Computational Intelligence &
Machine Learning Group

Deep Learning for Graphs – Module II

DAVIDE BACCIU (BACCIU@DI.UNIPI.IT) – ALESSIO MICHELI (MICHELI@DI.UNIPI.IT)

DIPARTIMENTO DI INFORMATICA - UNIVERSITA' DI PISA

https://sites.google.com/view/dl4sd

Module Outline

- Convolutional neural network for graphs
  - Spectral approach
  - Generalizing spatial convolutions
- Contextual approaches (layering)
  - Node2Vec et al
  - Learning unsupervised graph embedding
- Research directions
- Applications

A Nomenclature Nightmare

Deep learning for graphs

Neural networks for graphs

Graph neural networks

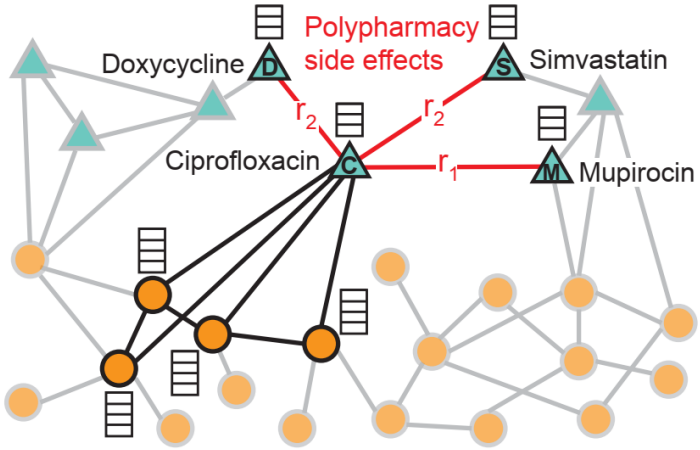
CNN for/on graphs

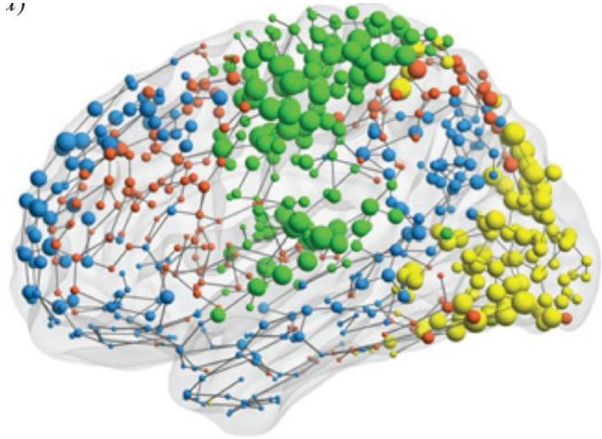
Graph CNN

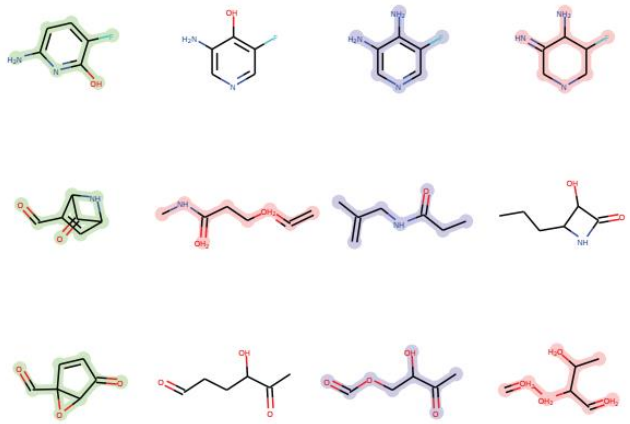
Learning graph/node embedding

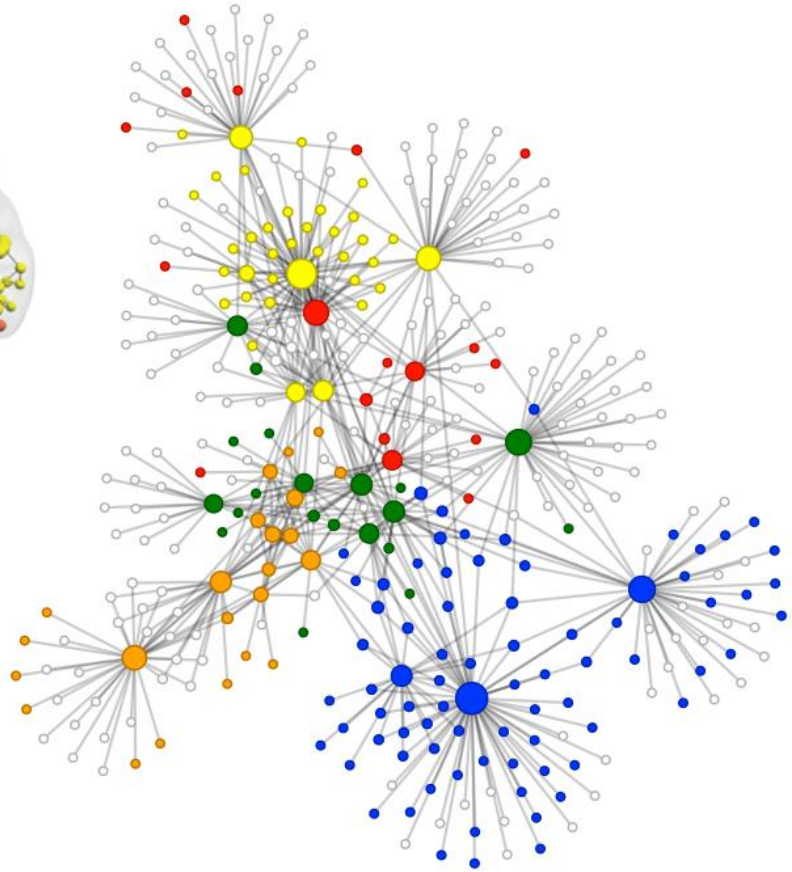
Geometric deep learning

Motivation – Learning with graphs



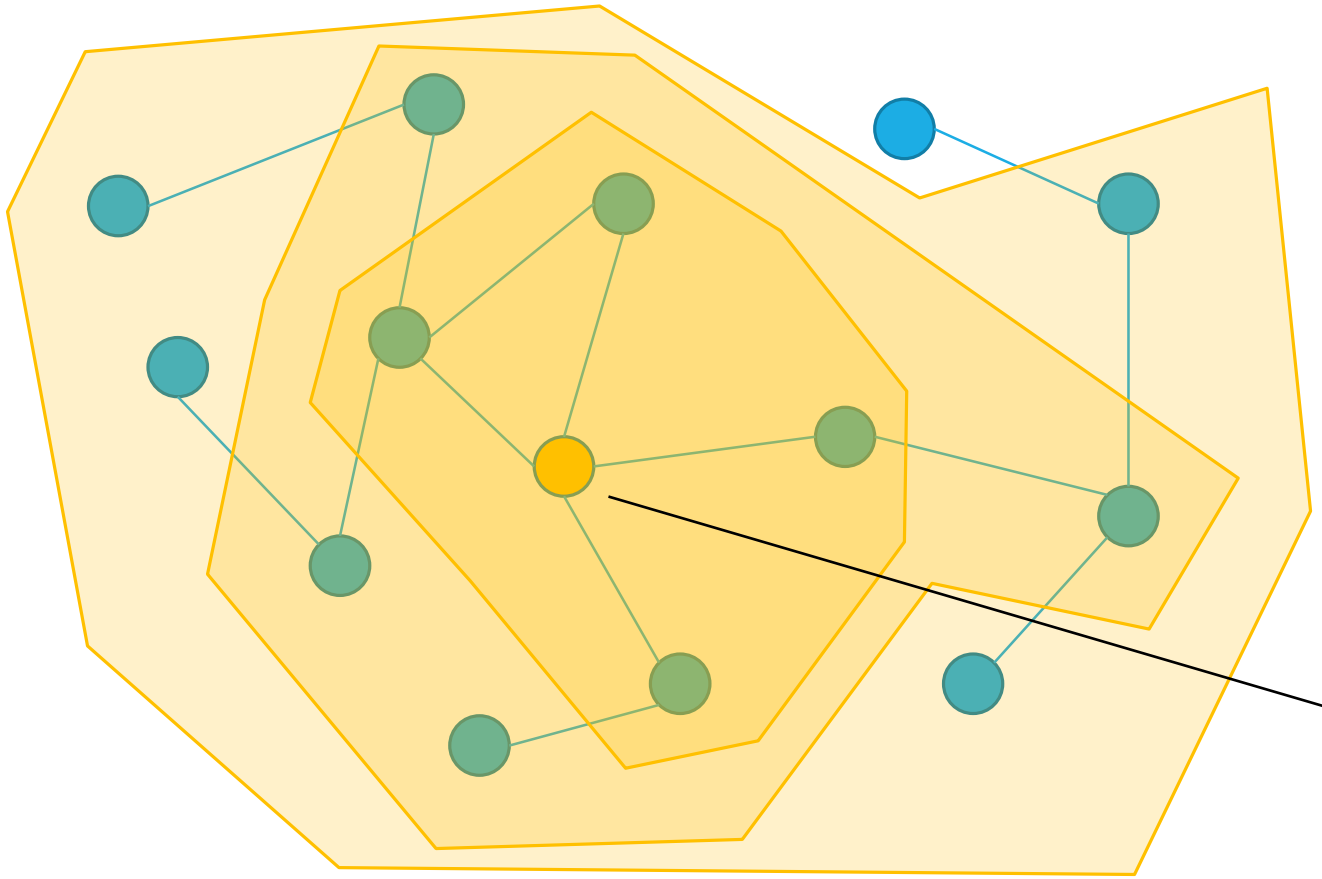




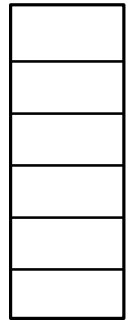


Motivation – **Deep** Learning with graphs

Hierarchical representation learning allows to efficiently diffuse information through graph structure

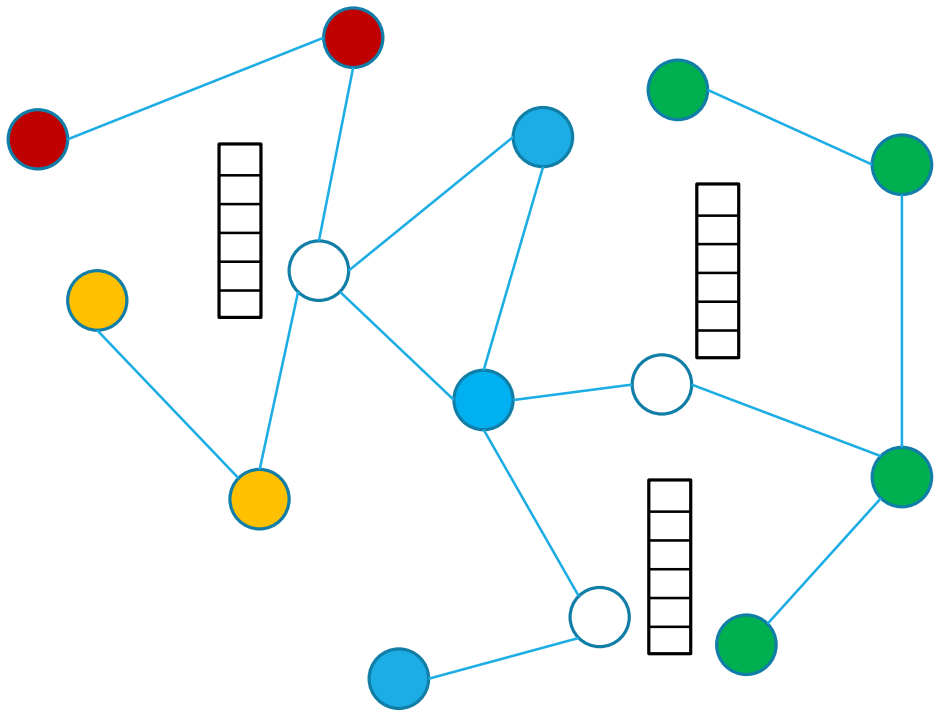


Node representation depends on its context (shorter first-longer later)

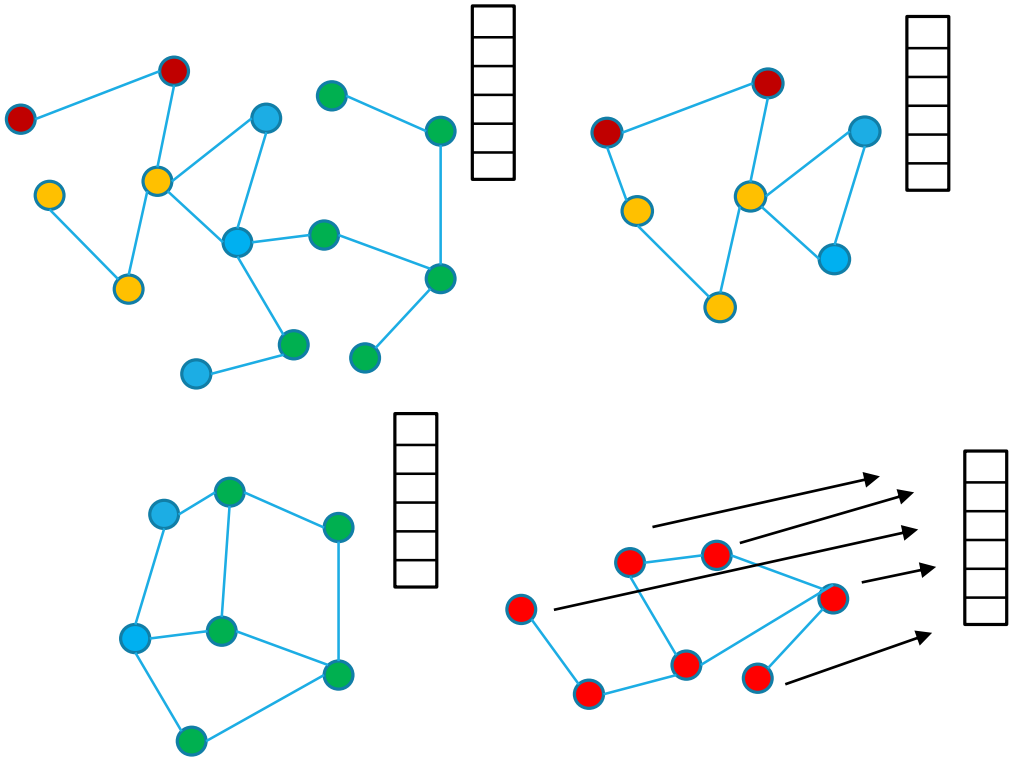


Predictive Tasks

Network data



Graph classification/regression

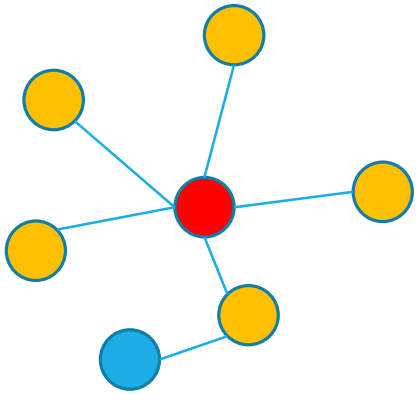


Convolutional Neural Networks for Graphs

How to Perform Convolutions on Graphs

SPATIAL DOMAIN

What is the equivalent of sliding a kernel to aggregate local spatial information?



