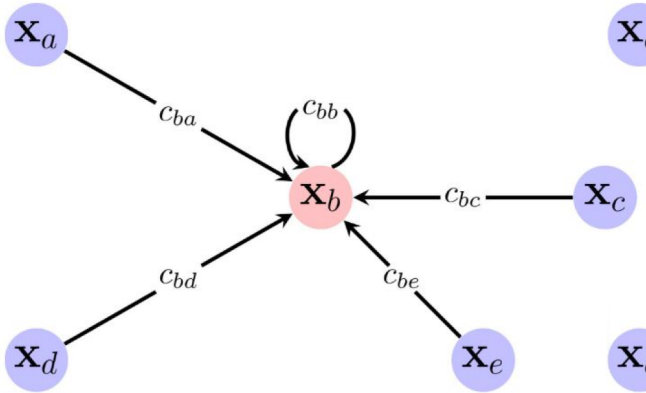
$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij}\psi(\mathbf{x}_j)\right)$$

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j)\psi(\mathbf{x}_j)\right)$$

Credits to Petar Veličković

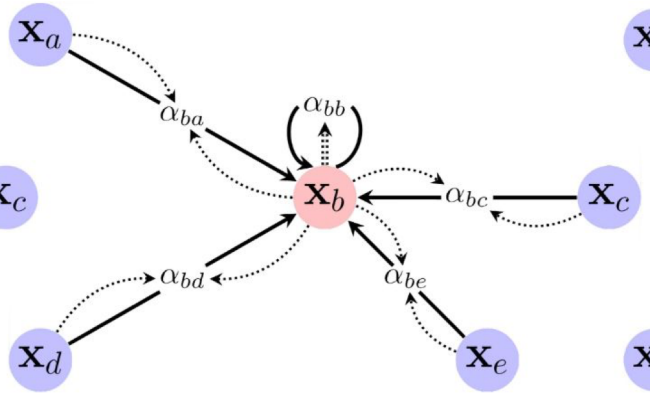
Core models

Kipf & Welling (ICLR 2017); Kipf et al. (ICML 2018); Veličković et al. (ICLR 2018)



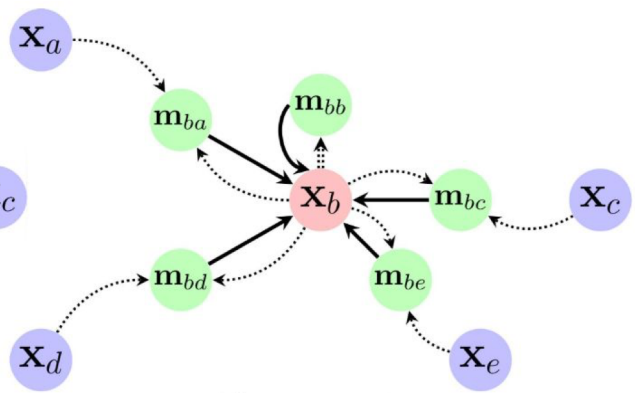
Convolutional

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij}\psi(\mathbf{x}_j)\right)$$



Attentional

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j)\psi(\mathbf{x}_j)\right)$$



Message-passing

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

Credits to Petar Veličković

2 More problem domains

Semi-supervised learning

Multi-relational data

Natural language processing

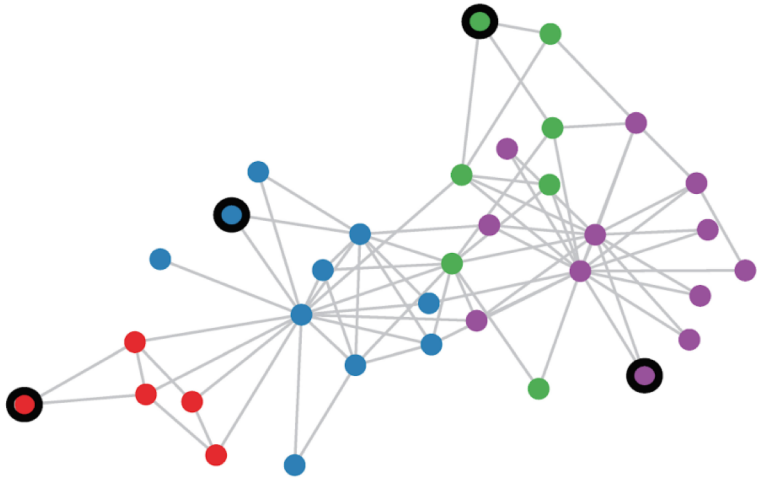
Semi-supervised classification on graphs

Setting:

Some nodes are labeled (black circle)
All other nodes are unlabeled

Task:

Predict node label of unlabeled nodes



Evaluate loss on labeled nodes only:

$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}$$

$\mathcal{Y}_L$ set of labeled node indices

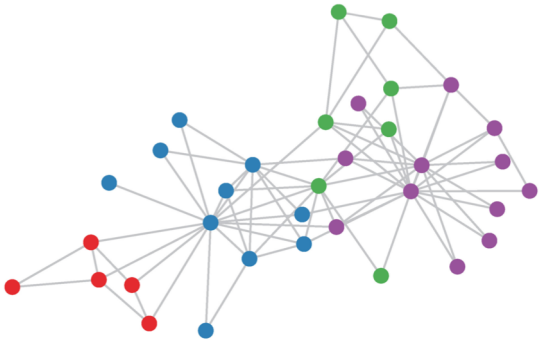
$\mathbf{Y}$ label matrix

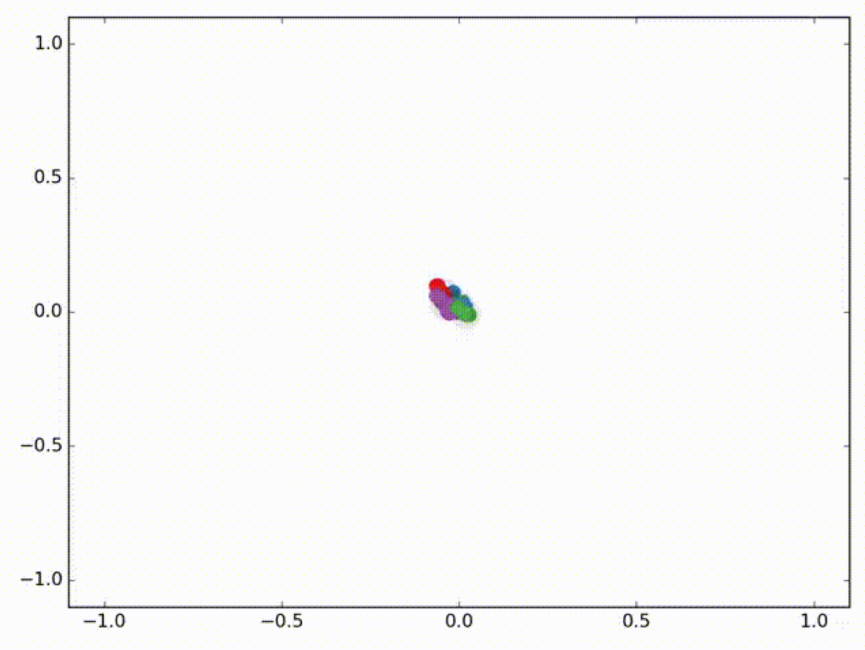
$\mathbf{Z}$ GCN output (after softmax)

Credits to Thomas Kipf

Toy example (semi-supervised learning)

from tkipf.github.io/graph-convolutional-networks





Latent space dynamics for 300 training iterations.
Labeled nodes are highlighted.

GCN model manages to linearly separate classes with only **1 training example per class, no node features!**

Credits to Thomas Kipf

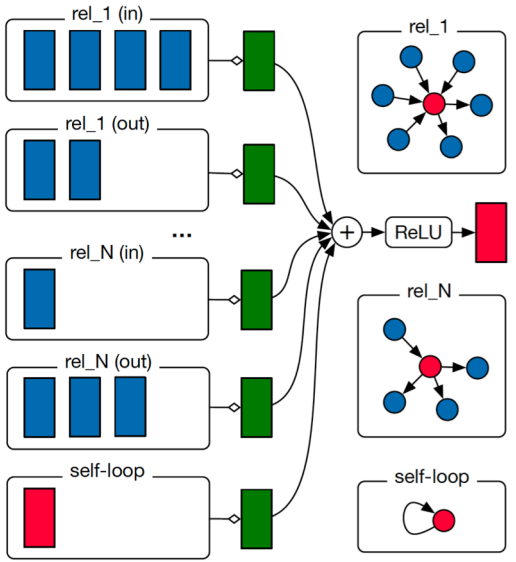
MoNet & Relational GCN for modeling (multi-)relational data

Monti et al. (CVPR 2017), Schlichtkrull & Kipf et al. (ESWC 2018)

$$\mathbf{h}_i' = \sigma\left(\sum_{r=1}^{R}\sum_{j\in\mathcal{N}_i} \alpha_{ij}^{r}\mathbf{W}_r\mathbf{h}_j\right)$$

