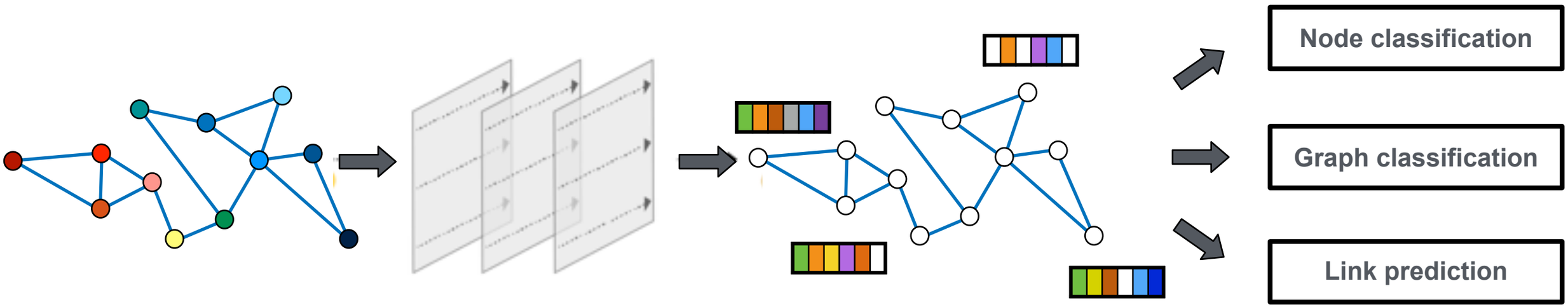
Graph neural networks: State-of-the-art architectures

Dr Dorina Thanou

May 8, 2023

Recap: Graph neural networks (GNNs)

- A different way of obtaining ‘deeper’ embeddings inspired by deep learning
- They generalize to graphs with node attributes



Graph and node attributes

$\mathcal{G}, X$

GNN blocks
(graph convolution, pooling, nonlinearity)

Transformed graph
(node/edge embeddings)

$\mathcal{G}', H$

Classification layer

End tasks
(node, edge, graph level)

Y

Recap of previous lecture

- Graph neural networks are designed based on a generalization of deep learning techniques on graphs

- Their key ingredients are:
  - Parameter sharing across nodes
  - Local exchange of information and neighbourhood aggregation on the graph (e.g., graph convolution)
  - Graph subsampling/pooling (mainly used for graph level tasks)

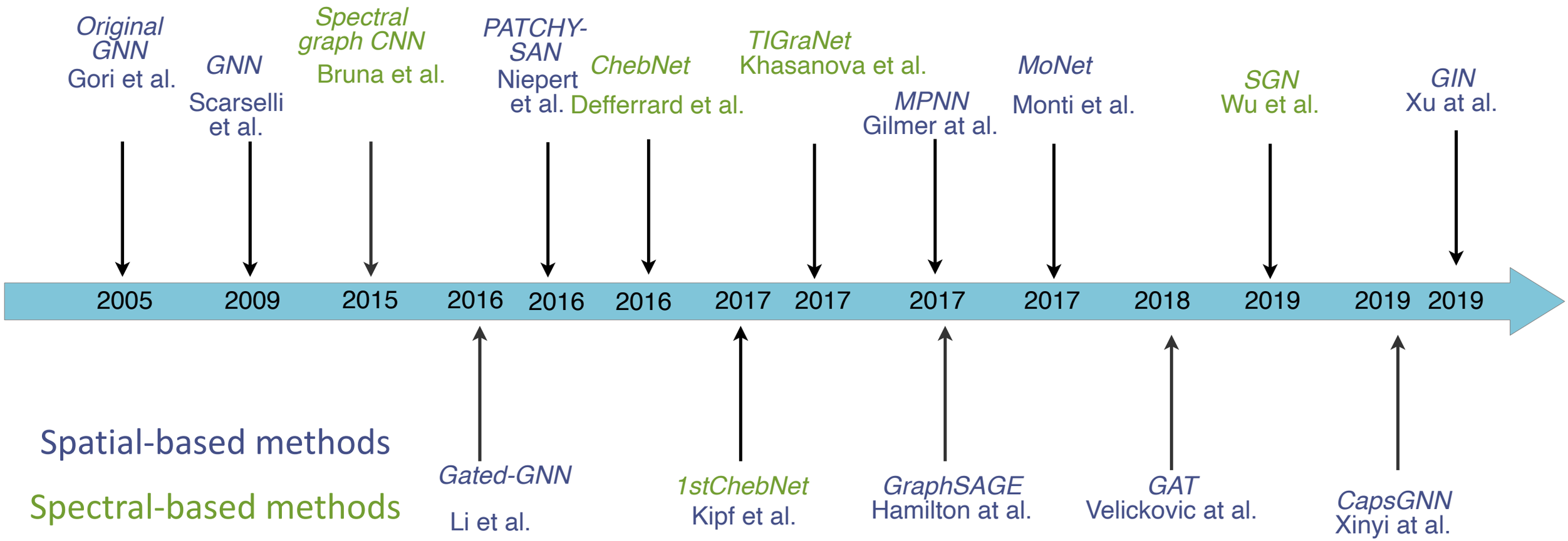
- Two different ways of defining graph convolution:
  - A spectral viewpoint
  - A spatial viewpoint