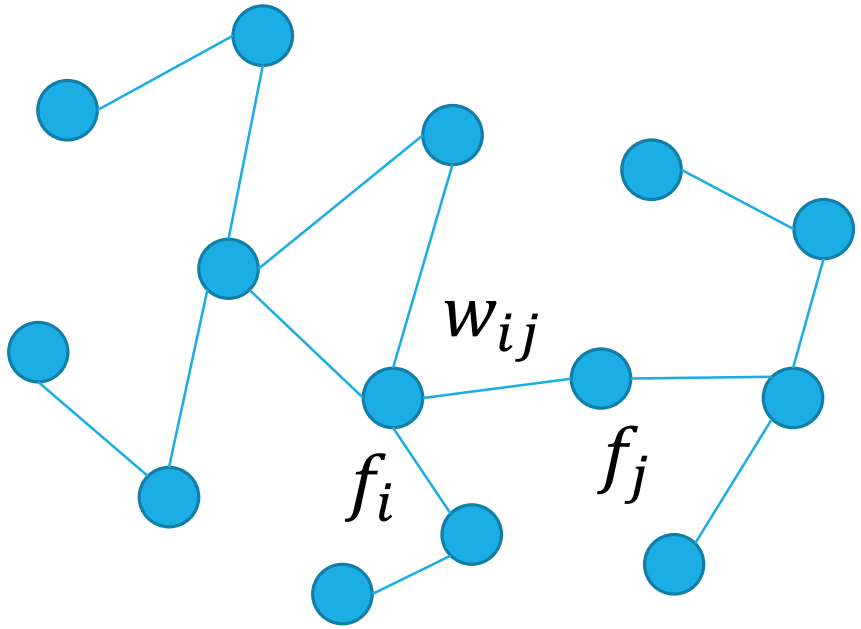
SPECTRAL DOMAIN

$$\mathcal{F}(f * g) = \mathcal{F}(f) \times \mathcal{F}(g)$$

Exploit the Convolution Theorem and Fourier analysis to perform convolutions in the spectral domain

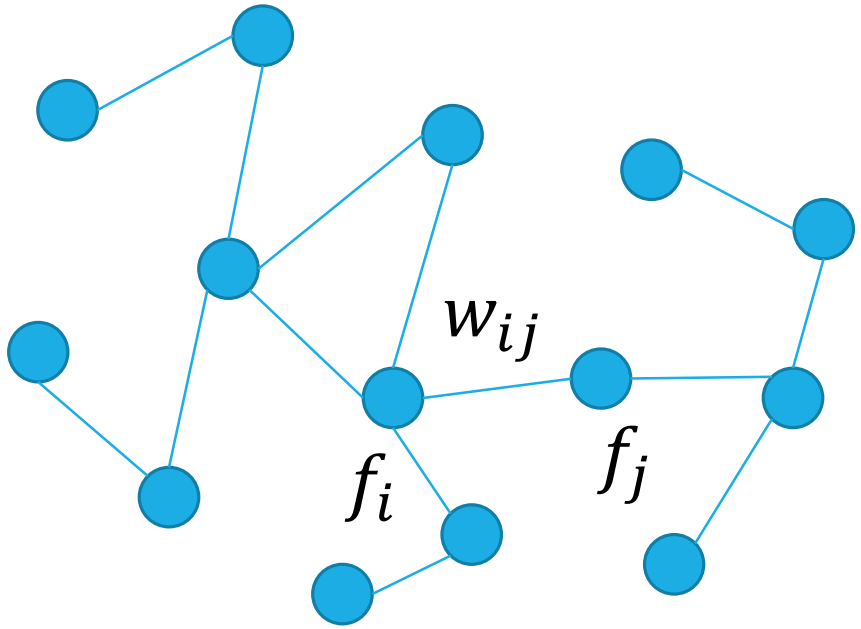
Spectral Domain Convolutions

The Scenario



- Single weighted undirected graph
  - $w_{ij} > 0$ weight of the i-j edge
- Functions f_i attaching values (i.e. labels/signals x_i) to nodes i
- Task: process the signals defined on the graph structure

Graph Fundamentals (I)



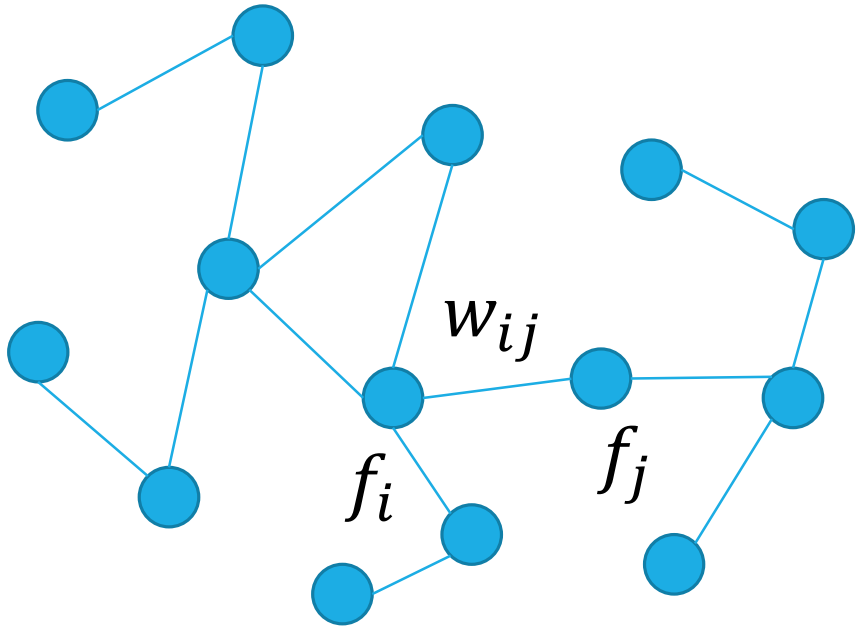
Adjacency matrix

$$A = \begin{bmatrix} 0 & \cdots & w_{1N} \\ \vdots & \ddots & \vdots \\ w_{N1} & \cdots & 0 \end{bmatrix}$$

Degree matrix

$$D = \begin{bmatrix} D_{11} = \sum_{j \epsilon \mathcal{N}_1} w_{1j} & 0 & 0 & 0 \\ 0 & D_{22} & 0 & 0 \\ 0 & 0 & D_{33} & 0 \\ 0 & 0 & 0 & D_{44} \end{bmatrix}$$

Graph Fundamentals (II)



We are interested in the Laplacian

$$L = D - A$$

L is symmetric and positive semidefinite

L is an operator on the vector space of functions $\boldsymbol{f} = [f_1 \dots f_N]$

Difference between vertex i information and the weighted contribution from its neighbors

$$(L\boldsymbol{f})(i) = \sum_{j \in \mathcal{N}_i} w_{ij}(f_i - f_j)$$

Normalized Laplacian

In practice the normalized (graph) Laplacian is often used

$$L = I_n - D^{-1/2} A D^{-1/2}$$

Normalization ensures that application of the Laplacian operator is scale invariant (performs an actual local averaging of the neighbors)

I_n is the nxn identity matrix allowing a node to consider its signal in the weighted local average

One last bit

Since the Laplacian is real symmetric and positive semidefinite

$$L = U\Lambda U^T$$

Λ diagonal matrix of eigenvalues $\lambda_k \geq 0$

U matrix containing complete orthonormal basis $\{u_k\}_k$ associated to each λ_k

So where are the graph convolutions?

The Convolution Theorem

The Fourier transform of the $f * g$ convolution can be transformed into a matrix multiplication of the transform of the functions f and g

$$\mathcal{F}(f * g) = \mathcal{F}(f) \times \mathcal{F}(g)$$

Fourier Analysis (in 1 ugly slide)

Decompose a function f as a combination of vectors $\boldsymbol{e}_k$ from an orthonormal basis

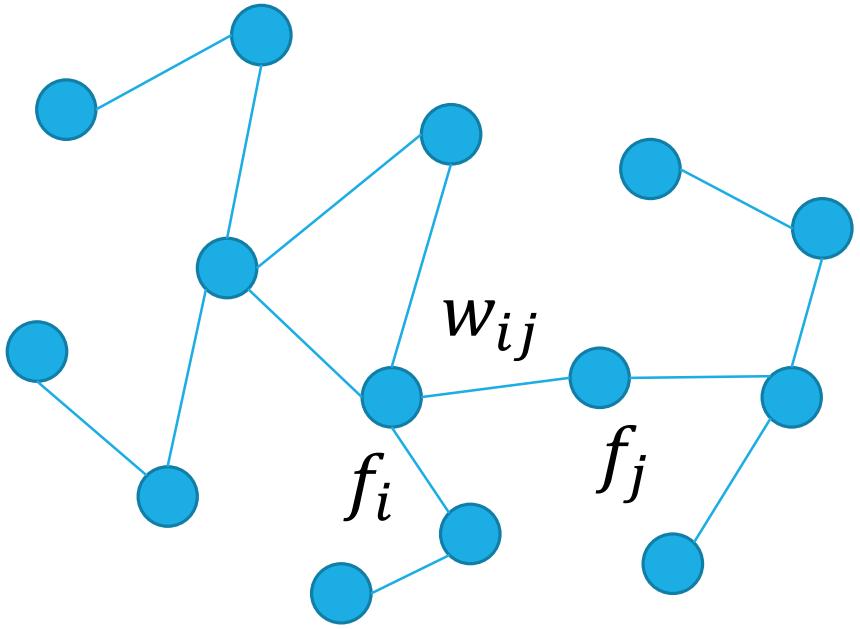
$$f = \sum_{k \geq 0} \tilde{f}_k \boldsymbol{e}_k$$

A signal is a superposition of basis functions at different frequencies k

$\tilde{f}_k$ are the Fourier coefficients (at frequency k), obtained by projecting f on the basis (easily sometimes, e.g. FFT)

Back on the Graph

The eigendecomposition of the graph Laplacian L gives us an orthonormal basis U for the graph



The graph Fourier transform of a signal $\mathcal{F}\ (f_i)$ is $\tilde{f}_i = U^T\ f_i$

Can be inverted using $f_i = U\ \tilde{f}_i$

Eigenvalues λ_k are the graph frequencies

Spectral Graph Convolution

Given a graph G and the associated Fourier basis we can compute the graph convolution of its node signals $\boldsymbol{f}$ with a filter

$$(\boldsymbol{f} *_G \boldsymbol{g}) = \mathcal{F}^{-1}\big(\mathcal{F}(\boldsymbol{f})\,\mathcal{F}(\boldsymbol{g})\big) = U\boldsymbol{U^T g}U^T\boldsymbol{f}$$

Convolutional filter in spectral domain

Graph equivalent of the learnable CNN filter matrix **W**

$$\boldsymbol{h} = U\boldsymbol{W}U^T\boldsymbol{f}$$

Parameterized graph convolution, with parameter matrix **W**

About the Spectral Filter Parameters

$$\boldsymbol{h} = U\boldsymbol{W}U^T\boldsymbol{f}$$

- $\boldsymbol{f}$ vertex vector function (e.g. the matrix holding vector numerical labels)
- $\boldsymbol{W}$ is a $[\mathbf{N} \times \boldsymbol{N}]$ diagonal matrix of learnable filter coefficients
- Like in spatial convolution $\boldsymbol{W}$ can contain all free parameters (easiest choice)

- Filters are basis dependent ($\boldsymbol{W} = U^T\boldsymbol{g}$), hence they are graph specific
- With unconstrained $\boldsymbol{W}$
  - Convolution has $O(N)$ parameters
  - Forward and inverse transform cost $O(N^2)$
  - No guarantee of spatial localization of the resulting filter

Need smart ways of constraining **W**

Spectral Graph Convolution Models (I)

Bruna, Zaremba, Szlam, LeCun, ICLR 2014; Henaff, Bruna, LeCun, Arxiv 2015

- Filter localization in space by filter smoothness in frequency domain
- Parametrize the filter using a smooth spectral transfer function $W(\Lambda)$

Defferrard, Bresson, Vandergheynst, NIPS 2016 (ChebNet)

- Represent spectral transfer function as a Chebychev polynomial of order *r*
- *O*(r) parameters θ_j and *O*(Nr) computational cost

$$W(\Lambda) = \sum_{j=0}^{r} \theta_j \Lambda^j$$

ChebNet Architecture

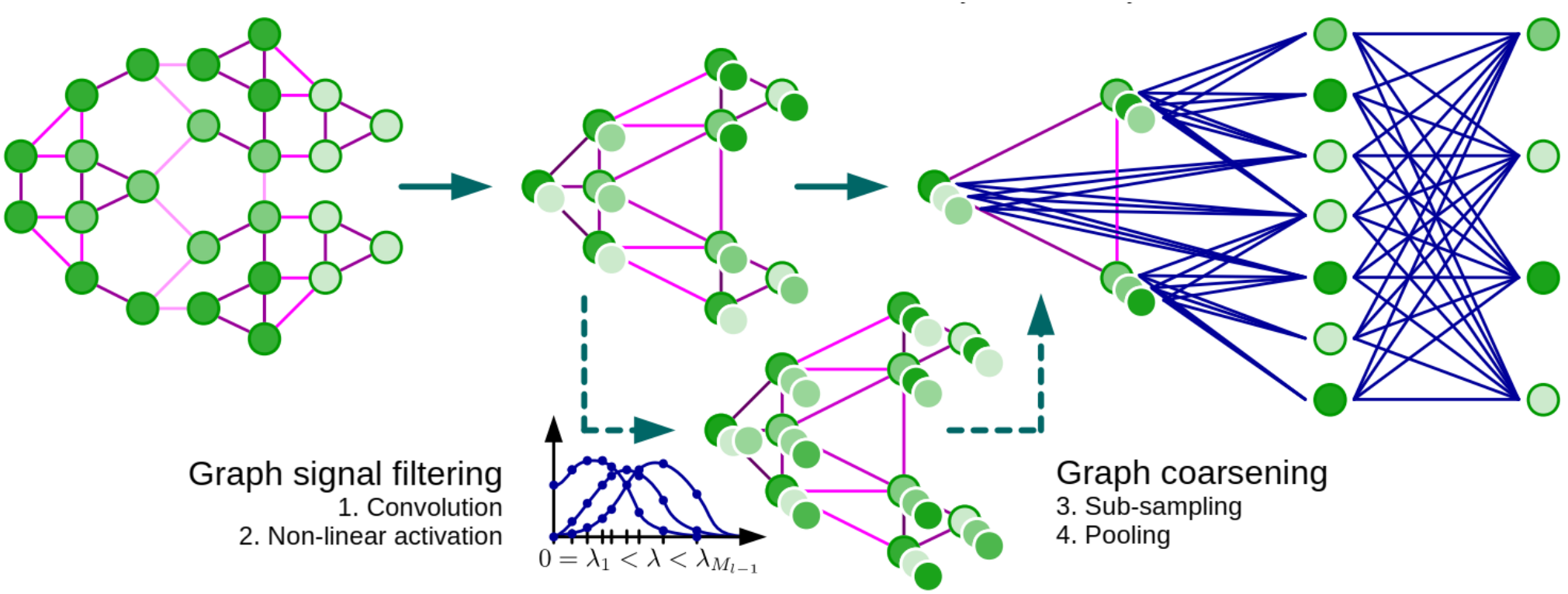


Figure from Defferrard, Bresson, Vandergheynst, NIPS 2016

Spectral Graph Convolution Models (II)

Kipf, Welling, ICLR 2017

- Simplify Chebychev polynomials to degree $r = 2$ and approximate largest Laplacian eigenvalue to ≈ 2 (using renormalization to control numerical stability)
- Constrain parameters so that $\theta = \theta_0 = \theta_1$ (single parameter)

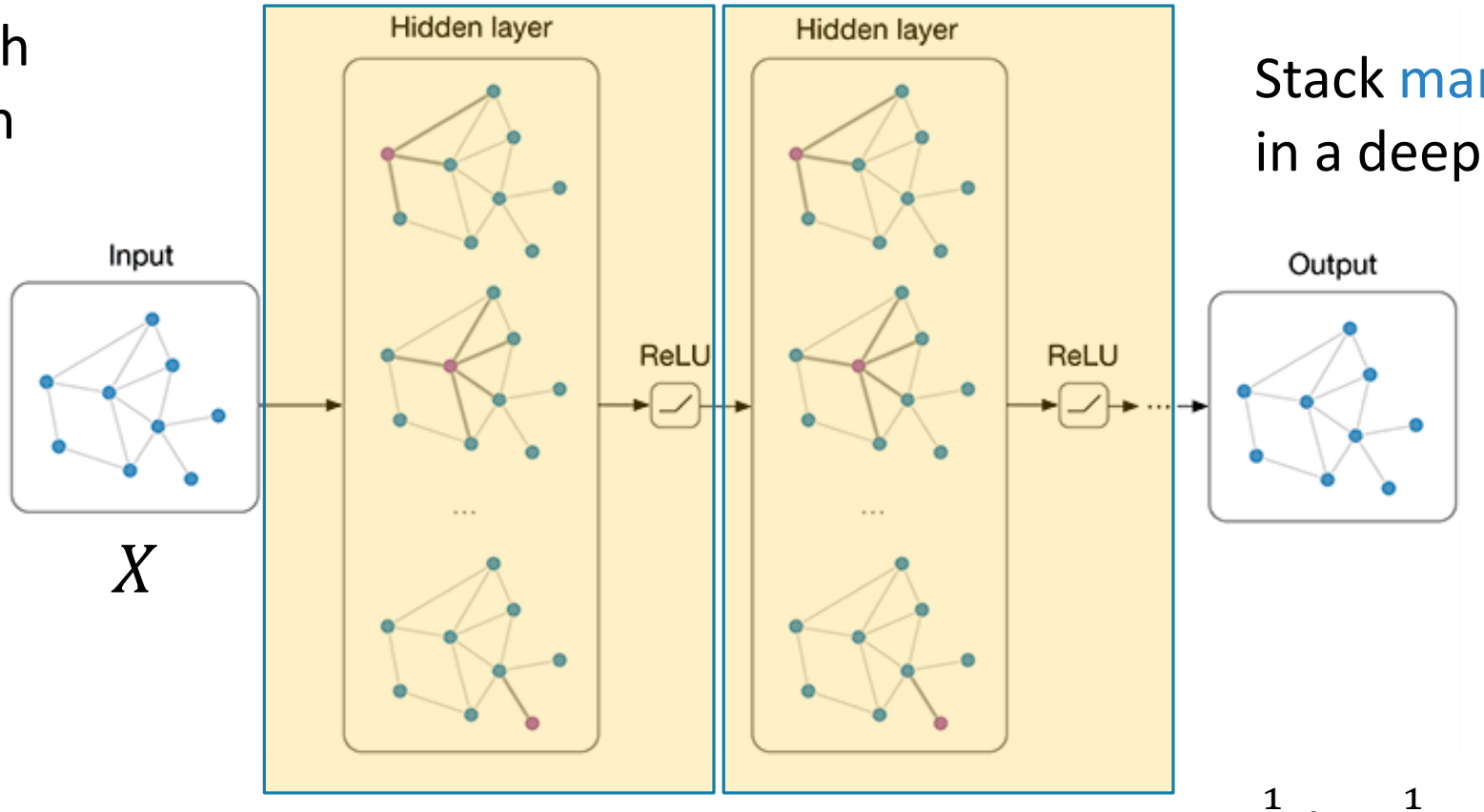
$$\boldsymbol{h} = \theta(I_n + D^{-1/2}AD^{-1/2})\,\boldsymbol{f}$$

Graph Convolutional Networks (GCN) framework

Figure credit to T. Kipf

GCN Neural Architecture

Relabel the graph structure at each layer with the newly obtained embedding (Weisfeiler-Lehman like)