α_{ij}^{r} **based on:**

- Edge type (Relational GCN)
- Auxiliary features (MoNet), e.g. node degree

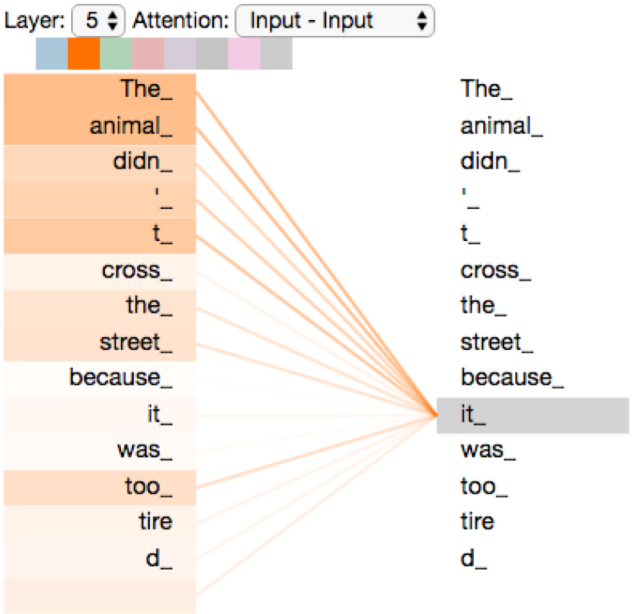


Relational GCN update rule

Credits to Thomas Kipf

Connection to NLP: Transformers

A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A Gomez, L Kaiser, I Polosukhin, Attention is all you need (2017)



Words in a sequence interact

- Define a graph over them

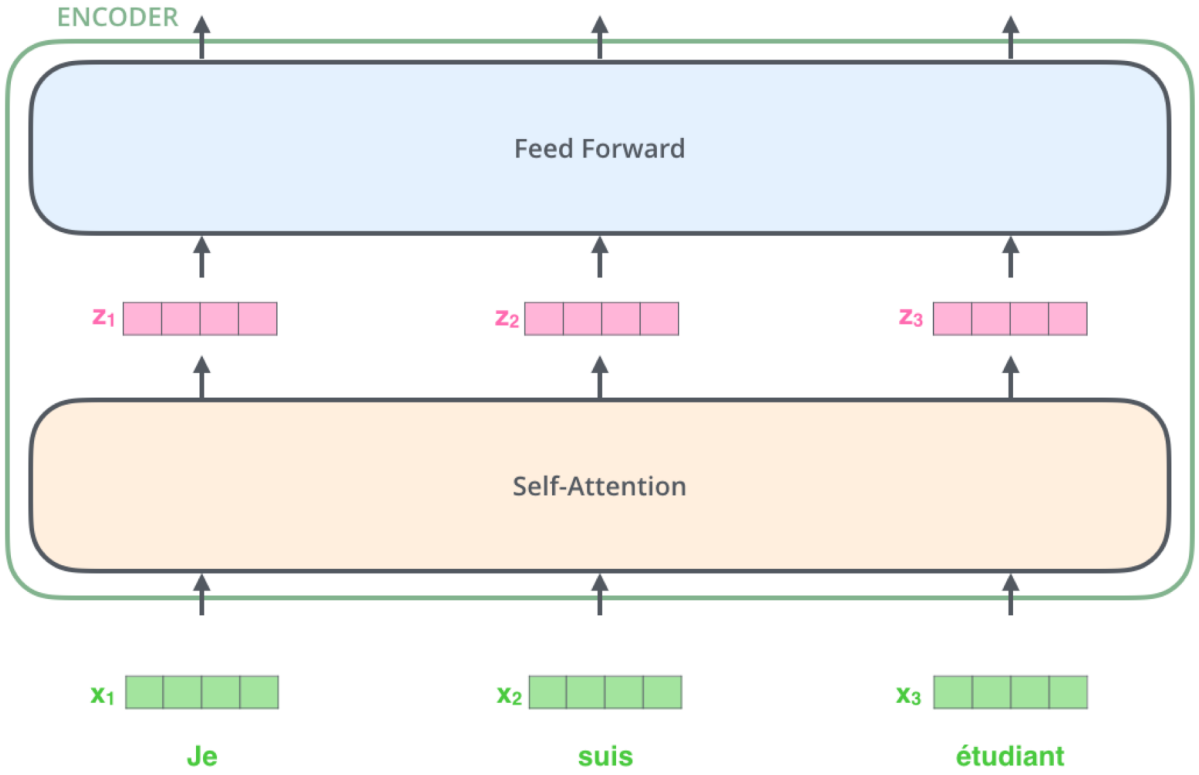
Why do we care about this connection?

- Cross-pollination (e.g., Strategies for Pre-training Graph Neural Networks, Hu et al. 2020)
- Fast and optimized libraries
- New way to consider (the validity of) edges in GNN datasets

Credits to Jay Alammar

Transformer “update”

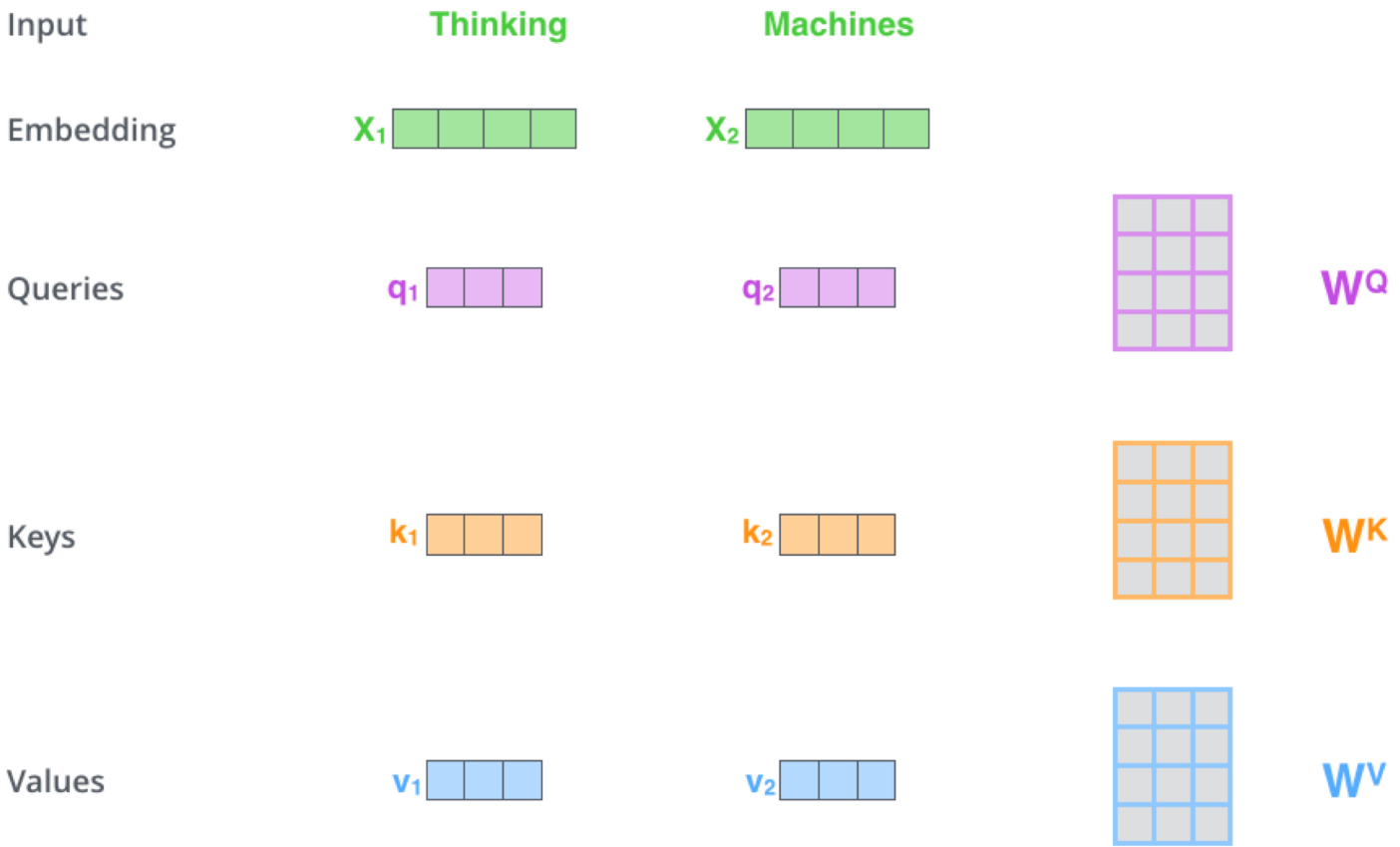
A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A Gomez, L Kaiser, I Polosukhin, Attention is all you need (2017)



Credits to Jay Alammar

Transformer “update”