Outline

- State-of-the-art GNN architectures

  - Spectral approaches
  - Spatial approaches

First GNN architectures



- Recent trends
  - Spectrally-inspired architectures: GraphHeat (Xu'19), GWNN (Xu'19), SIGN (Frasca'20), DGN (Beaini'20), Framelets (Zheng'21), FAGCN (Bo'21)
  - More expressive GNNs: higher order WL test (Maron'19, Morris'20), physics-inspired GNNs (Chamberlain'21), and many more!

Outline

- State-of-the-art GNN architectures

  - **Spectral approaches**
  - Spatial approaches

Spectral approaches in one slide

- Convolution is defined in the graph Fourier domain (see previous lecture)

$$x *_\theta g = \chi \hat{g}(\theta) \chi^T x$$

- **Spectral GCNN:** $\hat{g}(\lambda) = \theta$ ⇒ $x * g = \chi \theta \chi^T x$

- **ChebNet:** $\hat{g}(\lambda) = \sum_{k=0}^{K} \theta_i T_i(\lambda)$ ⇒ $x * g = \sum_{k=0}^{K} \theta_i T_i(L) x$

- **GCN:** $K = 1$ ⇒ $x * g = (\theta_0 - \theta_1 D^{-1/2} W D^{-1/2}) x$

- Parameters θ are learned through the network

Spectral GCNN

- **Main intuition:** Define graph convolutional filters in the spectral domain

$$x * g = \chi \hat{g}(\Lambda) \chi^T x = \hat{g}(L) x$$

- A GCNN is defined as a series of spectral convolutional layers:

$$h^{l+1} = \sigma(\chi \theta^{l+1} \chi^T h^l)$$

Non-linearity

- $\theta^{l+1} = diag(\theta_{\lambda_1}, \theta_{\lambda_2}, ..., \theta_{\lambda_N})$ are **learnable spectral coefficients** in each layer
- First spectral graph CNN architecture
- $O(N)$ parameters per layer
- $O(N^2)$ for the computation of the GFT, IGFT
- Filters are eigenbasis dependent: they do not generalize across graphs

[Bruna et al., Spectral Networks and Deep Locally Connected Networks on Graphs, ICLR 2014]

Parametrized spectral graph filters

- **Main intuition:** Represent **spectral filters with polynomials** of the Laplacian

$$\widehat{g^{l+1}}(\lambda) = \sum_{k=0}^{K} \theta_k^{l+1} \lambda^k$$

- Monomials:

$$h^{l+1} = \sigma(\widehat{g^{l+1}}(L)h^l) = \sigma(\sum_{k=0}^{K} \theta_k^{l+1} L^k h^l) = \sigma(\sum_{k=0}^{K} \theta_k^{l+1} z_k)$$