X

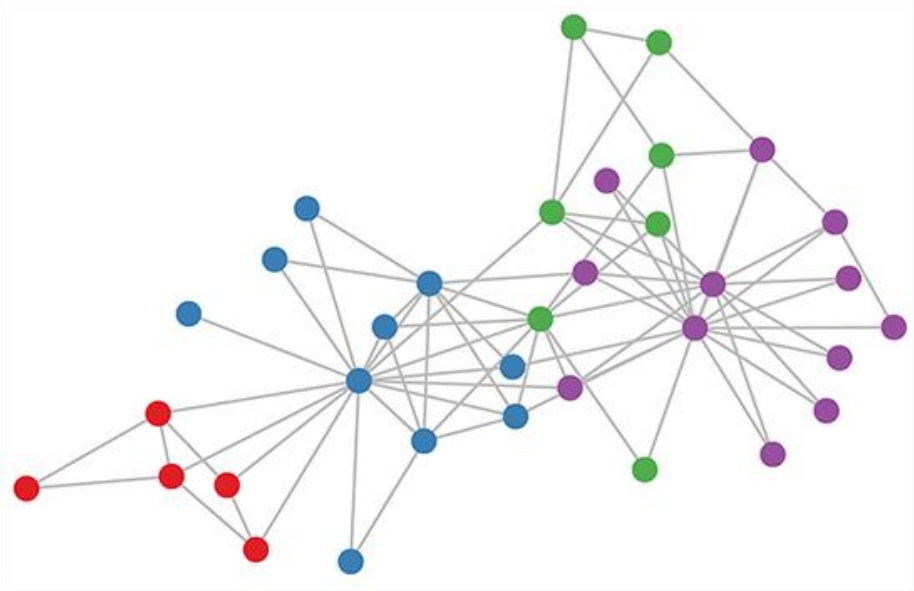
Stack many simple layers in a deep-like fashion

$$h^1 = \sigma(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}XW^1)$$

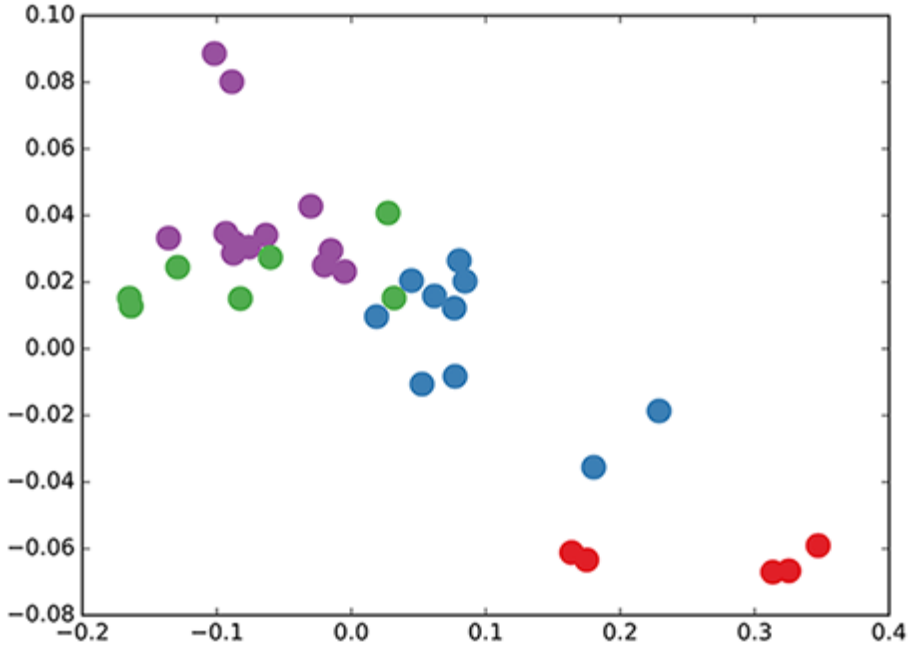
$$h^2 = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}h^1W^2)$$

Figure credits to T. Kipf

How Good Are the Vertices Embeddings?



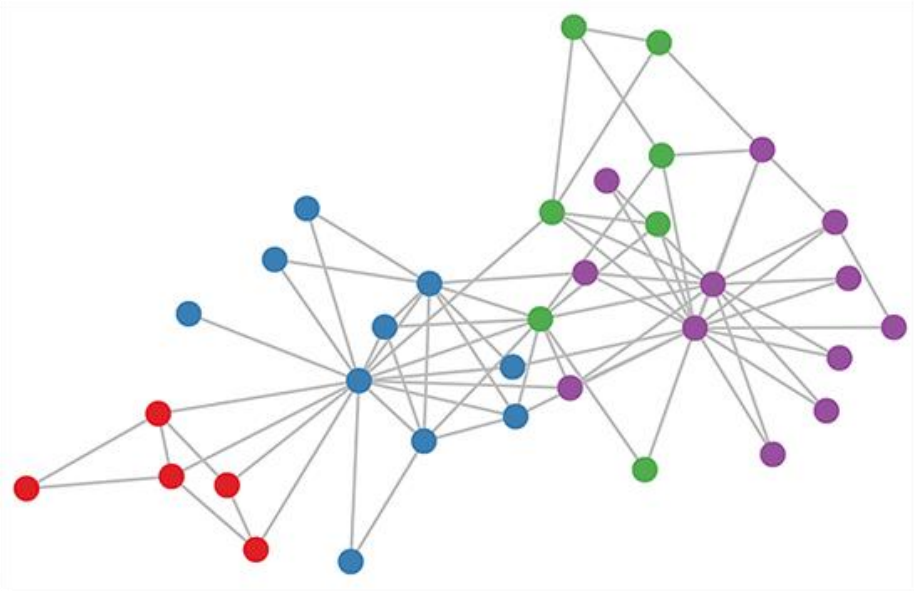
Karate club community detection network



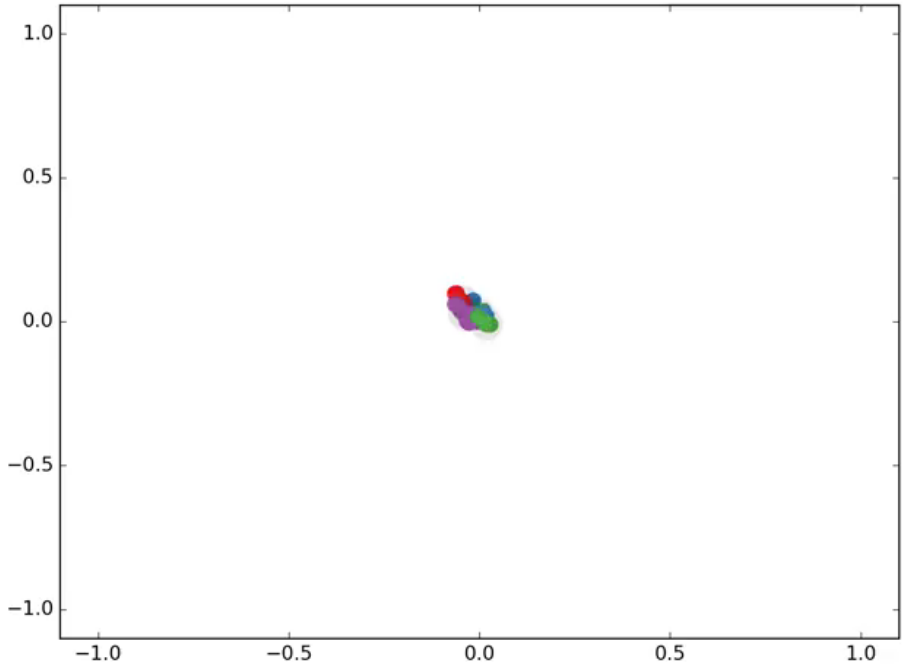
GCN embedding of vertices with randomly initialized untrained weights

Figure credits to T. Kipf

How Good Are the Vertices Embeddings?



Karate club community detection network



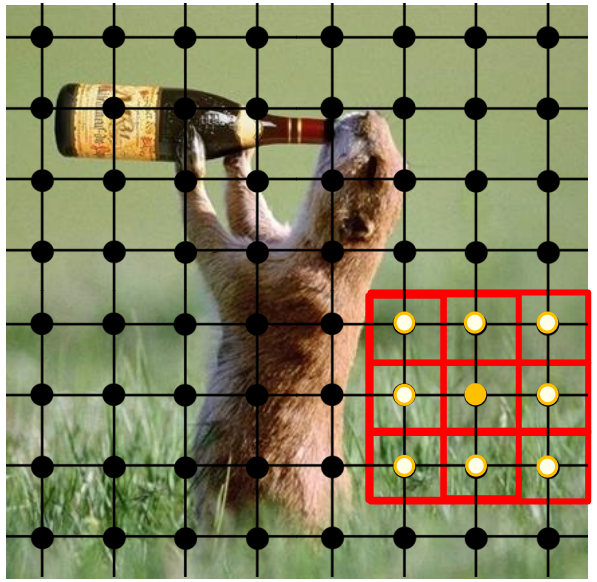
GCN embedding of vertices evolution during training

A Final Word on Spectral Approaches

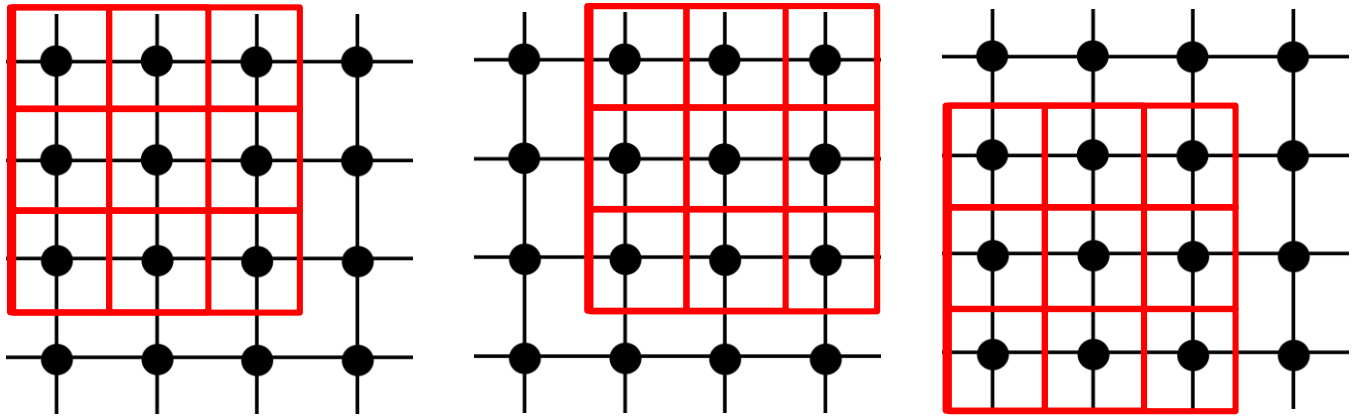
- Cannot handle multiple graphs due to convolution dependency on Laplacian (use on network data tasks)
- Mostly limited to undirected graphs with unlabeled edges
  - Extension to directed graphs using Laplacian block structure and triangular motifs (Benson et al 2016; Monti, Otness, Bronstein 2018)
- Difficult control on context diffusion through the graph structure
- Working with the Laplacian can be impractical for large graphs

Spatial Domain Convolutions

A Graph View on Convolutions



Visual convolutions are graph convolutions on a regular grid

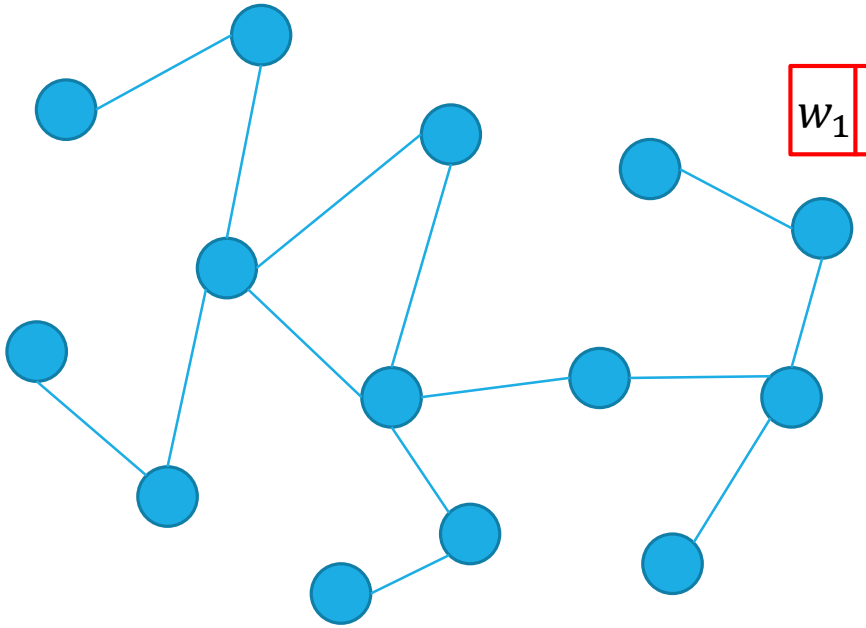


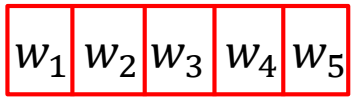
Plus some key assumptions which make it difficult to directly apply them to graphs

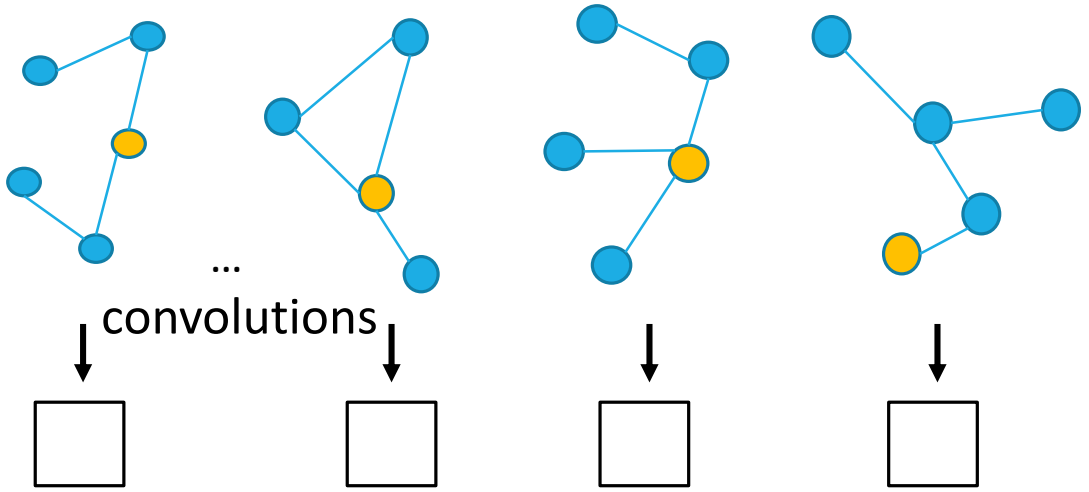
- Regular neighborhood
- Existence of a total node ordering

Node Neighborhoods

Example of 4-neighborhoods







Neighborhoods depend on node ordering: how can I get coherent node ordering across multiple graphs?

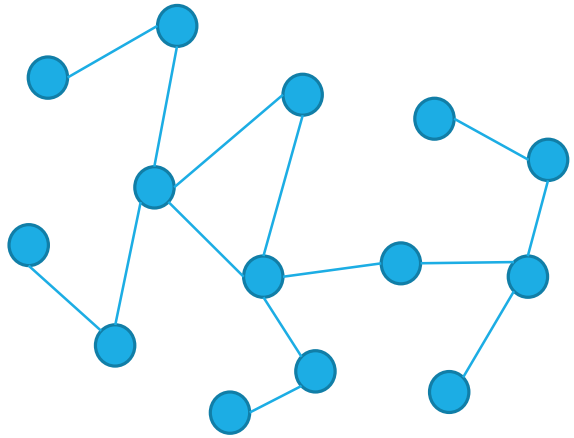
PATCHY-SAN (I)

Niepert, Ahmed, Kutzkov, ICML 2016

Leverage graph labelling techniques (e.g. Weisfeiler-Lehman) to determine a coherent ordering within the graph and between the graphs

1. Node Sequence Selection

Apply graph labelling to reorder the adjacency matrix and visit the graph in the given order



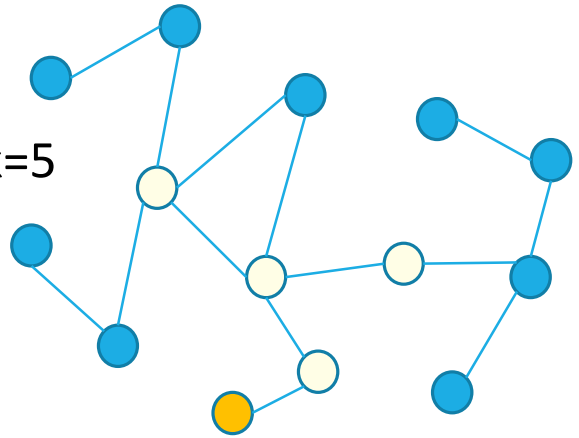
PATCHY-SAN (II)

Niepert, Ahmed, Kutzkov, ICML 2016

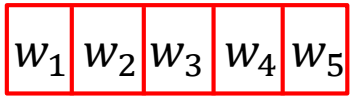
2. Assemble the neighbourhood

Given the current vertex assemble its neighbourhood (of fixed size k) by breadth first search, taking the first k vertices (pad with the neighbours if necessary to reach k nodes).