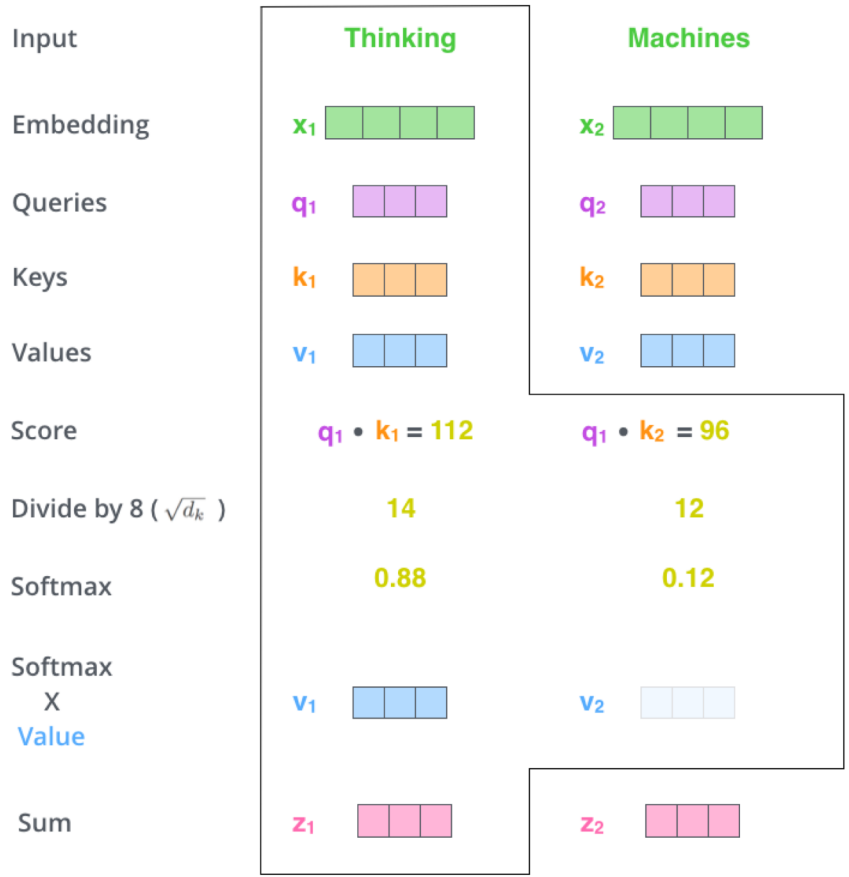
A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A Gomez, L Kaiser, I Polosukhin, Attention is all you need (2017)



Credits to Jay Alammar

Transformer “update”

A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A Gomez, L Kaiser, I Polosukhin, Attention is all you need (2017)

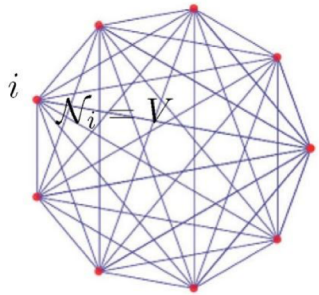


Credits to Jay Alammar

Graph Transformers (Li et al. 2018)

A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A Gomez, L Kaiser, I Polosukhin, Attention is all you need (2017)

- **We can frame transformers as a special case of GCNs when the graph is fully connected.**
- **The neighborhood $\mathcal{N}_i$ is the whole graph.**



Fully connected graph

$$h_i^{\ell+1} = W^\ell \ \text{Concat}_{k=1}^{K}\Big(\sum_{j \in \mathcal{N}_i} e_{ij}^{k,\ell} \ V^{k,\ell} \ h_j^\ell \Big)$$

$$e_{ij}^{k,\ell} = \text{Softmax}_{\mathcal{N}_i}(\hat{e}_{ij}^{k,\ell}) = \frac{\exp(\hat{e}_{ij}^{k,\ell})}{\sum_{j' \in \mathcal{N}_i} \exp(\hat{e}_{ij'}^{k,\ell})}$$

$$\hat{e}_{ij}^{k,\ell} = (Q^{\ell,k} h_i^\ell)^T (K^{\ell,k} h_j^\ell)$$

$$\mathcal{N}_i = V \Rightarrow$$

$$\underset{n \times d}{h^{\ell+1}} = \text{Concat}_{k=1}^{K}\Big(\text{Softmax}(Q^\ell K^{\ell T}) V^\ell\Big) \overset{d \times d}{W^\ell}$$

$$\underset{n \times \frac{d}{K}}{Q^\ell} = \underset{n \times d}{h^\ell} \ \underset{d \times \frac{d}{K}}{W_Q^\ell}$$

$$K^\ell = h^\ell \ W_K^\ell$$

$$V^\ell = h^\ell \ W_V^\ell$$

$$\underbrace{\underbrace{n \times \frac{d}{K} \quad \frac{d}{K} \times n}_{n \times n} \quad n \times \frac{d}{K}}_{n \times \frac{d}{K}}$$

Credits to Xavier Bresson

3 Research frontiers

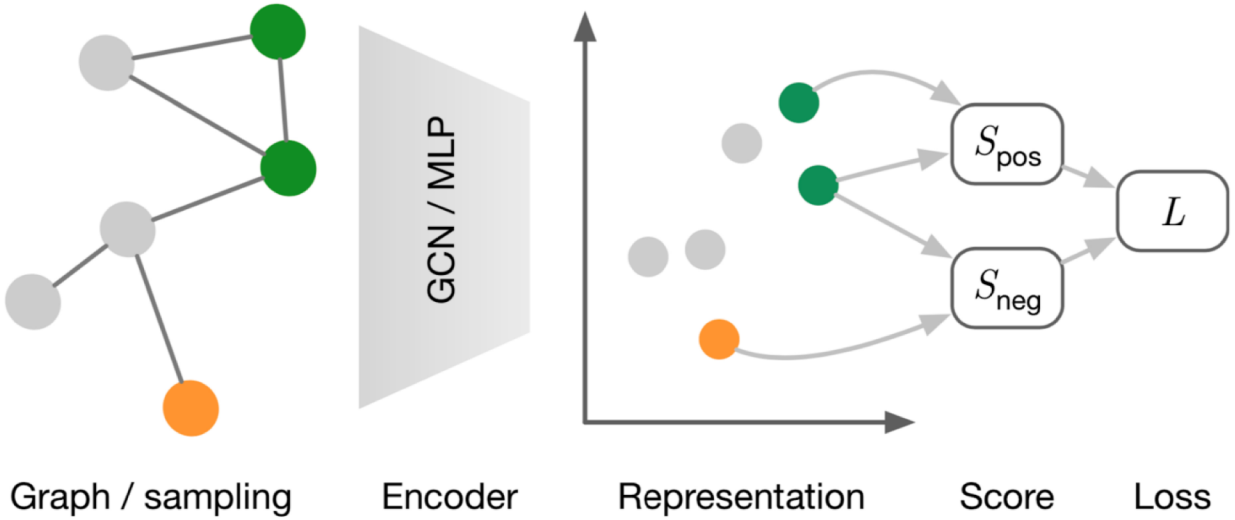
Deep generative graph models

Latent graph inference

Unsupervised learning with GNNs

Objective: Learn node embeddings for downstream tasks

Most approaches follow a contrastive learning approach:

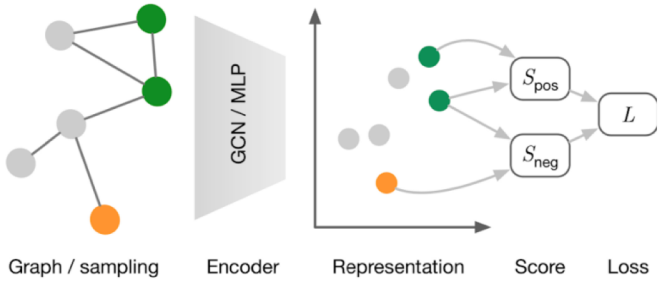


Credits to Thomas Kipf

Unsupervised learning with GNNs

Objective: Learn node embeddings for downstream tasks

Most approaches follow a contrastive learning approach:

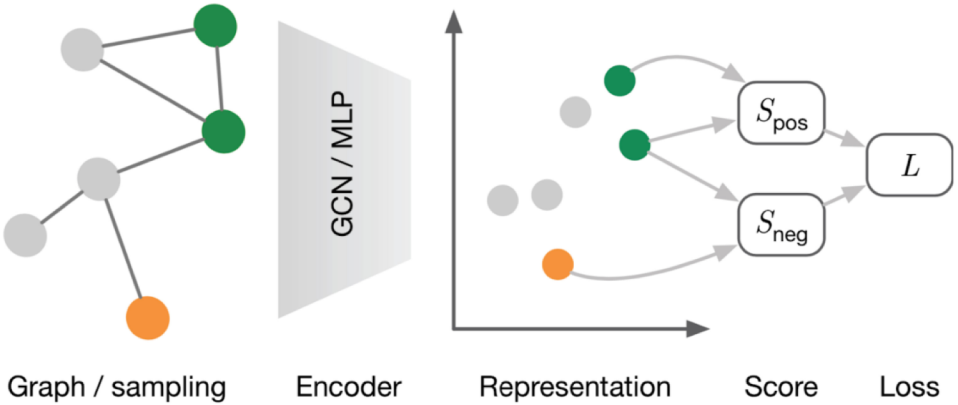


- **Sampling strategies**
  e.g. positive: neighbor; negative: random node
- **Encoder variants**
  GCN, GAT, MLP, Lookup table
- **Node representations**
  Geometry of latent space, distributional embeddings (e.g. Hyperbolic GCNN, Chami et al. 2019)
- **Score functions**
  Inner/bilinear product, local vs. global (e.g. Deep Graph Infomax, Velickovic et al. 2019)
- **Loss**
  (Cross-entropy, MSE, exponential)