Neighborhood for k=5



Parametric convolutional filter of size k



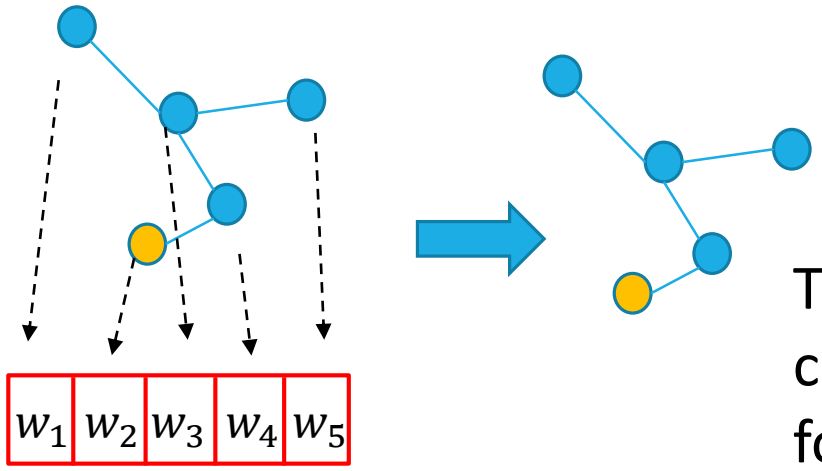
In what order do I map nodes to filter parameters?

PATCHY-SAN (III)

Niepert, Ahmed, Kutzkov, ICML 2016

3. Neighbourhood normalization

Given the current neighbourhood apply labelling techniques to reorder nodes before applying them to the convolutional filter



Graph normalization is NP-complete

The result of the convolution is the encoding for the target vertex

PATCHY-SAN Summary

- Can handle multiple graphs, undirected and directed, with labels on both edges and nodes
- Can reuse CNN machinery: striding, pooling, …
- Performance relies heavily on quality of the ordering
- Edge labels are used only for computing node ordering
- How to choose neighborhood size?
- Worst case complexity is exponential due to graph normalization

Contextual Graph Processing (Layering)

On the Need of Efficient Context Diffusion

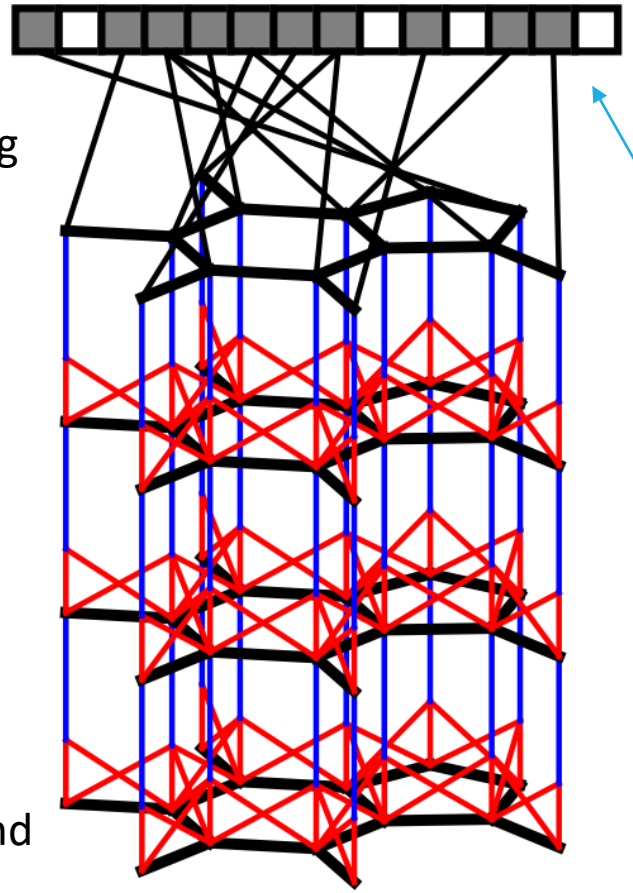
- Defining explicit vertex neighborhoods on multi-graphs is difficult and requires computationally intensive solutions
- An implicit approach exploiting the available graph structure while handling loops
  - Contractive approach – Use dynamical systems with guaranteed attractors to obtain stable vertex encodings (GNN)
  - Contextual approaches – Use layering to avoid direct dependencies on cycles

Contextual Graph Convolutions

Duvenaud, Maclaurin et al, NIPS 2015

Molecular (Circular) Fingerprints

Fixed length encoding



Chemical compound

Sets one bit in the encoding

Smooth hash function

Algorithm 1 Circular fingerprints

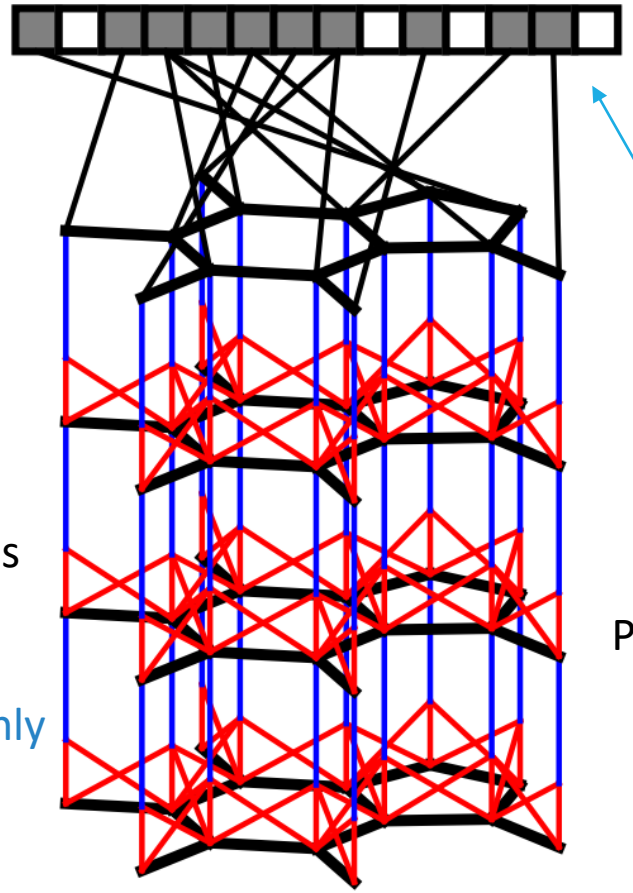
Input: molecule, radius R, fingerprint length S
Initialize: fingerprint vector $\mathbf{f} \leftarrow \mathbf{0}_S$
for each atom a in molecule
  $\mathbf{r}_a \leftarrow g(a)$ ▷ lookup atom features
for $L = 1$ to R ▷ for each layer
  for each atom a in molecule
    $\mathbf{r}_1 \ldots \mathbf{r}_N = \text{neighbors}(a)$
    $\mathbf{v} \leftarrow [\mathbf{r}_a, \mathbf{r}_1, \ldots, \mathbf{r}_N]$ ▷ concatenate
    $\mathbf{r}_a \leftarrow \text{hash}(\mathbf{v})$ ▷ hash function
    $i \leftarrow \text{mod}(r_a, S)$ ▷ convert to index
    $\mathbf{f}_i \leftarrow 1$ ▷ Write 1 at index
Return: binary vector $\mathbf{f}$

Duvenaud, Maclaurin et al, NIPS 2015

Neural (differentiable) Fingerprints

Training by error backprop

Current node representation is computed from previous layer contributions only (contextual)



Differentiable encoding

Pooling

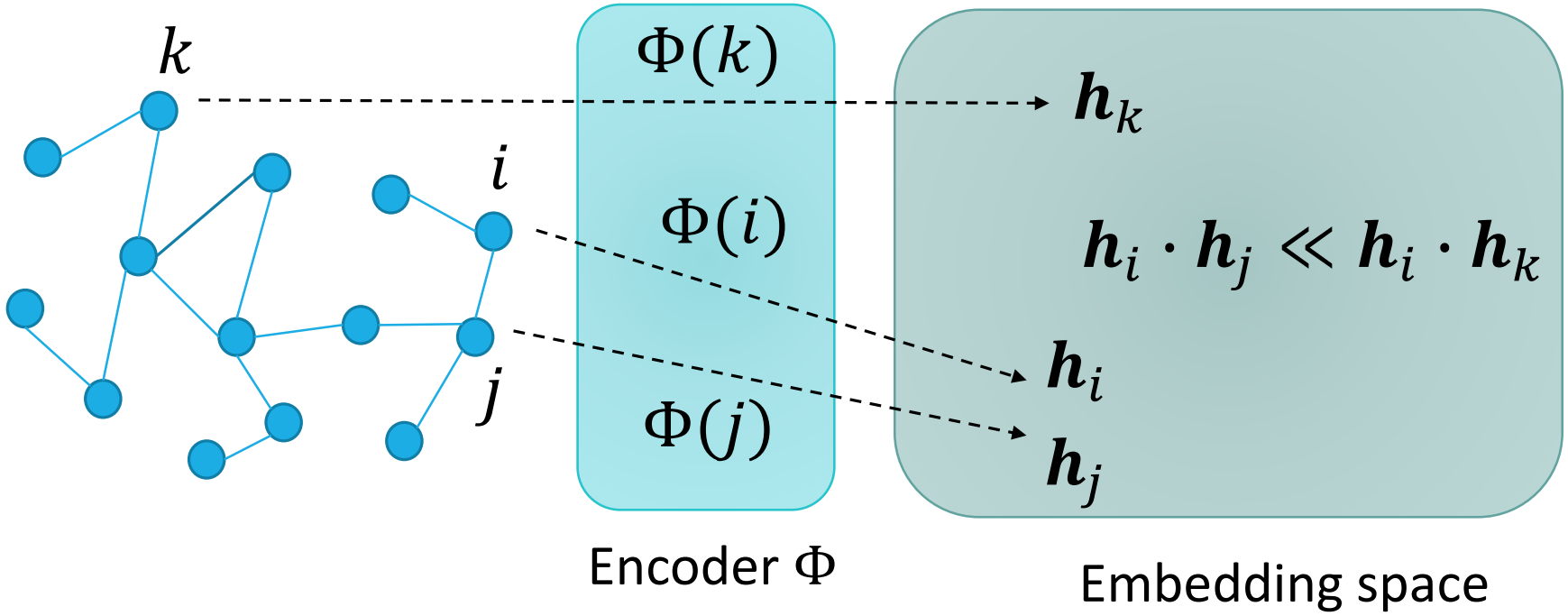
Algorithm 2 Neural graph fingerprints

Input: molecule, radius R, hidden weights $H_1^1 \dots H_R^5$, output weights $W_1 \dots W_R$
Initialize: fingerprint vector $\mathbf{f} \leftarrow \mathbf{0}_S$
for each atom a in molecule
  $\mathbf{r}_a \leftarrow g(a)$ ▷ lookup atom features
for $L = 1$ to R ▷ for each layer
  for each atom a in molecule
    $\mathbf{r}_1 \dots \mathbf{r}_N = \text{neighbors}(a)$
    $\mathbf{v} \leftarrow \mathbf{r}_a + \sum_{i=1}^{N} \mathbf{r}_i$ ▷ sum
    $\mathbf{r}_a \leftarrow \sigma(\mathbf{v} H_L^N)$ ▷ smooth function
    $\mathbf{i} \leftarrow \text{softmax}(\mathbf{r}_a W_L)$ ▷ sparsify
    $\mathbf{f} \leftarrow \mathbf{f} + \mathbf{i}$ ▷ add to fingerprint
Return: real-valued vector $\mathbf{f}$

Node Embedding

The Key Idea

❖ Encode graph vertices into a vector space where vertex similarities (however defined) are preserved



Vertex Encoder

- A complex function which can take into account node context when generating the vectorial encoding

$$\Phi(k) = \Phi(k|G) \text{ or } \Phi(k|N_k)$$

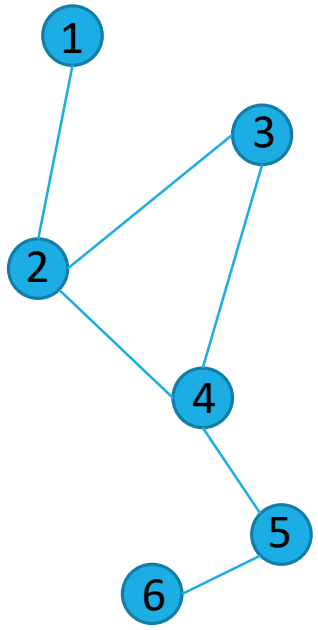
  - Parameterized function so that embedding can be learned to minimize an error (supervised or unsupervised)

$$\boldsymbol{h}_k = \Phi(k|G, \theta) \text{ and } \min_{\theta} L_{\theta}(\boldsymbol{h}_1, \dots, \boldsymbol{h}_k, \dots, \boldsymbol{Y})$$

  - Model differ in how they compute the encoding and the loss

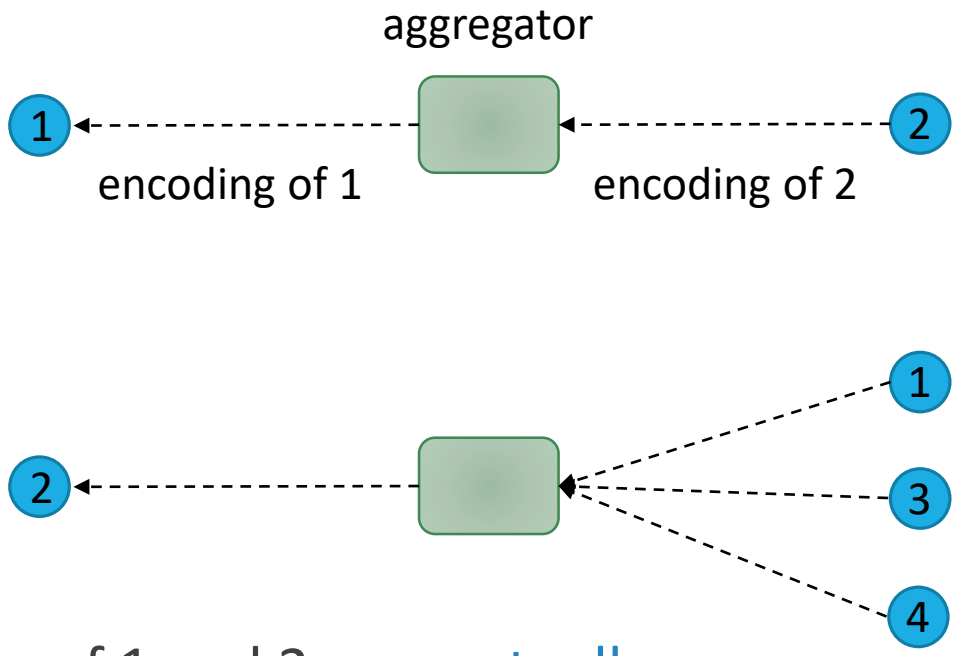
Neighborhood Aggregation

Vertex encoding is generated (again) using the node neighborhood



Neighborhood can be made

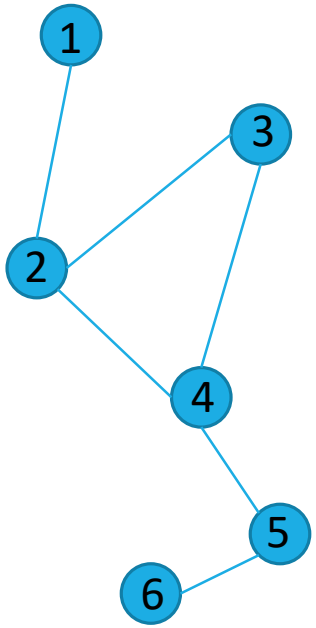
- By adjacent nodes
- By sampling a fixed set of nodes in a predefined neighborhood

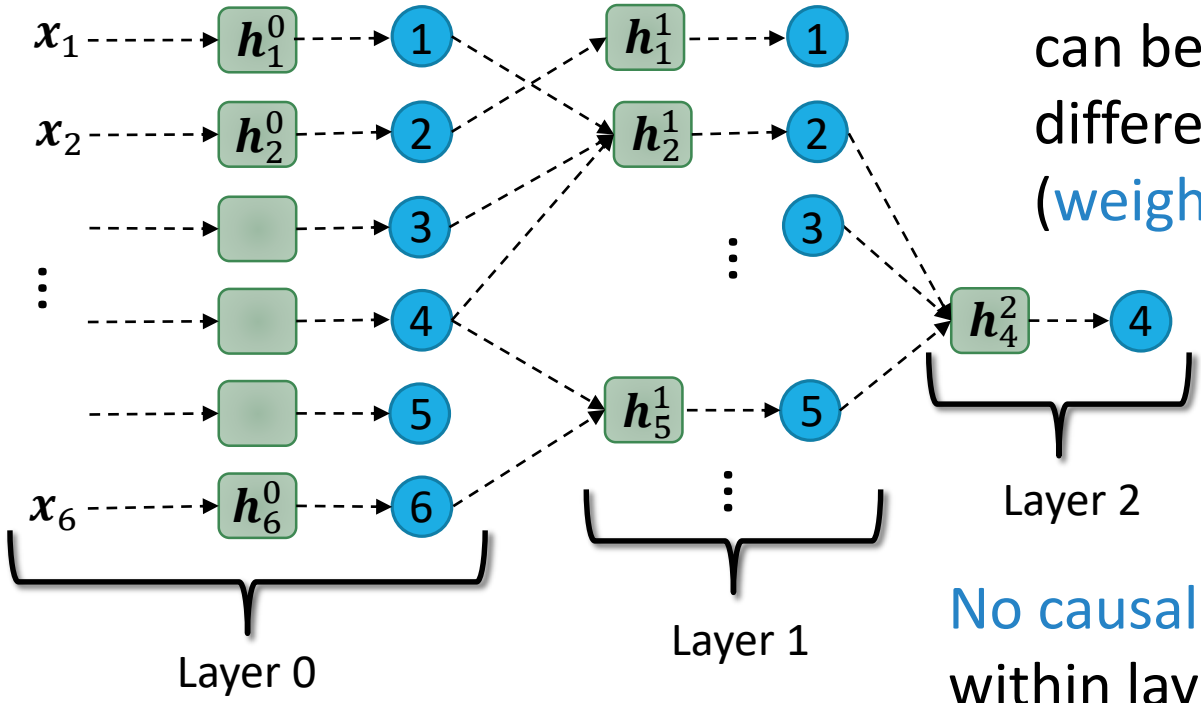


The encoding of 1 and 2 are mutually dependent: how do we solve it?

Layered Encoding

Compute embedding in a layered fashion (contextual)



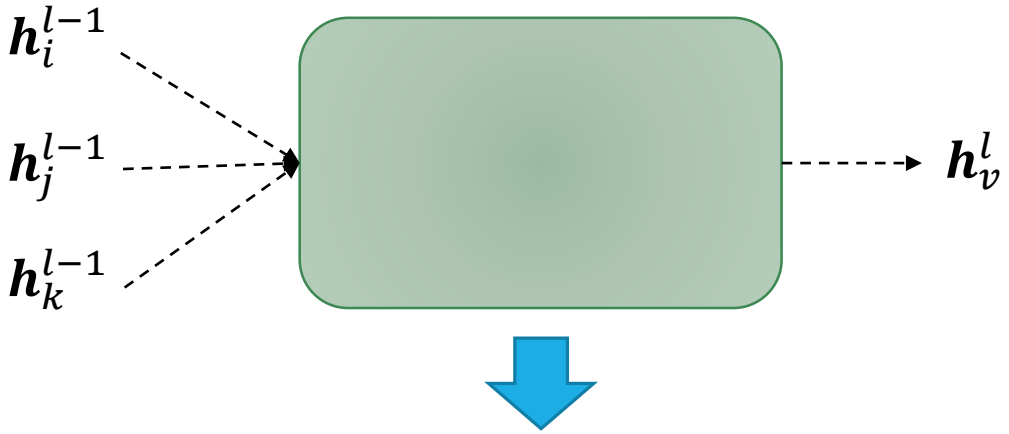


Each layers' aggregator can be parameterized differently or tied (weight sharing)

No causal dependencies within layer

What is inside of the Box?

A learning model of course (e.g. a neural network) including an aggregation function to handle size-varying neighborhoods



A simple model

$$h_v^l = \sigma\left(W_l \sum_{i \in N(v)} \frac{h_i^{l-1}}{|N(v)|} + \widehat{W}_l h_v^{l-1}\right)$$

Simple feedforward layer

Neighborhood average

Can also include contribution from vertex label at each level

Generalizing Aggregation - GraphSage

Hamilton, Ying, Leskovec, NIPS 2017

$$\boldsymbol{h}_v^{\mathrm{l}} = \sigma\left(\boldsymbol{W}_l AGG\left(\left\{\boldsymbol{h}_i^{l-1} : i \in N(v)\right\}\right), \widehat{\boldsymbol{W}}_l \boldsymbol{h}_v^{l-1}\right)$$

Can use any reasonable aggregation strategy

Mean

$$\sum_{i \in N(v)} \frac{\boldsymbol{h}_i^{l-1}}{|N(v)|}$$

Pool

$$max\left(\left\{\sigma\left(\boldsymbol{Q}\boldsymbol{h}_i^{l-1}\right) : i \in N(v)\right\}\right)$$

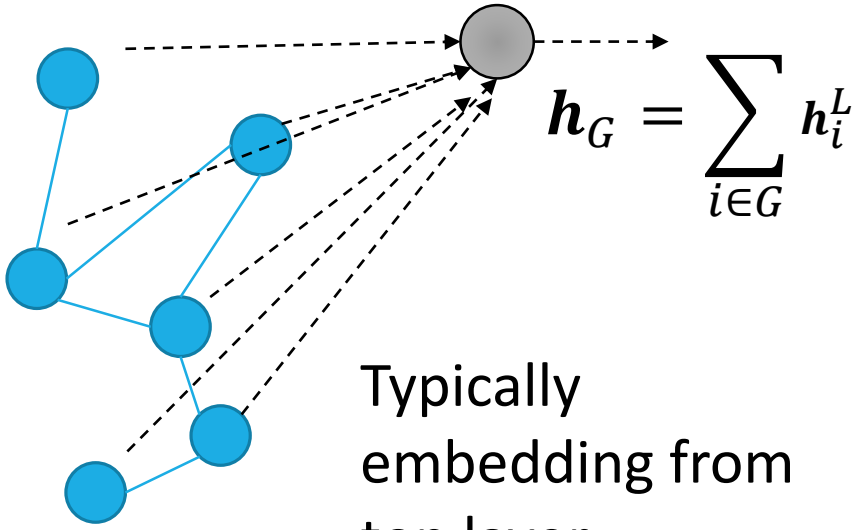
Transform embedding and apply element-wise max pooling

Sequential Aggregation

Feed random permutations of the neighbor embeddings $\boldsymbol{h}_i^{l-1}$ to a recurrent layer (LSTM)

Using Node Embedding

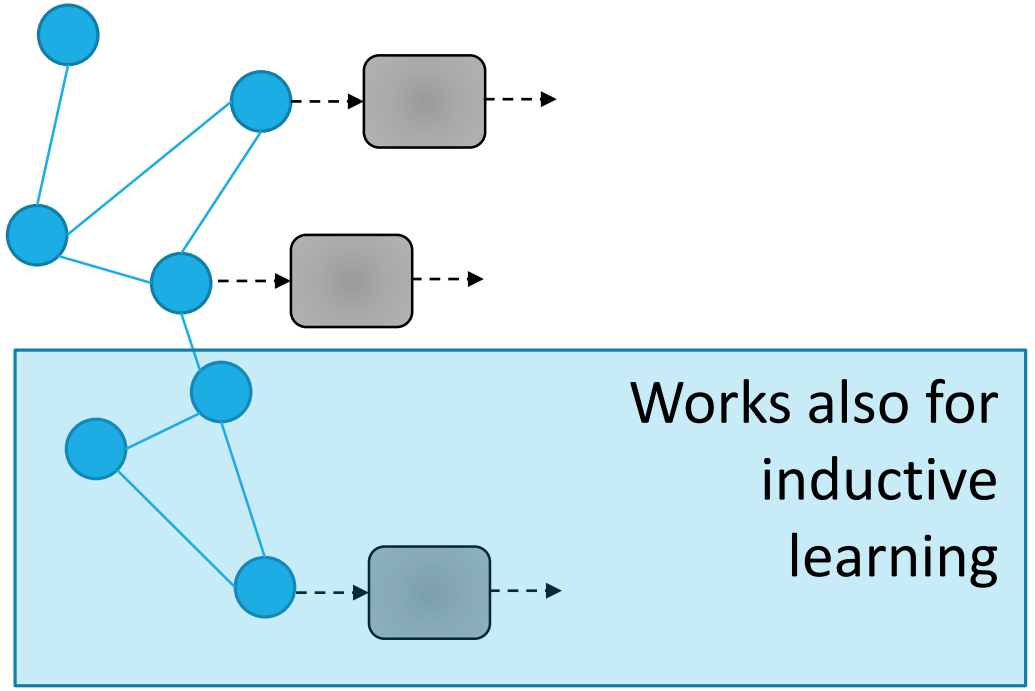
Aggregate all node embeddings to compute graph level predictions



$$\boldsymbol{h}_G = \sum_{i \in G} \boldsymbol{h}_i^L$$

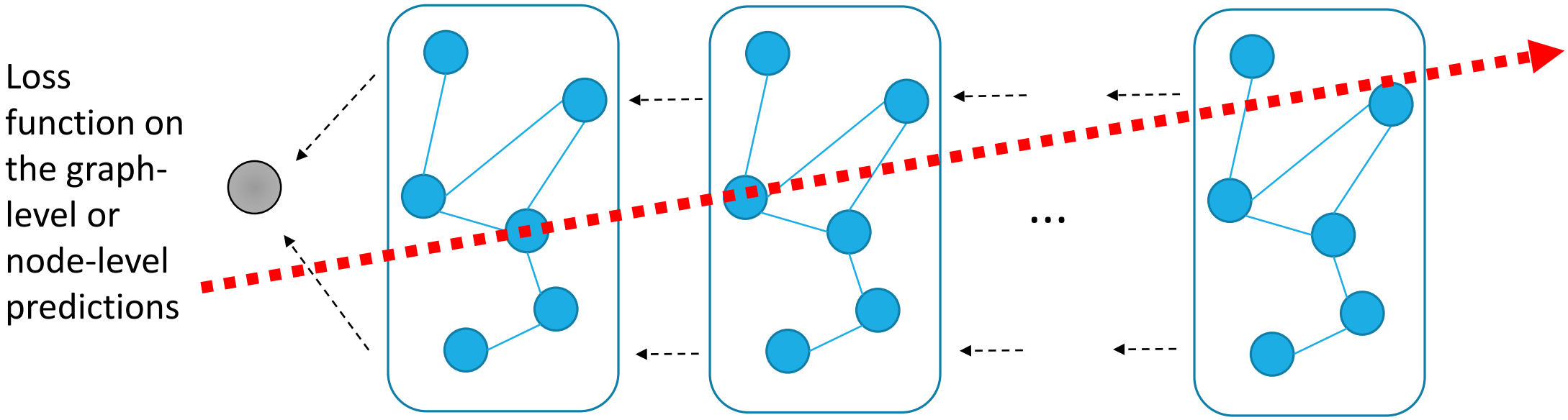
Typically embedding from top layer

Train node level predictors



Training the Embedding

Backpropagate from the (graph or node level) error computed from the top layer embeddings to the early layers



Can also be unsupervisedly trained by using structure induced notions of node similarity (e.g. Node2Vec)

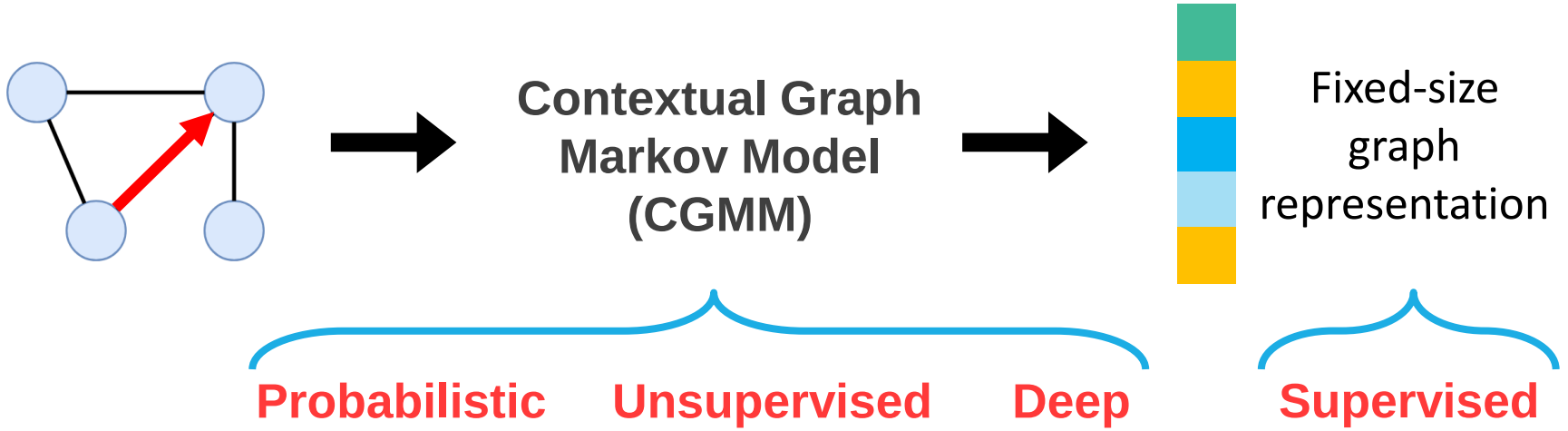
Node Embedding Recap

- Exploit contextual approach to avoid complex neighborhood construction strategies
- No causal dependencies within layers, hence need no fixed-point recurrence
- Restrict context to the preceding layer alone (less general than NN4G)
- Number of layers is typically small due to computational and parameterization issues (end-to-end training requires backpropagation through all layers)
- Embedding are either task-dependent (supervised learning) or need to hand-define similarity in node space (unsupervised learning)

Layer-wise Unsupervised Learning of Embedding

Generative learning for graphs

- General, efficient and scalable architecture
- Handle arbitrary structure (directed, undirected or mixed), labelled edges and nodes
- Learn in both supervised and unsupervised way

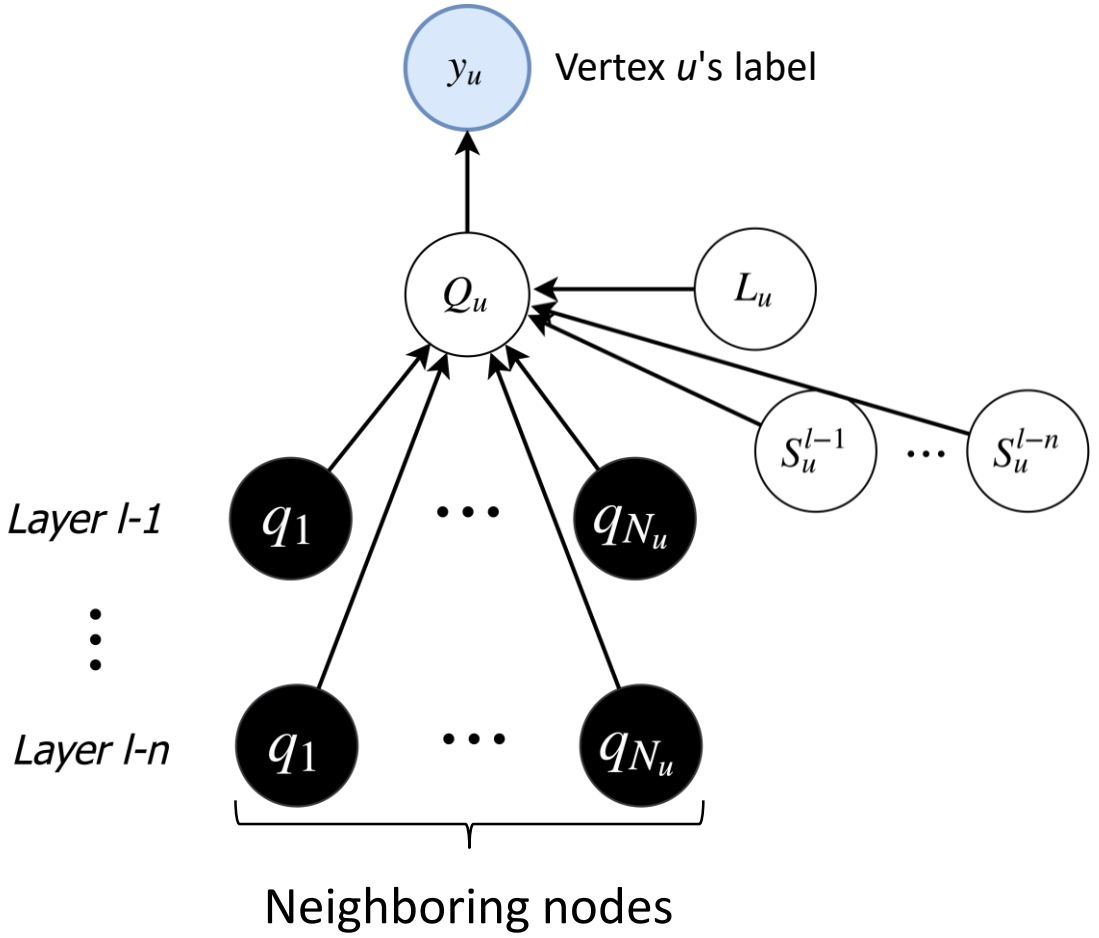


Bacciu, Errica, Micheli , ICML 2018

CGMM in a nutshell

The single layer graphical model

- **Extension** of a standard mixture model
  - Discrete variables
- Likelihood becomes **intractable**
  - Exploit a **Switching Parent approximation**
- Consider only the **direct neighborhood**
- **Full** stationarity