Credits to Thomas Kipf

Unsupervised learning takeaways

A Modular Framework for Unsupervised Graph Representation Learning, Daza & Kipf (WIP)



- Graph-based encoders often improve performance
- Neighbor-based scoring (GAE) effective for both link prediction & node classification
- Local-global scoring (DGI) especially effective for node classification
- Ideal node representation (distributional, hyperspherical, etc.) heavily data-dependent

Credits to Thomas Kipf

Likelihood-based (deep) graph generation

Likelihood-based:

- we have some ground truth graphs
- define likelihood as how well a generated graph matches a ground truth graph

Credits to Thomas Kipf

Likelihood-based (deep) graph generation

Version 1: Generate graph (or predict new links) between known entities

Graph-based autoencoders:

- Encoder: GNN/GCN
- Decoder: Pairwise scoring function

$$p(A_{ij}) = f(\mathbf{z}_i, \mathbf{z}_j)$$

e.g. $p(A_{ij}) = \sigma(\mathbf{z}_i^T \mathbf{z}_j)$

Likelihood-based:

- we have some ground truth graphs
- define likelihood as how well a generated graph matches a ground truth graph

Credits to Thomas Kipf

Likelihood-based (deep) graph generation

Version 1: Generate graph (or predict new links) between known entities

Graph-based autoencoders:

- Encoder: GNN/GCN
- Decoder: Pairwise scoring function

$$p(A_{ij}) = f(\mathbf{z}_i, \mathbf{z}_j)$$

e.g. $p(A_{ij}) = \sigma(\mathbf{z}_i^T \mathbf{z}_j)$

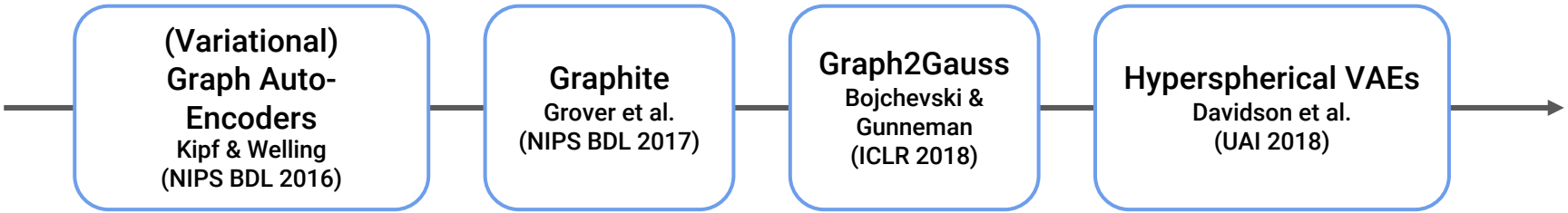
Likelihood-based:

- we have some ground truth graphs
- define likelihood as how well a generated graph matches a ground truth graph

$$p(\mathbf{A} \mid \mathbf{Z}) = \prod_{i=1}^{N} \prod_{j=1}^{N} p(A_{ij} \mid \mathbf{z}_i, \mathbf{z}_j) \ , \quad \text{with} \quad p(A_{ij} = 1 \mid \mathbf{z}_i, \mathbf{z}_j) = \sigma(\mathbf{z}_i^\top \mathbf{z}_j)$$

VGAE generative model (with ELBO loss)

(Incomplete) History:



Credits to Thomas Kipf

Likelihood-based (deep) graph generation

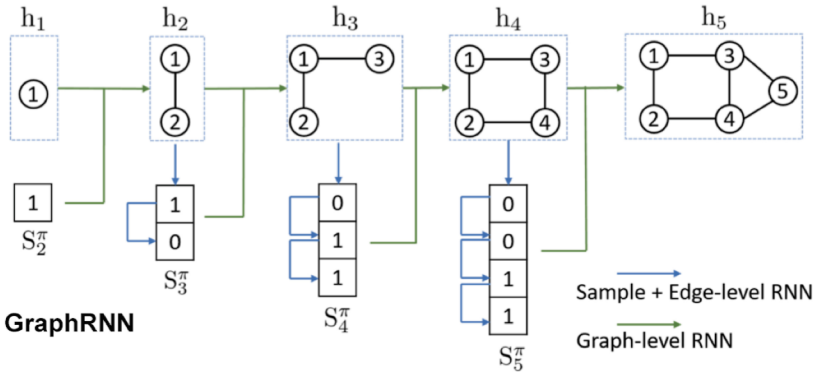
Version 2: Generate graphs from scratch (single embedding vector)

Credits to Thomas Kipf

Likelihood-based (deep) graph generation

Version 2: Generate graphs from scratch (single embedding vector)

Sequentially:

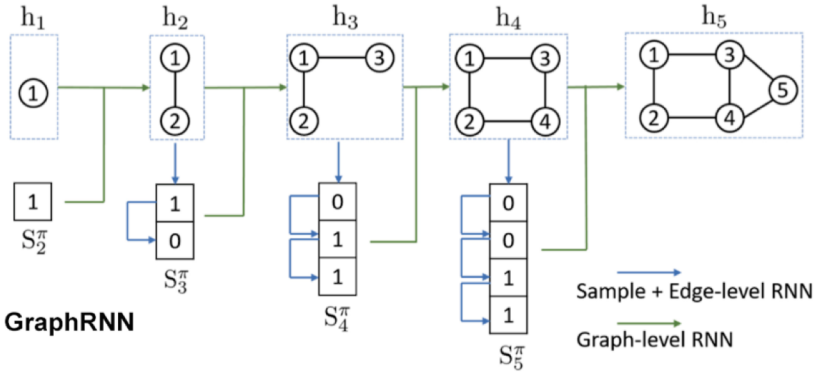


Credits to Thomas Kipf

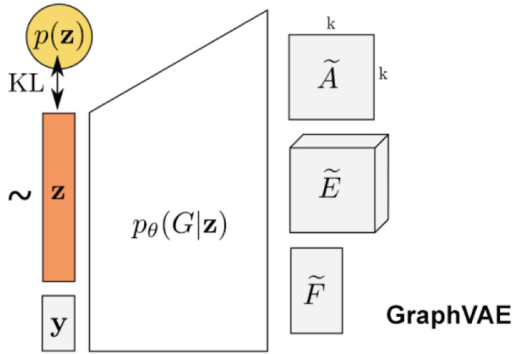
Likelihood-based (deep) graph generation

Version 2: Generate graphs from scratch (single embedding vector)

Sequentially:



Or in a single step:

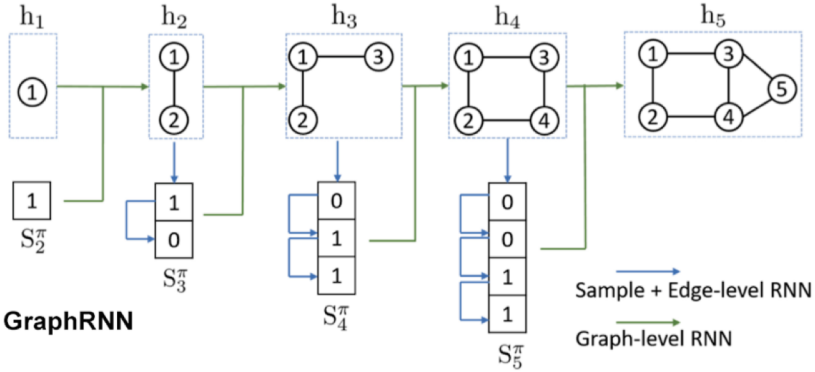


Credits to Thomas Kipf

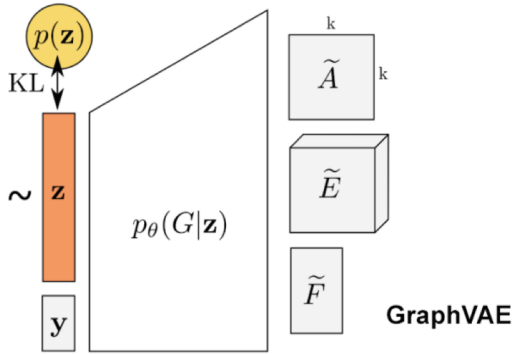
Likelihood-based (deep) graph generation

Version 2: Generate graphs from scratch (single embedding vector)

Sequentially:



Or in a single step:



Learning Graphical State Transitions
Johnson
(ICLR 2017)

Deep Generative Models of Graphs
Li et al.
(arXiv 2018)

GraphVAE
Simonovsky et al.
(arXiv 2018)

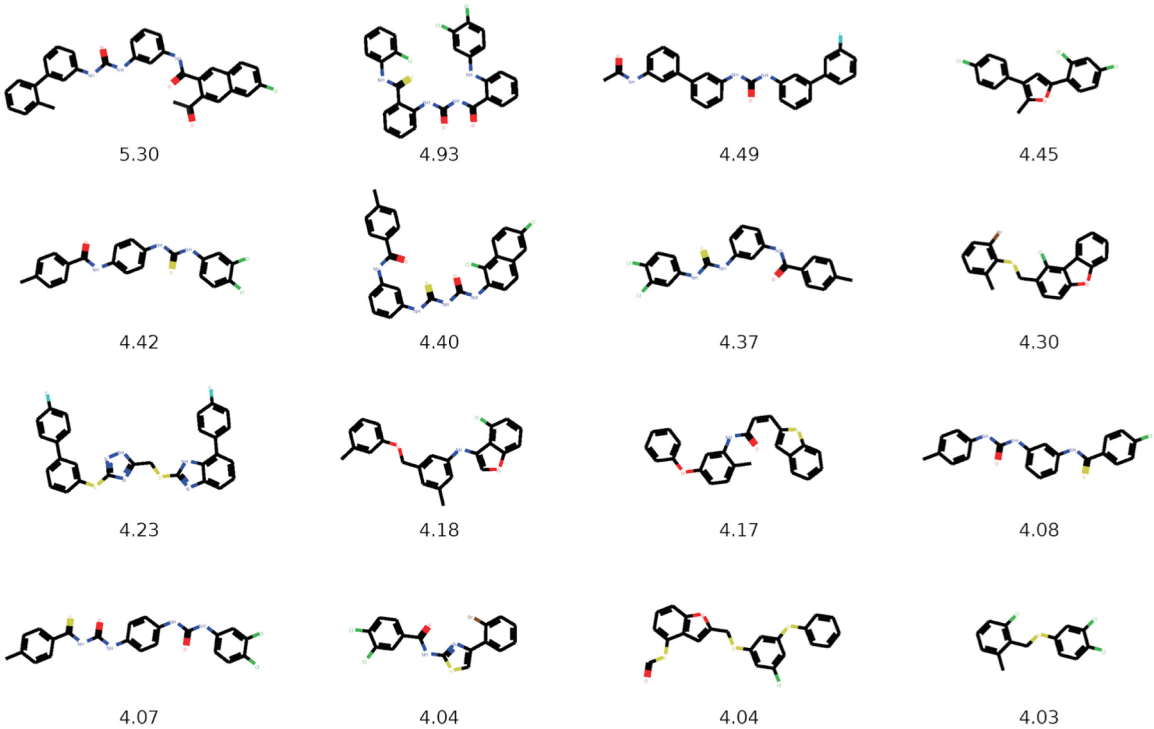
GraphRNN
You et al.
(ICML 2018)

Credits to Thomas Kipf

Graph generation for drug discovery

Junction Tree Variational Autoencoder, Jin et al. (ICML 2018)

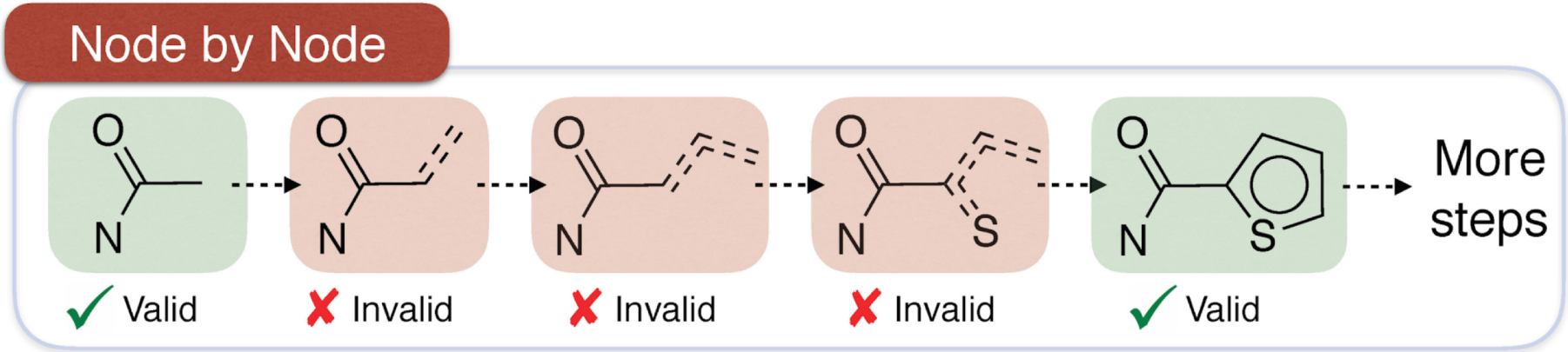
Aim: generate molecules with high potency



Credits to Wengong Jin

How should we decode the graph?

Junction Tree Variational Autoencoder, Jin et al. (ICML 2018)

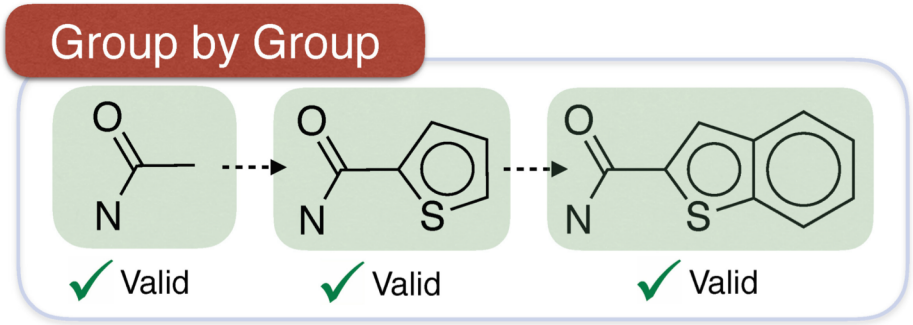


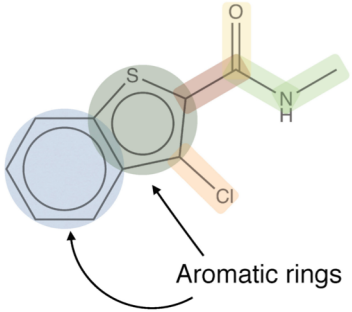
- Not every graph is chemically valid
- Invalid intermediate states → hard to validate
- Many intermediate states (i.e. long sequences) → difficult to train (Li et al. 2018)

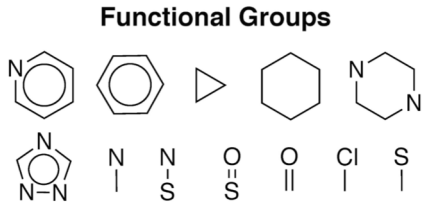
Credits to Wengong Jin

How should we decode the graph?

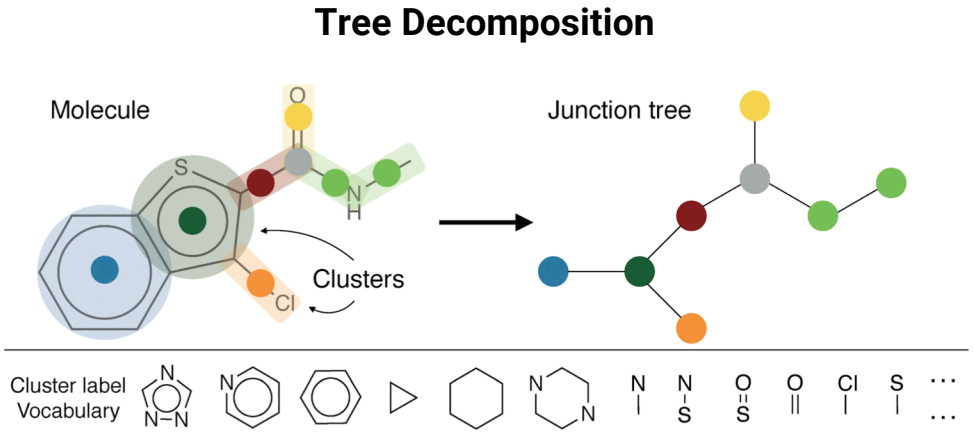
Junction Tree Variational Autoencoder, Jin et al. (ICML 2018)