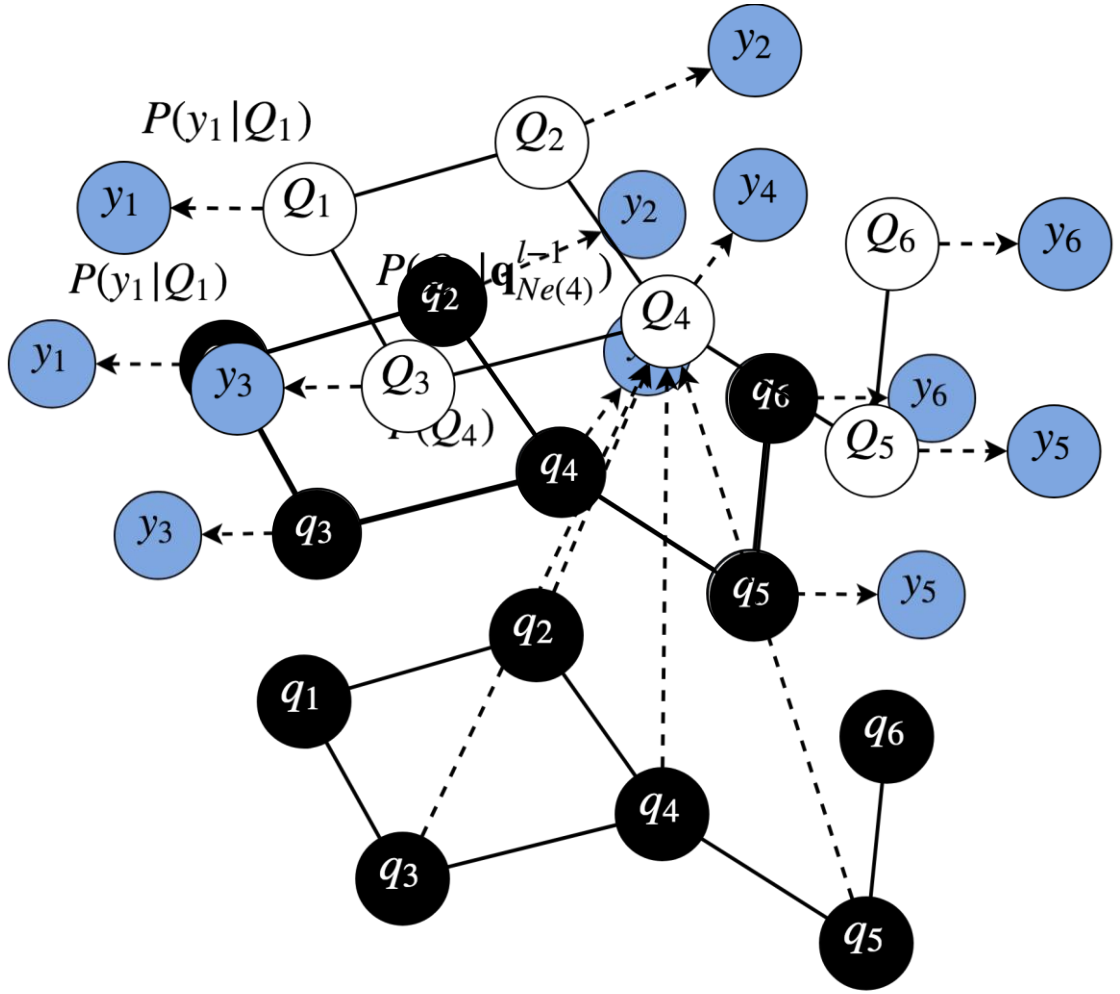
How to build the model

1. Map the graph to the model (base case)

2. Perform inference and freeze states

3. Add a new layer and use frozen states as observed variables in the graphical model

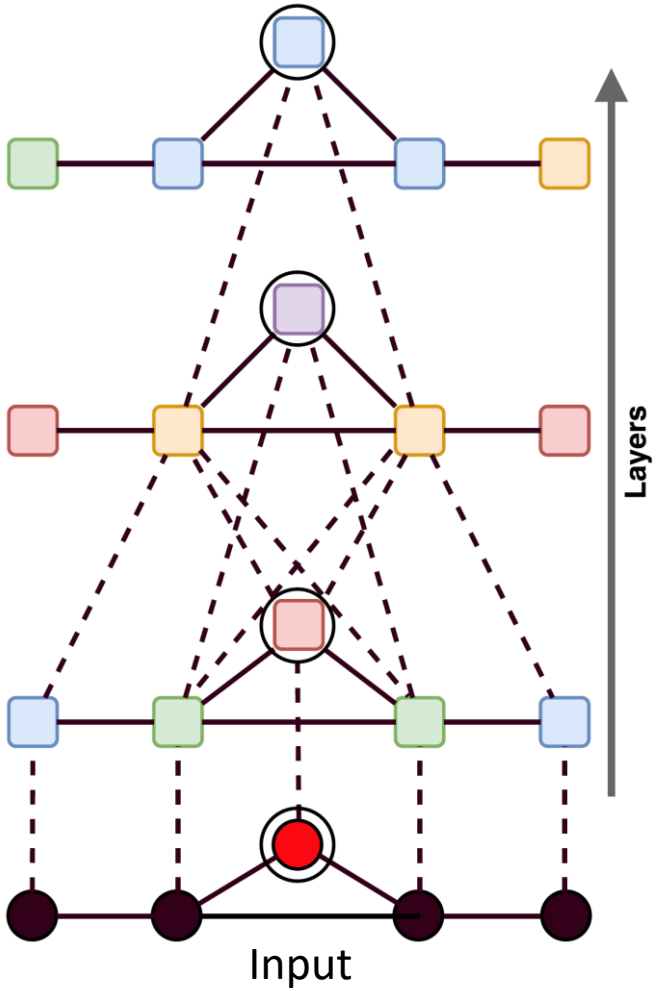
Go back to step 2



CGMM in a nutshell

Symmetric context spreading between vertexes

- Each layer is trained in **isolation**
- Inference computes hidden states' assignments
  - Variables encode information
- The **architecture** diffuses information
  - Deeper net → Wider context window



Learning phase

A maximum likelihood approach to learning

Assumption: i.i.d. graphs

Emission distrib.

Switching Parents distrib.

Transition distrib.

$$\mathcal{L} = \prod_{g \in G} \prod_{u \in g} \sum_{i}^{C} P^l(y_u | Q_u = i) P^l(Q_u = i | \mathbf{q}^{\mathbf{L_{prec}}}_{\mathcal{N}(\mathbf{u})}(\mathbf{g}))$$

Split by layer and by arc

$$= \prod_{g \in G} \prod_{u \in g} \sum_{i}^{C} P^l(y_u | Q_u = i) \sum_{\bar{l} \in \bar{L}} P(L_u = \bar{l}) \sum_{a=1}^{A} P^{\bar{l}}(S_u = a) \underbrace{\frac{\sum_{v \in \mathcal{N}^a(u)} P^{\bar{l},a}(Q_u = i | q_v)}{|\mathcal{N}^a(u)|}}_{\text{Average of the remaining contributions}}$$

Trained by EM

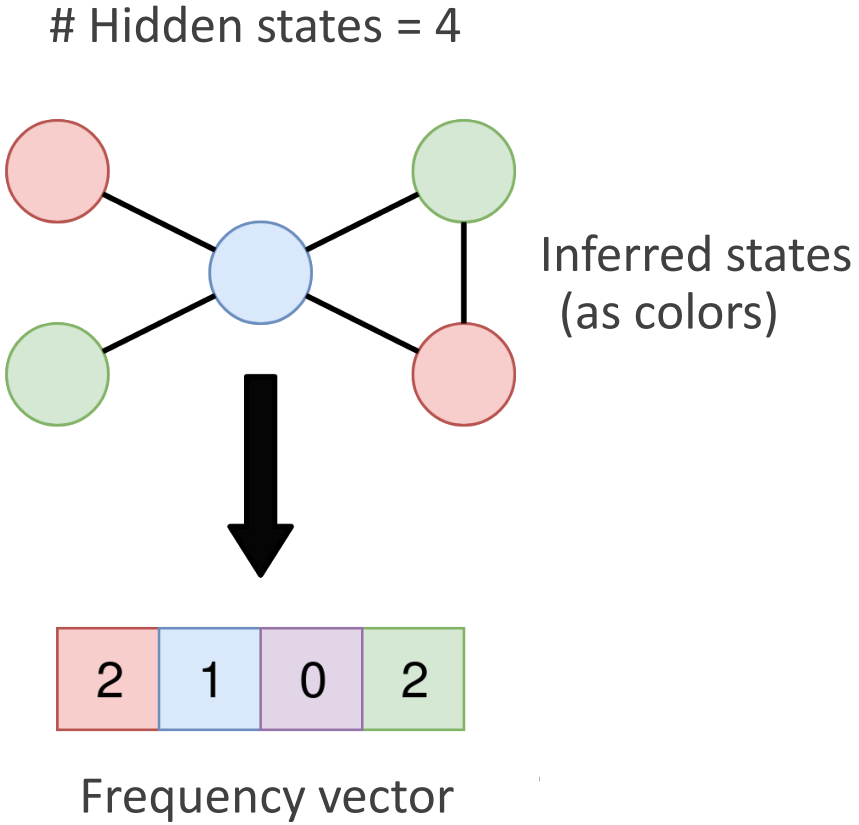
Inference

- Finding the most likely **state assignment**

$$\max_i P(y_u|Q_u = i)P(Q_u = i|\mathbf{q}_{\mathcal{N}(u)})$$

- The inferred latent states are used as observable variables in subsequent layers

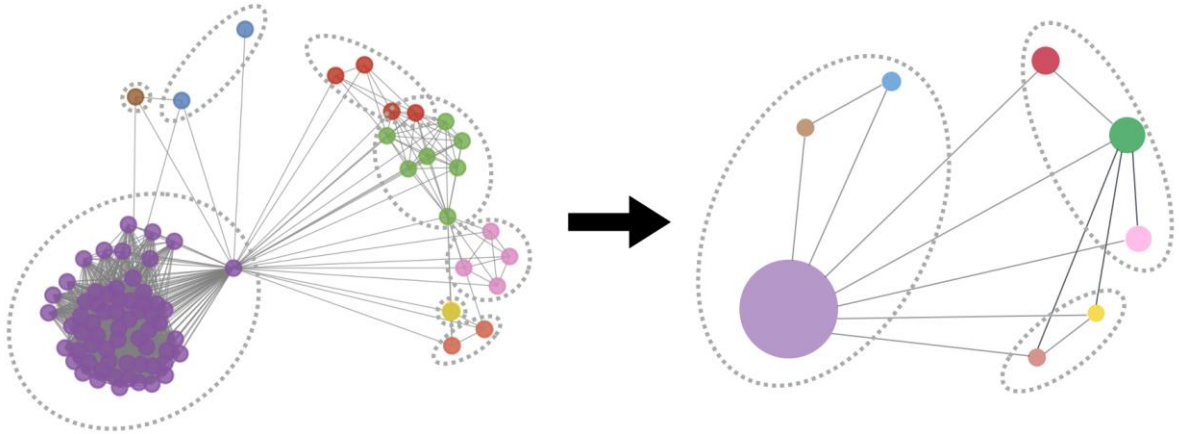
- A **fixed-size vector of states frequencies** as graph encoding



Research Directions

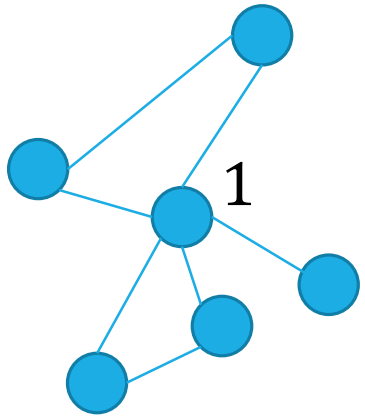
What About Pooling?

- Standard aggregation operates of predefined node subsets (whole graph, vertex neighborhoods)
- Ignore community/hierarchical structure in the graph
- Advanced (differentiable) pooling operators

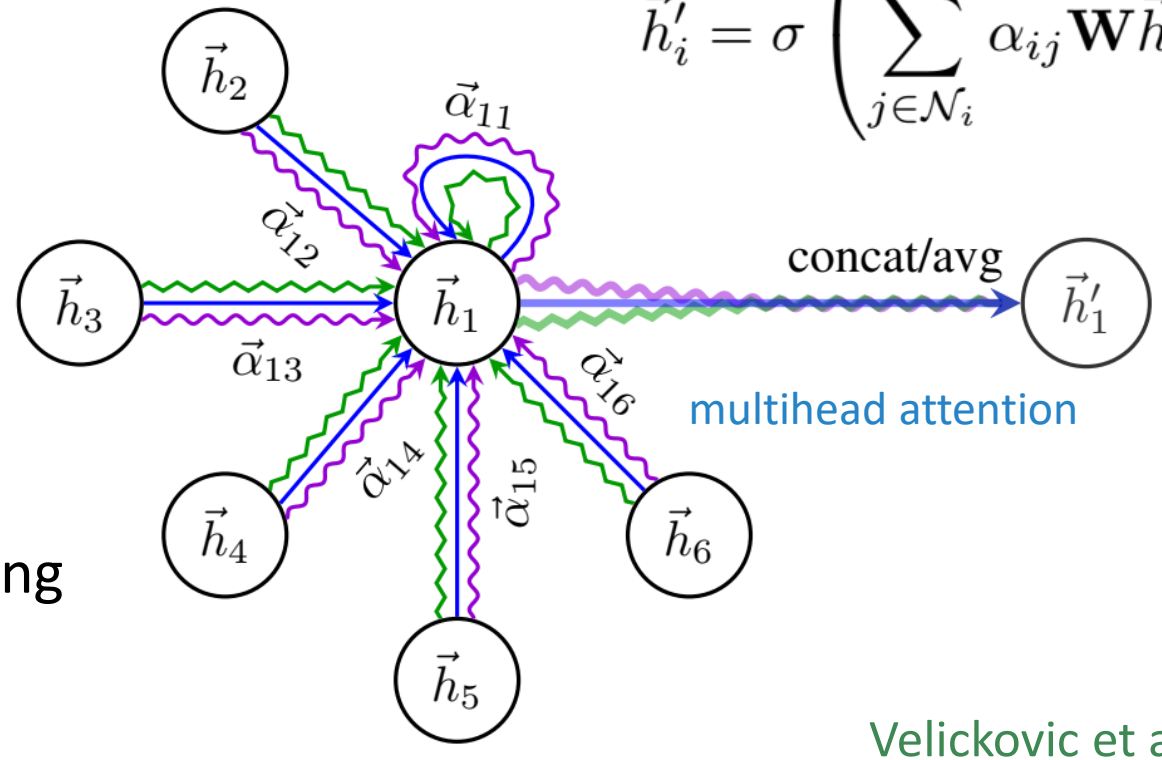


Rex Ying et al, Arxiv 2018

Graph Attention



$$\vec{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} \vec{h}_j\right)$$

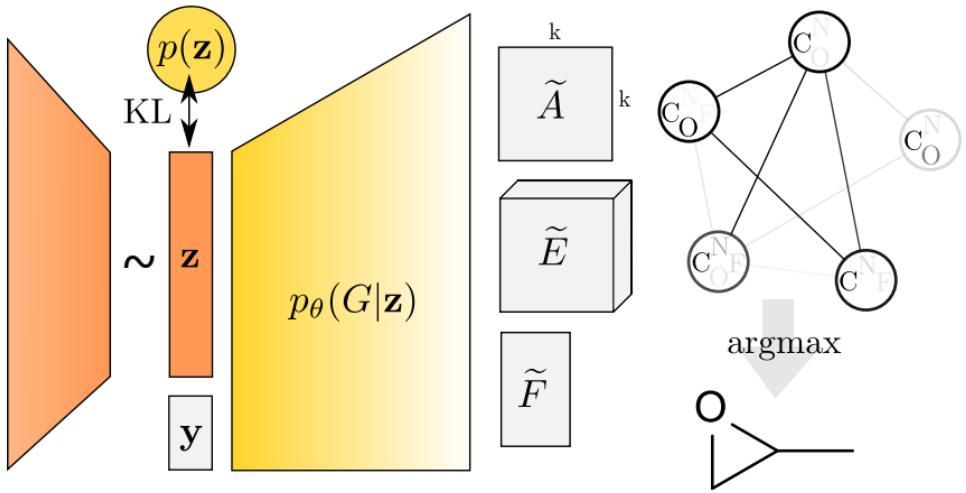


Learning to weight contribution of other nodes when aggregating to form the node embedding

Velickovic et al, ICLR 2018

Graph Generation

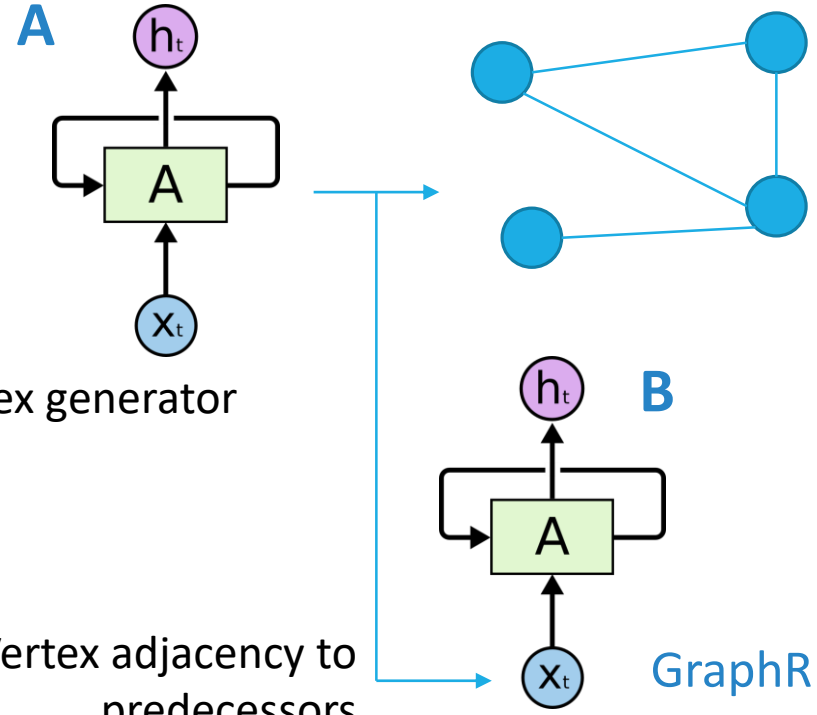
Generate a prediction that is itself a graph