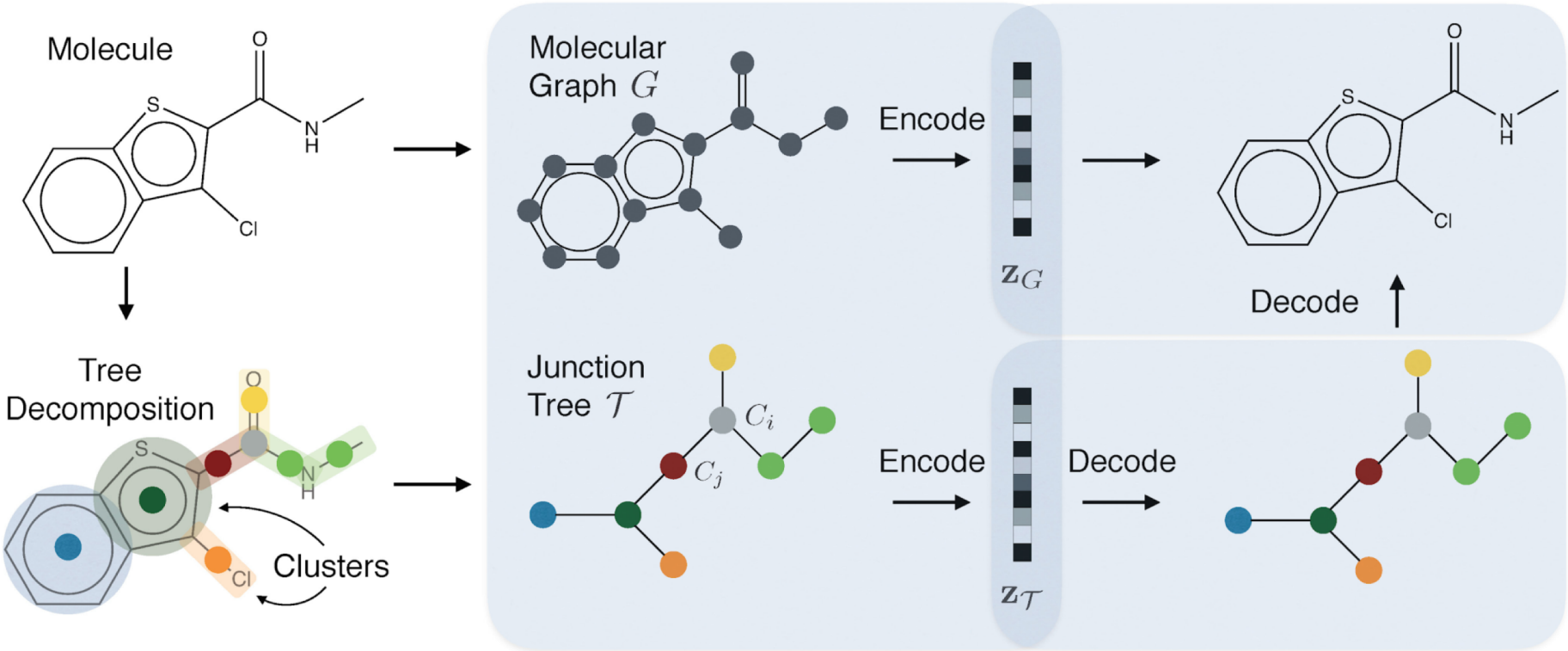
- Shorter action sequence
- Easy to check validity as we construct
- Vocabulary size: ~800 for 250K molecules



Credits to Wengong Jin

High-level approach

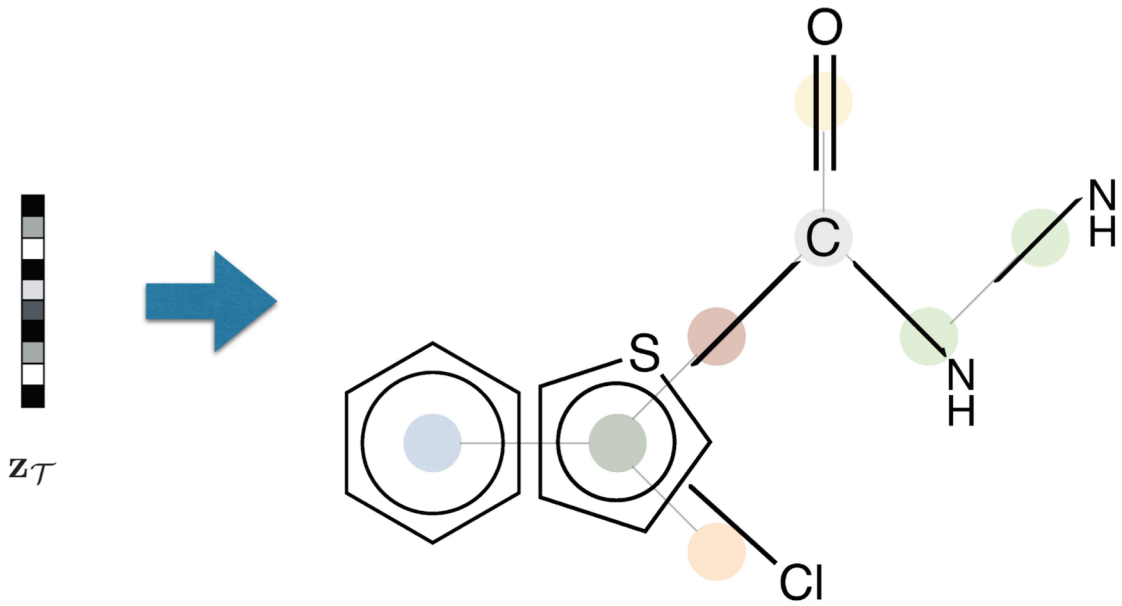
Junction Tree Variational Autoencoder, Jin et al. (ICML 2018)



Credits to Wengong Jin

Focus on a cool part: tree decoding

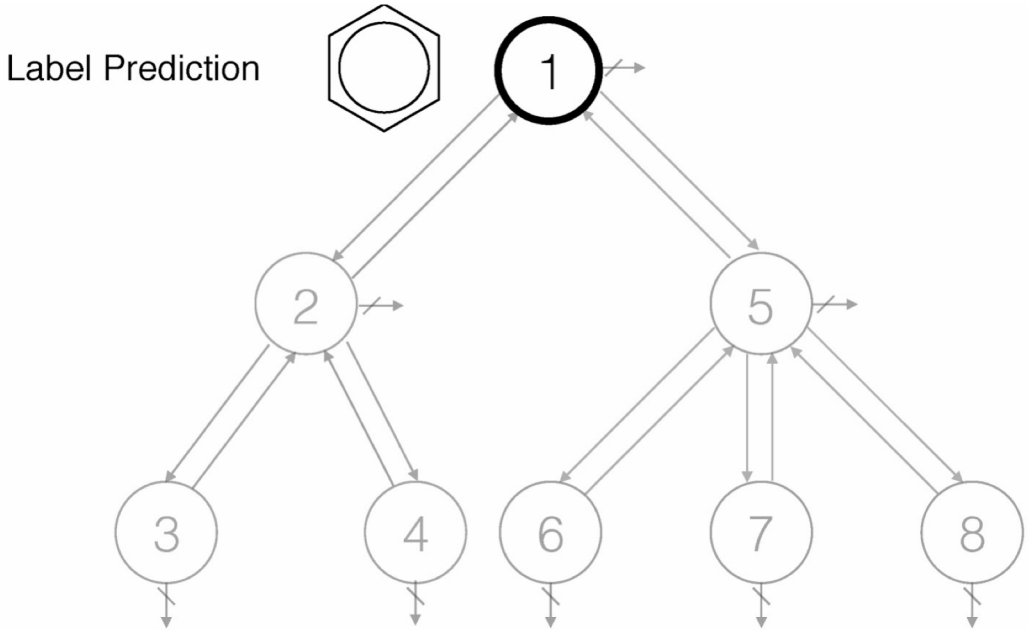
Junction Tree Variational Autoencoder, Jin et al. (ICML 2018)



Credits to Wengong Jin

Tree decoding

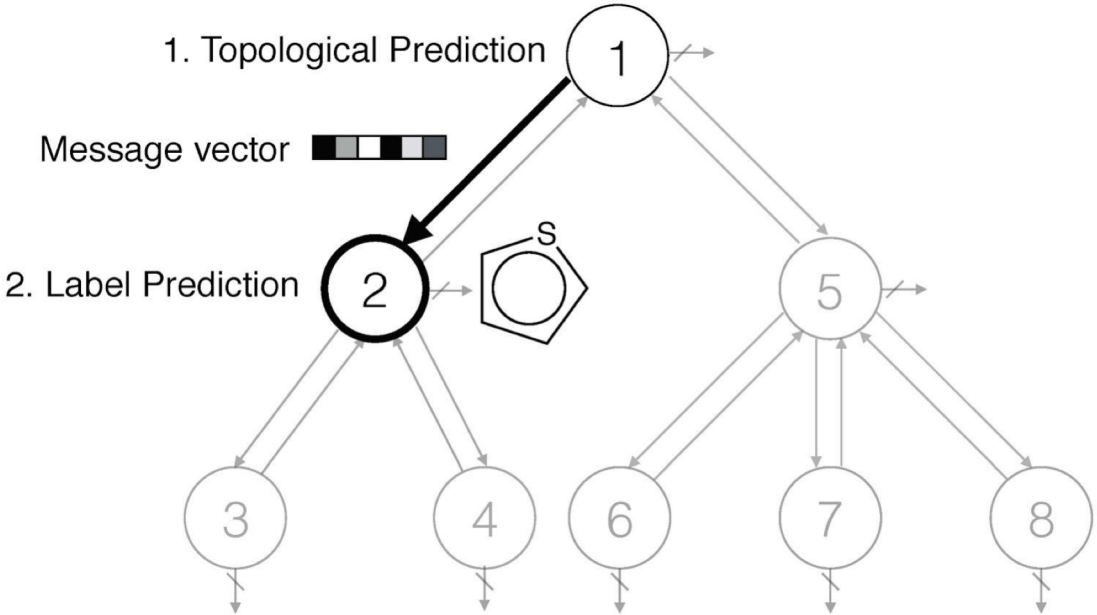
Junction Tree Variational Autoencoder, Jin et al. (ICML 2018); Tree-Structured Decoding, Alvarez-Melis & Jaakkola (ICLR 2017)



Credits to Wengong Jin

Tree decoding

Junction Tree Variational Autoencoder, Jin et al. (ICML 2018); Tree-Structured Decoding, Alvarez-Melis & Jaakkola (ICLR 2017)



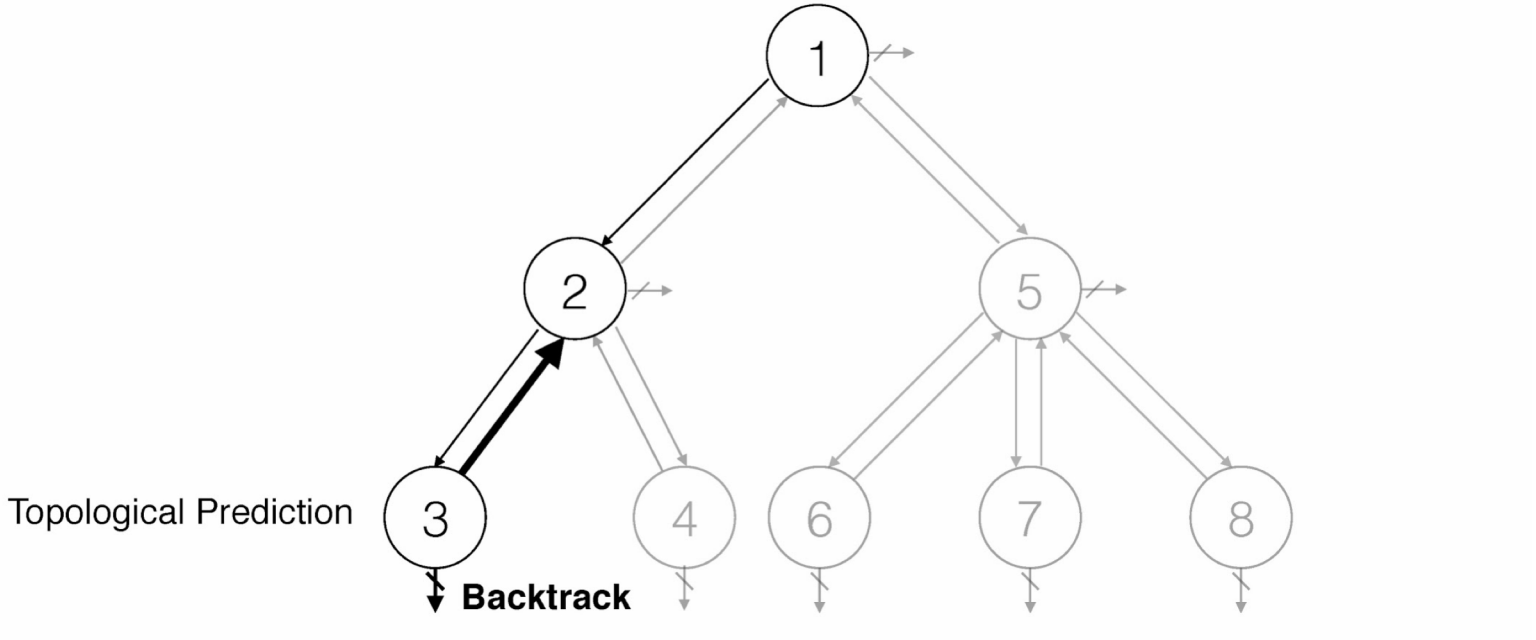
Topological Prediction: Should we add a child node, or backtrack?

Label Prediction: What do we label the new node?

Credits to Wengong Jin

Tree decoding

Junction Tree Variational Autoencoder, Jin et al. (ICML 2018); Tree-Structured Decoding, Alvarez-Melis & Jaakkola (ICLR 2017)



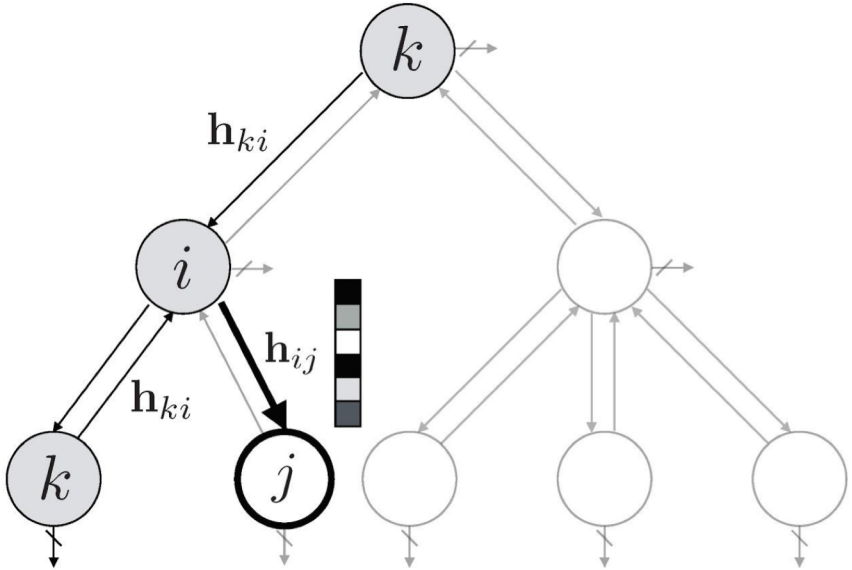
Topological Prediction: Should we add a child node, or backtrack?

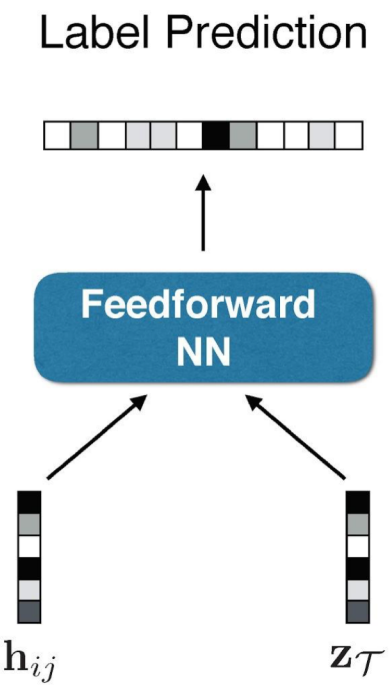
Label Prediction: What do we label the new node?

Credits to Wengong Jin

Tree decoding

Junction Tree Variational Autoencoder, Jin et al. (ICML 2018); Tree-Structured Decoding, Alvarez-Melis & Jaakkola (ICLR 2017)





$$\mathbf{h}_{ij} = \mathrm{GRU}(\mathbf{x}_i, \{\mathbf{h}_{ki}\}_{k \in N_t(i) \setminus j})$$

Encodes state of subtree thus far

Functional group features

Credits to Wengong Jin

JTVAE evaluation

Junction Tree Variational Autoencoder, Jin et al. (ICML 2018)

Method	Reconstruction	Validity
CVAE	44.6%	0.7%
GVAE	53.7%	7.2%
SD-VAE	76.2%	43.5%
GraphVAE	-	13.5%
Atom-by-Atom LSTM	-	89.2%
JT-VAE	**76.7%**	**100.0%**

1 Molecule Reconstruction

100 forward passes per molecule, report portion of decoded molecules identical to input

2 Molecule Validity

Random samples from latent z, report portion that are chemically valid (RDKit)
JTVAE without validity checking: 93.5%

Method	1st	2nd	3rd
CVAE	1.98	1.42	1.19
GVAE	2.94	2.89	2.80
SD-VAE	4.04	3.50	2.96
JT-VAE	**5.30**	**4.93**	**4.49**

3 Bayesian Optimization

1. Train a VAE, associate each molecule with latent vector (mean of encoding distribution)
2. Train a **sparse GP** to **predict target chemical property y(m)** given the latent representation
3. Use property predictor for BO