GraphVAE generates adjacency matrix up to k vertices

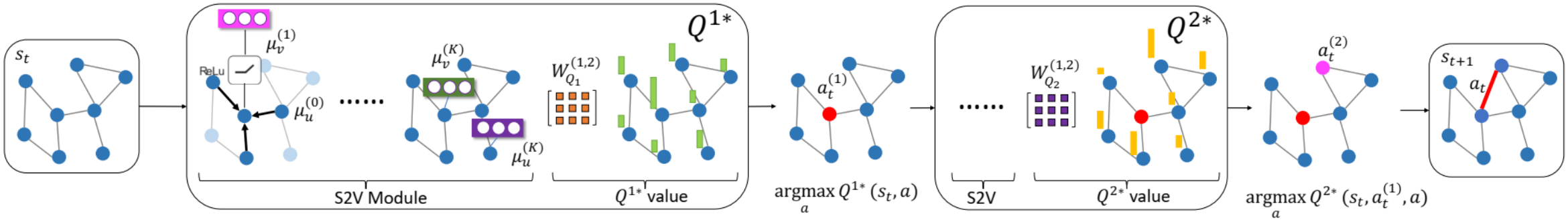
Simonovsky, Komodakis, ICLR-WS 2018

You et al, ICML 2018



Adversarial Attacks

Learn an attack policy by Q-learning (edge addition or deletion)

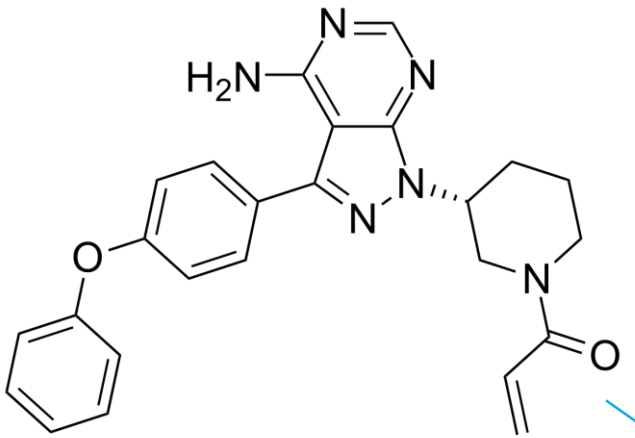


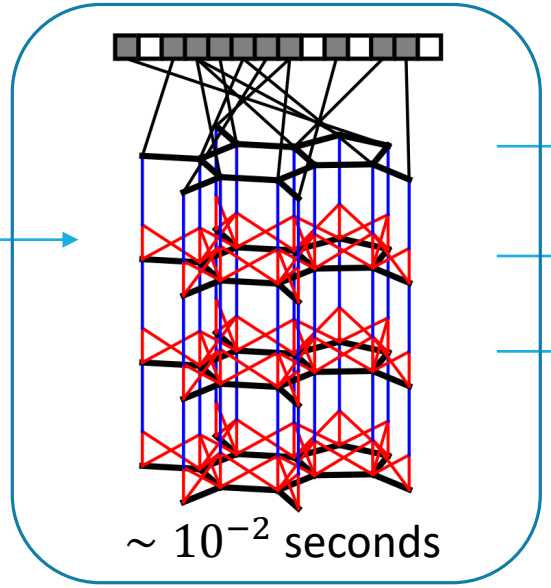
Show GraphRNN vulnerability to both black-box and white-box attacks (attack edges with maximum gradient)

H. Dai et al, ICML 2018

Applications

Predicting Properties of Chemical Compounds





Toxicity

Solubility

Quantum mechanical properties

Simulation methods

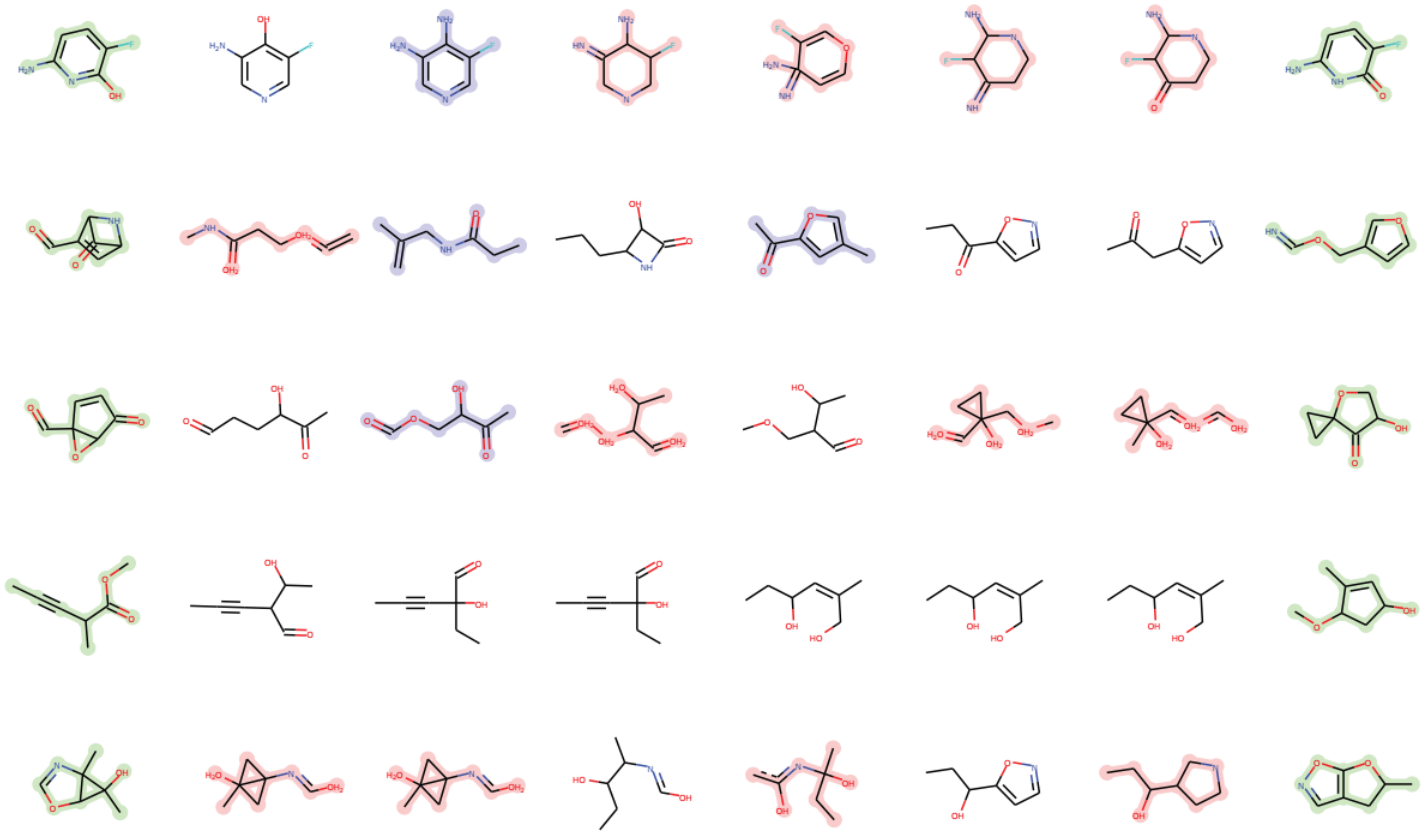
~ 10^{3} seconds

Micheli et al, JCICS 2001

Duvenaud, Maclaurin et al, NIPS 2015

Gilmer et al, ICML 2017

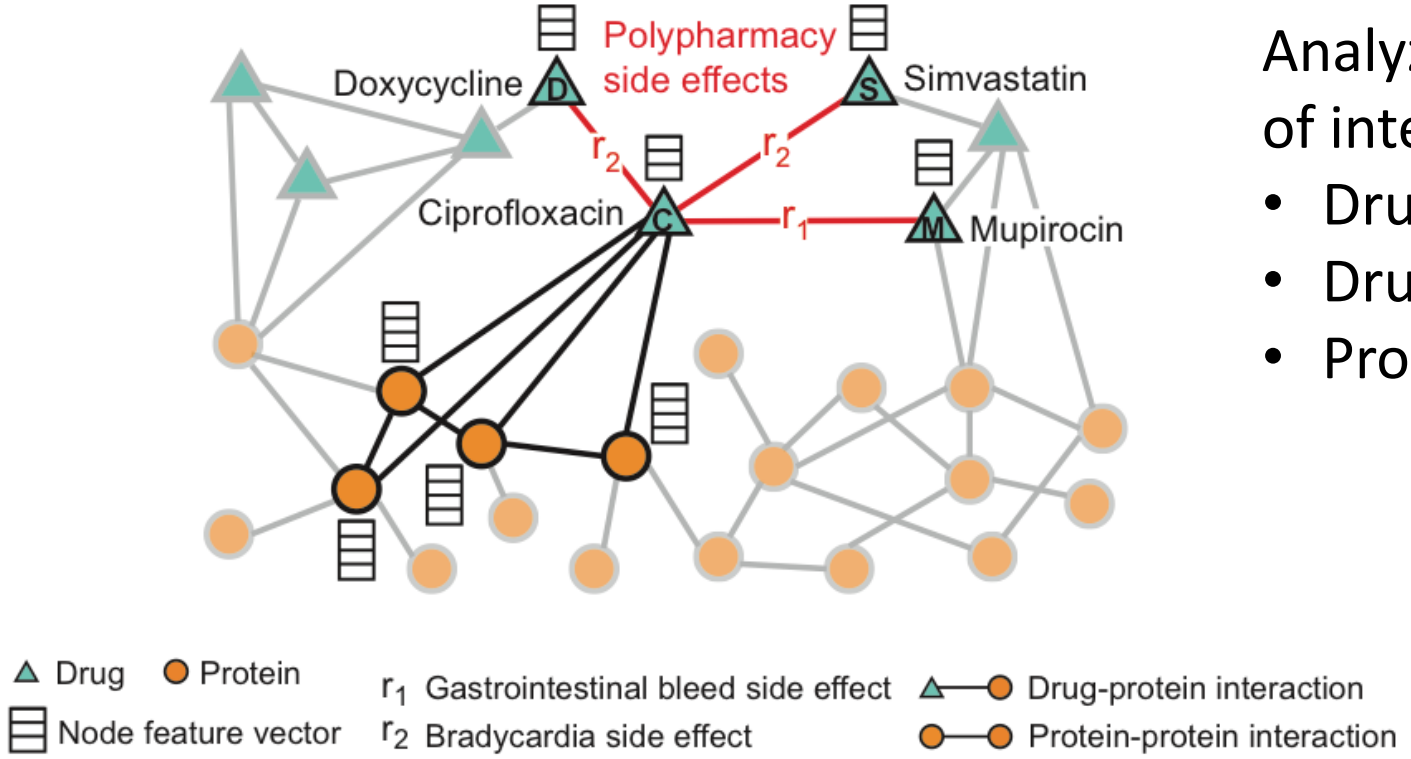
Generating Molecules



Molecule generation by linear interpolation in the latent space of a Graph Variational Autoencoder

Simonovsky, Komodakis, ICLR-WS 2018

Side Effects of Drug Combinations



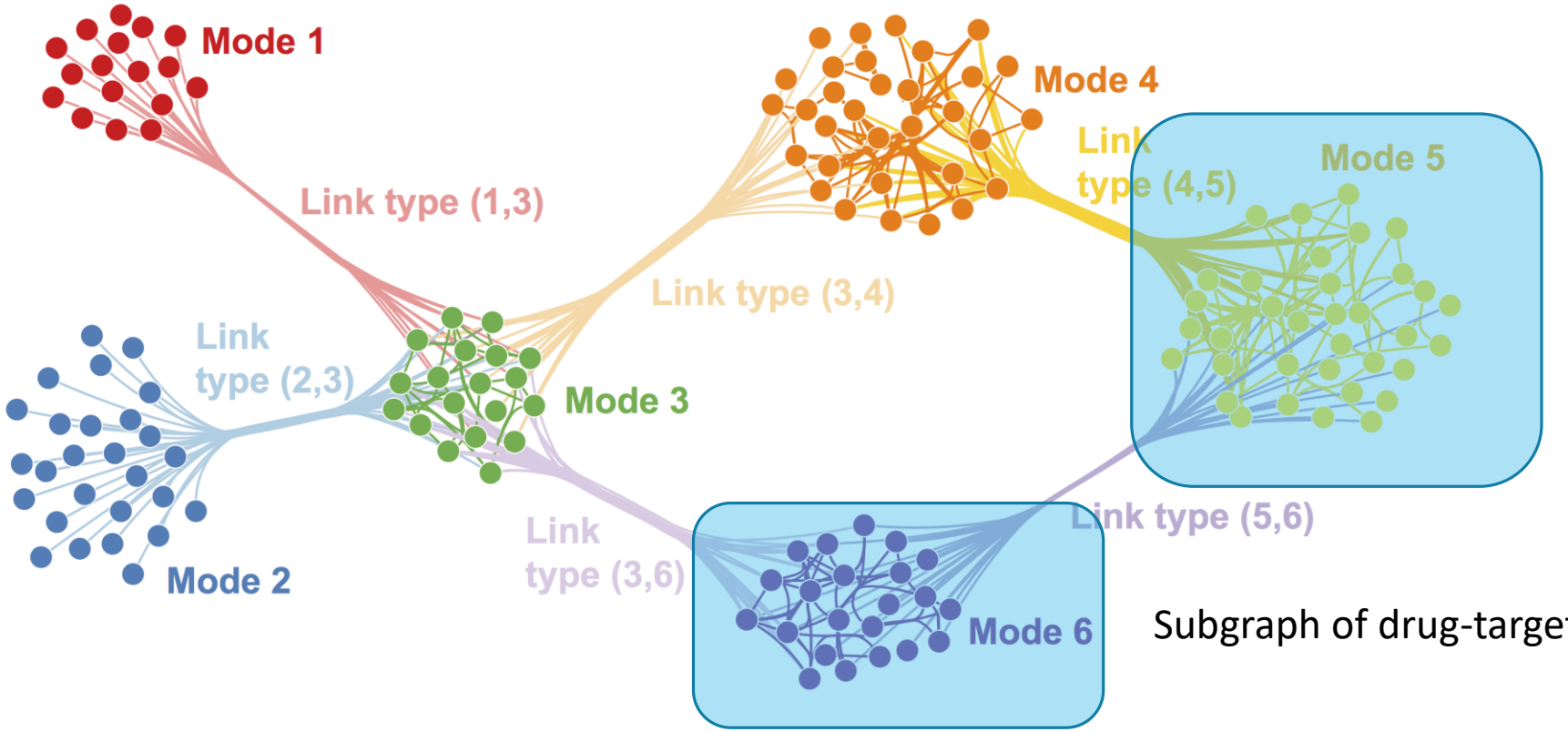
Analyzing a multimodal graph of interactions

- Drug-drug
- Drug-protein
- Protein-protein

Zitnik, Agrawal, Leskovec, Bioinformatics 2018

Drug Repurposing

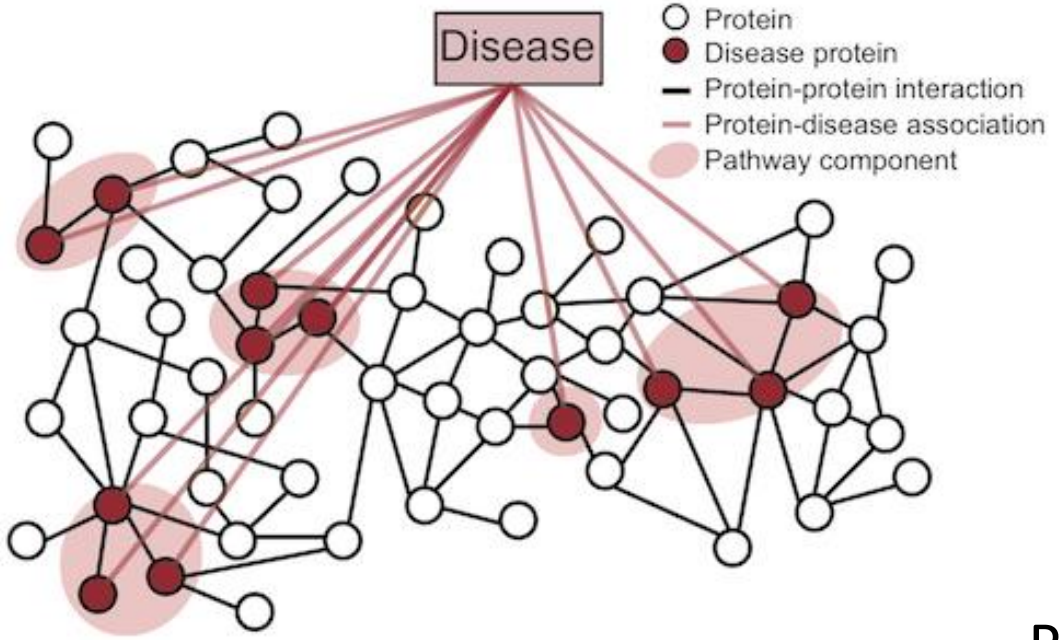
Predict unknown disease-drug interactions