Credits to Wengong Jin

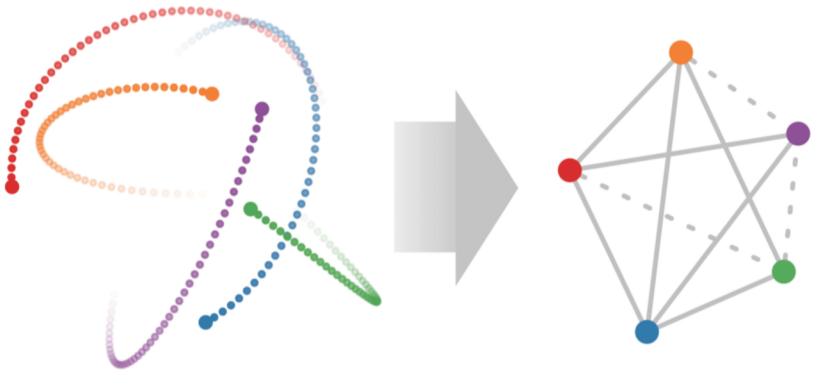
3 Research frontiers

Deep generative graph models

Latent graph inference

Modeling implicit/hidden structure

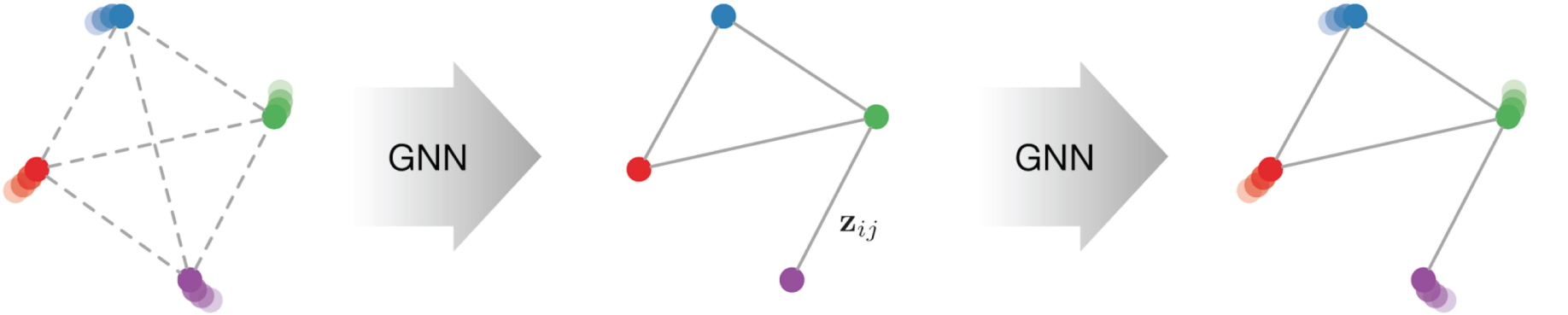
Neural Relational Inference for Interacting Systems, Kipf & Fetaya et al. (ICML 2018)



Credits to Thomas Kipf

Neural Relational Inference with GNNs

Neural Relational Inference for Interacting Systems, Kipf & Fetaya et al. (ICML 2018)



Observe dynamics

$\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^T)$

Interaction graph proposal

$\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$

Discrete (Gumbel softmax trick)

[Jang et al., 2016, Maddison et al., 2016]

Reconstruct dynamics*

$p_\theta(\mathbf{x}|\mathbf{z})$

VAE Objective (ELBO) $\mathcal{L} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})]$

Reconstruction **Regularization**

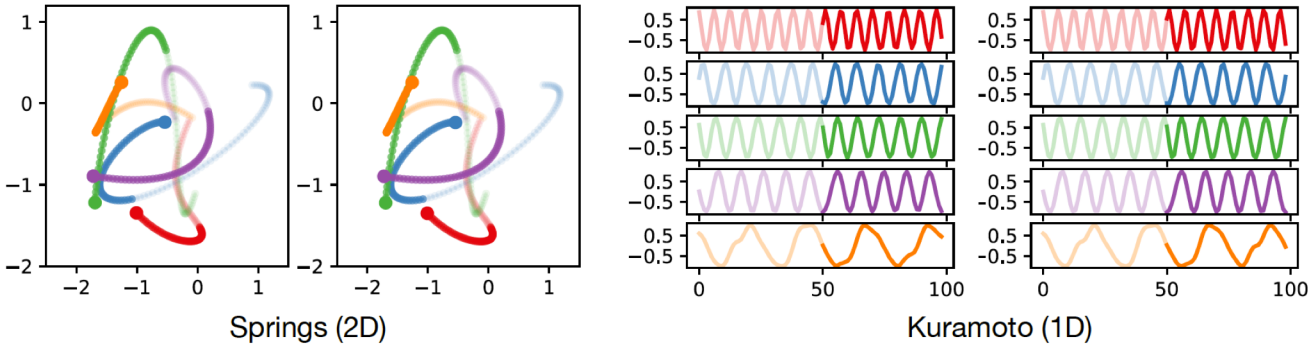
Credits to Thomas Kipf

NRI evaluation - toy data

Neural Relational Inference for Interacting Systems, Kipf & Fetaya et al. (ICML 2018)

Table 6.1: Accuracy (in %) of unsupervised interaction recovery.

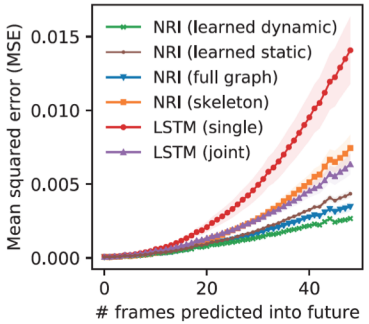
	Springs		Kuramoto	
Number of objects	5	10	5	10
Correlation (path)	52.4±0.0	50.4±0.0	62.8±0.0	59.3±0.0
Correlation (LSTM)	52.7±0.9	54.9±1.0	54.4±0.5	56.2±0.7
NRI (simulation decoder)	**99.8**±0.0	**98.2**±0.0	–	–
NRI (learned decoder)	**99.9**±0.0	**98.4**±0.0	**96.0**±0.1	**75.7**±0.3
Supervised	99.9±0.0	98.8±0.0	99.7±0.0	97.1±0.1



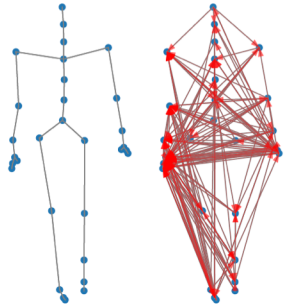
Credits to Thomas Kipf

NRI evaluation - CMU Motion Capture (e.g. walking)

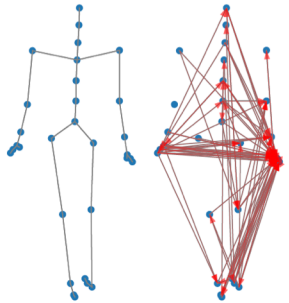
Neural Relational Inference for Interacting Systems, Kipf & Fetaya et al. (ICML 2018)



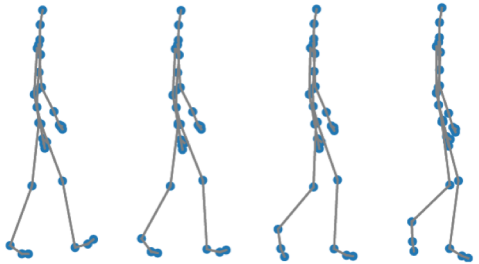
(a) Test MSE comparison



(b) Latent graph (left step)



(c) Latent graph (right step)



(c) Motion capture data

Credits to Thomas Kipf

NRI applications - causal discovery

Amortized Causal Discovery: Learning to Infer Causal Graphs from Time-Series Data, Lowe et al. 2020

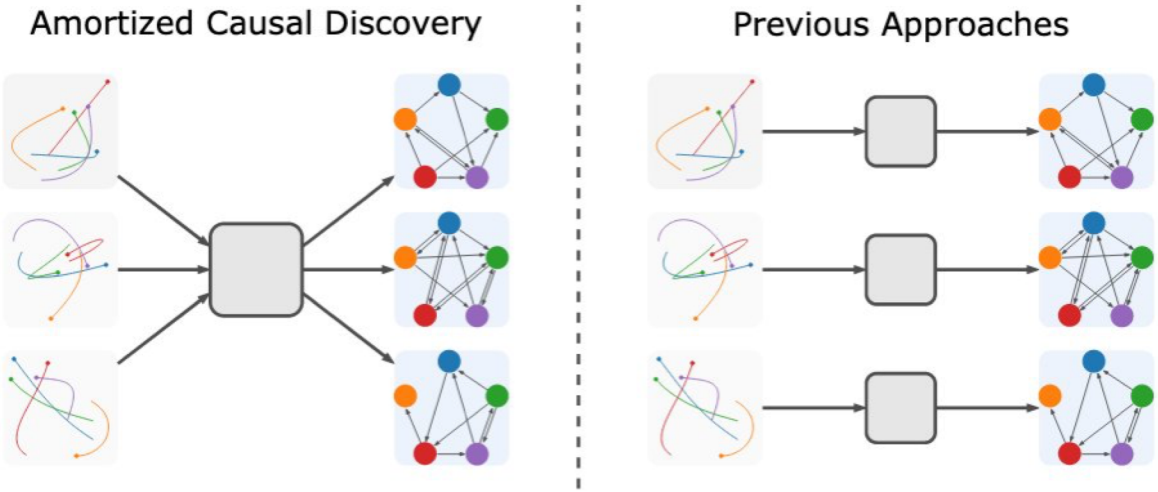


Figure 1: Amortized Causal Discovery. We propose to train a single model that infers causal relations across samples with different underlying causal graphs but shared dynamics. This allows us to generalize across samples and to improve our performance with additional training data. In contrast, previous approaches fit a new model for every sample with a different underlying causal graph.

Credits to Thomas Kipf

Challenges and future work in graph neural nets

- **Problems of neighborhood aggregation / message passing**
  - Theoretical relation to WL isomorphism, simple graph convolutions; tree-structured computation graphs → **bounded power**
  - **Oversmoothing** (residual/gated updates help, but don't solve)
  - See recent work from Max Welling e.g. Natural Graph Networks
- **Scalable, stable generative models**
- Learning on **large, evolving data**
- (Mostly) **assume a graph structure is provided** as input
  - Neural Relational Inference is a preliminary work here, also see Pointer Graph Networks (Velickovic et al., NeurIPS 2020)
- **Multi-modal and cross-modal learning** (e.g. sequence2graph)