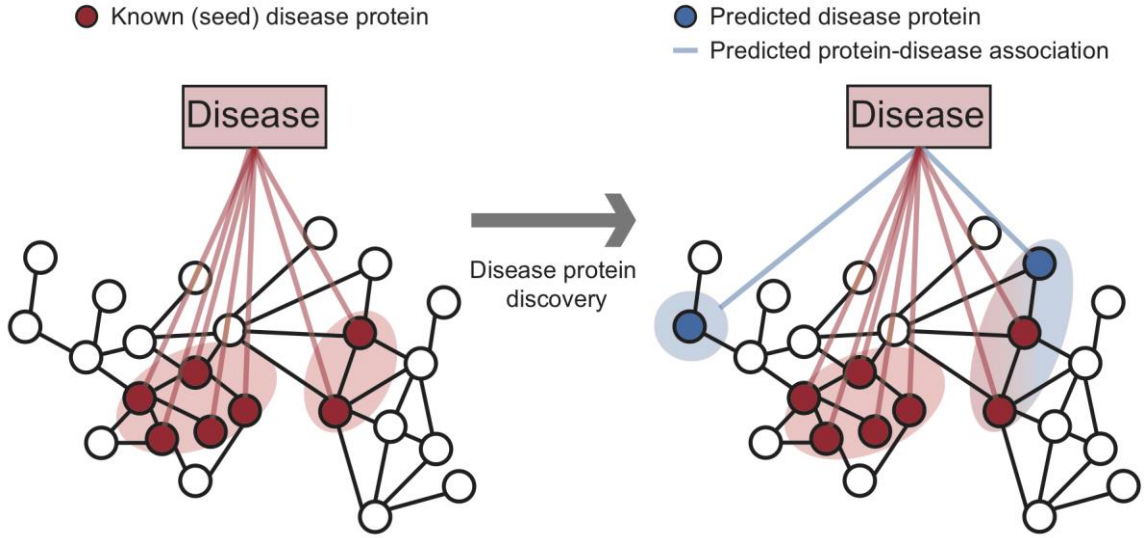


Subgraph of disease associated protein-protein interactions

Subgraph of drug-target proteins

Pathway Prediction

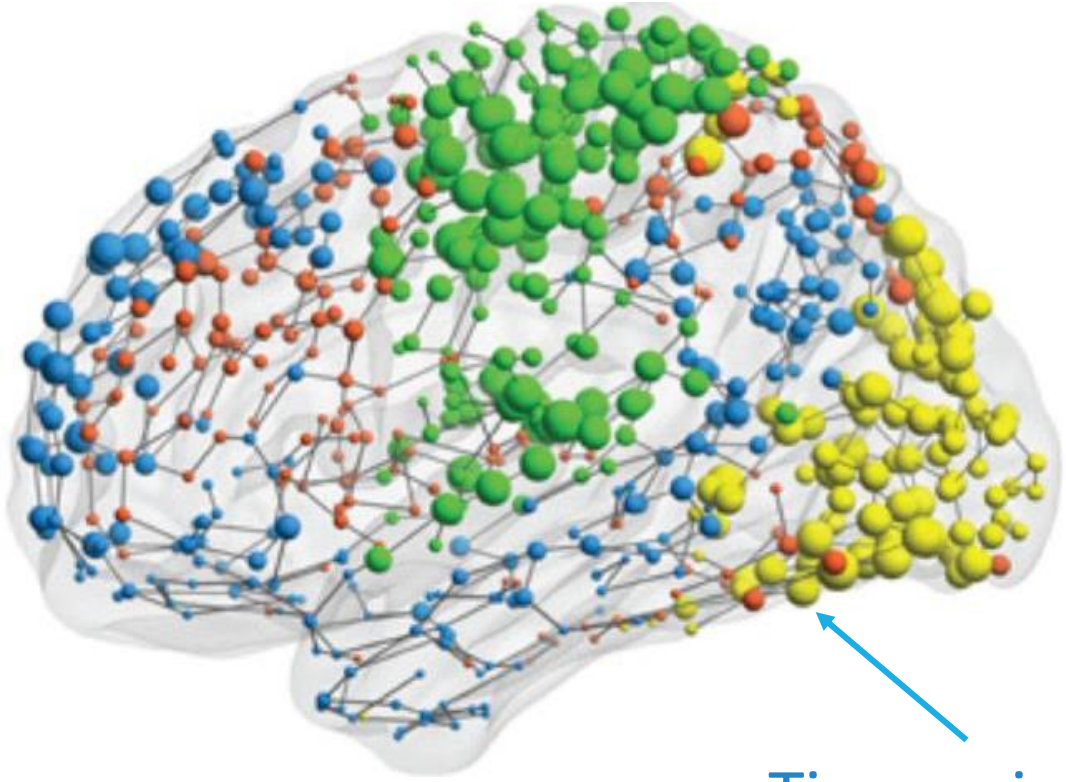




Predicting unknown protein-disease associations from interaction networks

Zitnik, Agrawal, Leskovec, PSB 2018

Functional Brain Images



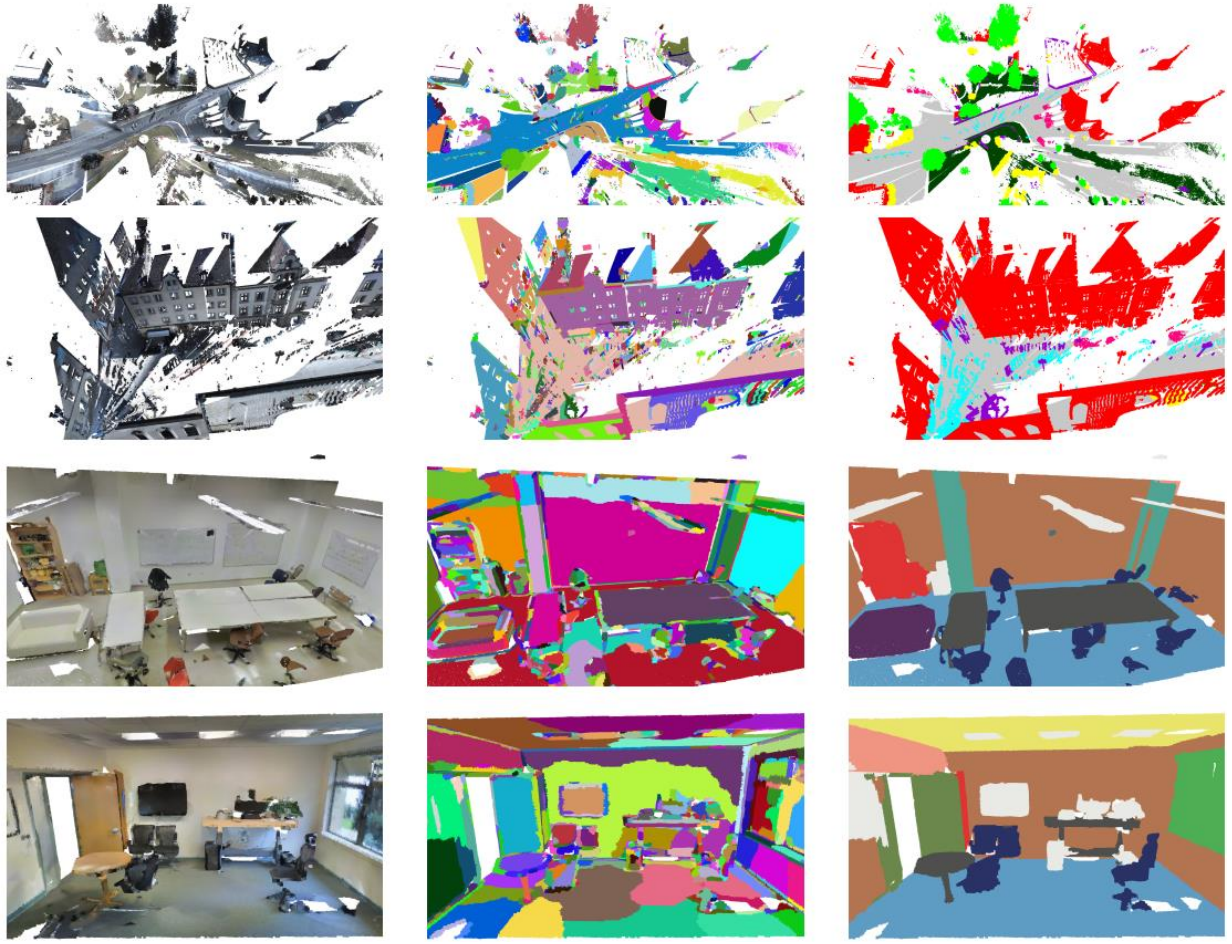
Brain state prediction

Disease prediction

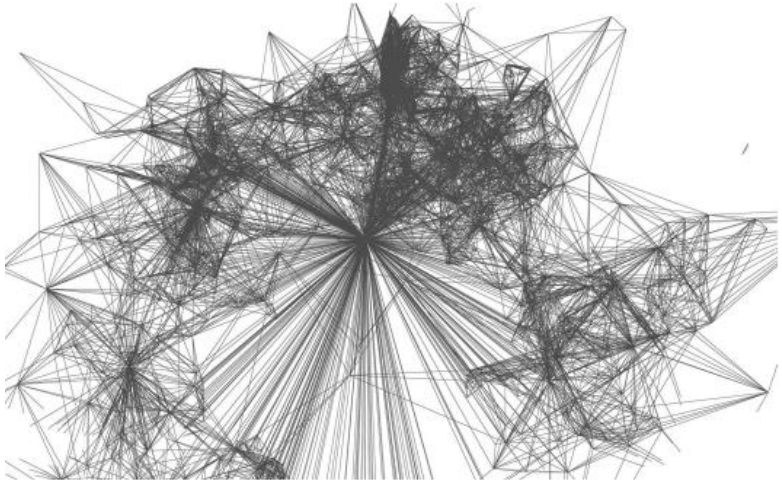
Brain reading

Timeseries data attached to nodes

Point Clouds – Semantic Segmentation



Build point cloud graphs and train semantic class predictors based on vertex embeddings

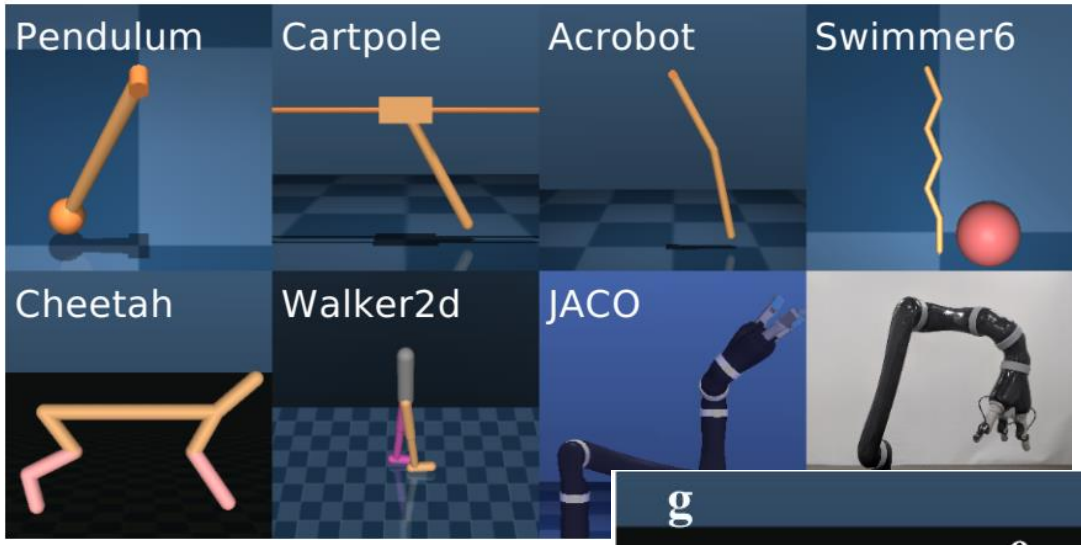


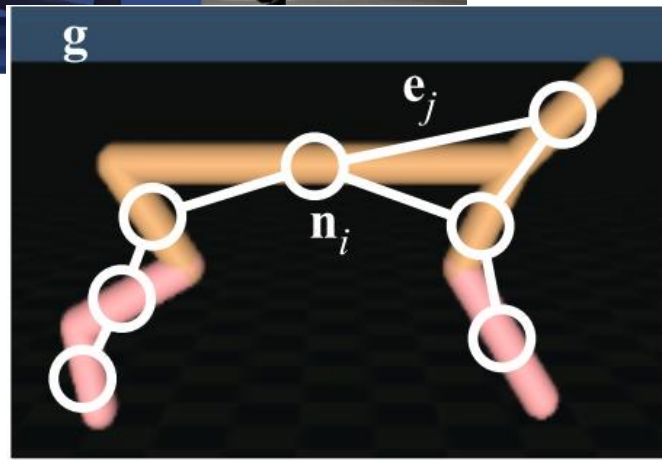
Landrieu, Simonovsky, CVPR 2018

Object Part Prediction

Te et al, ACM MultiMedia, 2018

Graphs in Physics



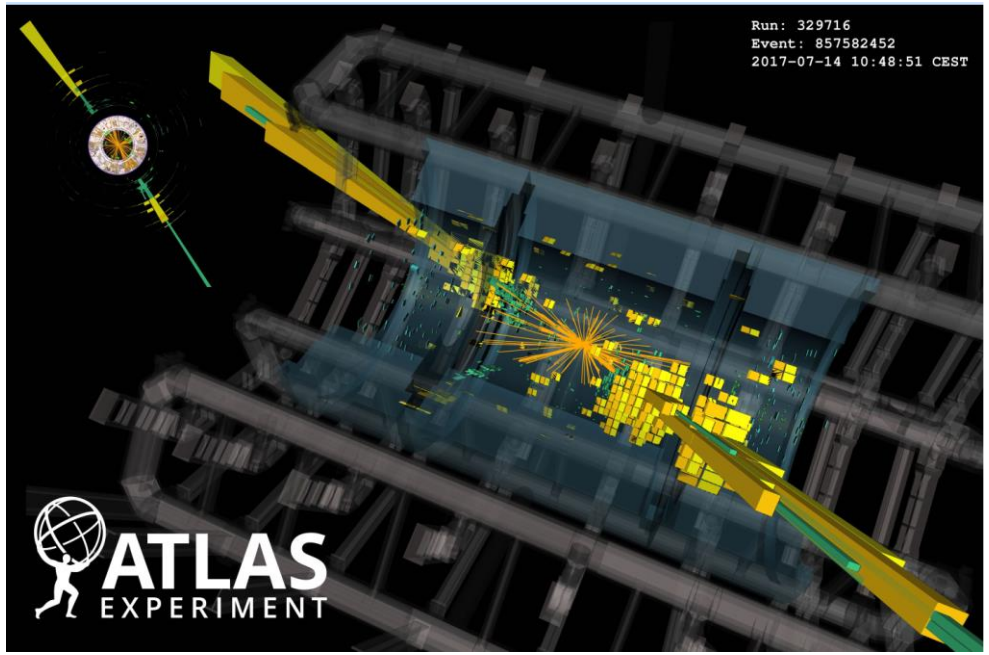


Learning dynamics of a physical system through a graph-based representation

Sanchez-Gonzalez et al, ICML 2018

Particle interactions represented as graphs

Henrion et al, NIPS WS 2017



Conclusions

- Deep learning for graphs is an exploding research topic
  - Many works sharing same underlying idea (adjacency, contractive, contextual)
  - Much early work left unacknowledged
- What should we focus on?
  - Unifying frameworks for graph processing (neural message passing)
  - Theoretical characterization (VC dimension)
  - Efficiency and efficacy of context propagation (unsupervised, gradient issues)
  - Killer applications still missing
  - Research directions (pooling, generative)

Bibliography

Spectral Domain Convolutions

1. Joan Bruna, Wojciech Zaremba, Arthur Szlam, Yann LeCun , Spectral Networks and Locally Connected Networks on Graphs, ICLR 2014
2. Mikael Henaff, Joan Bruna, Yann LeCun, Deep Convolutional Networks on Graph-Structured Data, Arxiv 2015
3. Michaël Defferrard, Xavier Bresson, Pierre Vandergheynst, Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, NIPS 2016
4. Thomas N. Kipf, Max Welling, Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017

Spatial Domain Convolutions

1. David Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, Ryan P. Adams, Convolutional Networks on Graphs for Learning Molecular Fingerprints, NIPS 2015
2. Mathias Niepert, Mohamed Ahmed, Konstantin Kutzkov, Learning Convolutional Neural Networks for Graphs , ICML 2016

Bibliography

Contextual Approaches

1. Alessio Micheli, Neural Network for Graphs: A Contextual Constructive Approach. IEEE TNN, 2009
2. Yujia Li, Daniel Tarlow, Marc Brockschmidt, Richard Zemel , Gated Graph Sequence Neural Networks, ICLR 2016
3. William L Hamilton, Rex Ying, Jure Leskovec, Inductive Representation Learning on Large Graphs, NIPS 2017.
4. Davide Bacciu, Federico Errica, Alessio Micheli , Contextual Graph Markov Model: A Deep and Generative Approach to Graph Processing, ICML 2018

Bibliography

Miscellanea

1. Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, George E. Dahl, Neural Message Passing for Quantum Chemistry, ICML 2017 (Framework)
2. Martin Simonovsky, Nikos Komodakis, GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders, NIPS Workshop, 2017 (Graph Generation)
3. Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, Jure Leskovec, GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models, ICML 2018 (Graph Generation)
4. Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, Yoshua Bengio, Graph Attention Networks, ICLR 2018 (Attention)
5. Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, Le Song, Adversarial Attack on Graph Structured Data, ICML 2018 (Adversarial Attacks)

Contact and Ack's

Deep Learning for Graphs

- (Web) https://sites.google.com/view/dl4sd

Contextual graph Markov Model

- (Code) https://github.com/diningphil/CGMM
- (Paper) http://arxiv.org/abs/1805.10636

Get in touch with me

- (Email) bacciu@di.unipi.it
- (Web) http://www.di.unipi.it/~bacciu/

New tutorial on DL4Graphs

INNS BIG DATA AND DEEP LEARNING 2019
April 16 – 18, Sestri Levante, Italy

https://innsbddl2019.org/

Research funded by MIUR-SIR project LIST-IT (grant n. RBSI14STDE)

Computational Intelligence &
Machine Learning Group

Deep Learning for Graphs

DAVIDE BACCIU (BACCIU@DI.UNIPI.IT) – ALESSIO MICHELI (MICHELI@DI.UNIPI.IT)

DIPARTIMENTO DI INFORMATICA - UNIVERSITA' DI PISA

https://sites.google.com/view/dl4sd