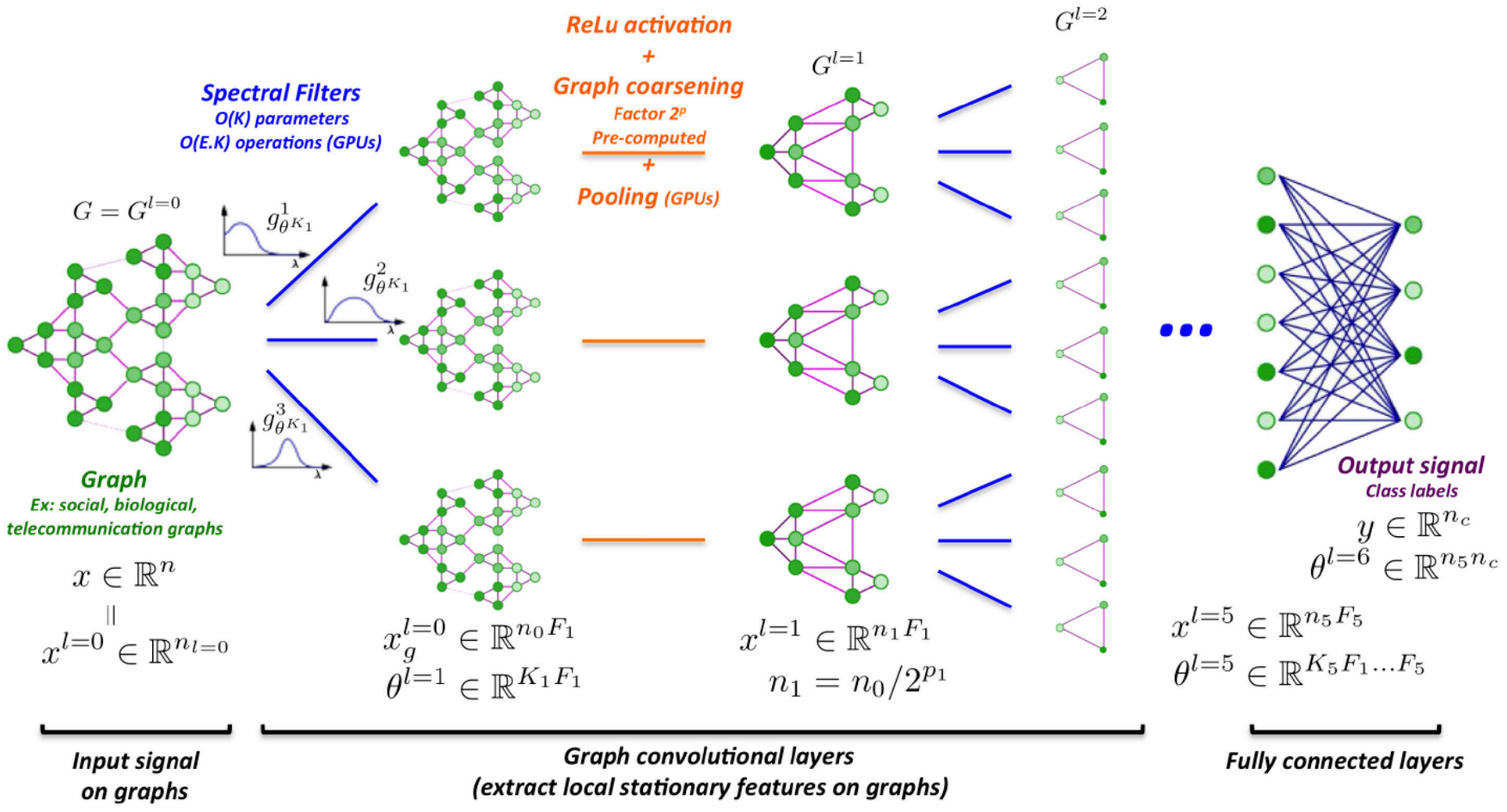
  - with $z_0 = 1, z_1 = Lh^l, \ldots, z_{k+1} = Lz_k$

- Chebyshev polynomials:

$$h^{l+1} = \sigma(\widehat{g^{l+1}}(L)h^l) = \sigma(\sum_{k=0}^{K} \theta_k^{l+1} T_k(L)h^l) = \sigma(\sum_{k=0}^{K} \theta_k^{l+1} z_k)$$

  - with $z_0 = 1, z_1 = Lh^l, \ldots, z_{k+1} = 2Lz_k - z_{k-1}$

ChebNet



- Coefficients of polynomials are learnable; $O(1)$ parameters per layer
- Filters have K-hops support
- No explicit computation of the eigenvectors of the Laplacian

[Defferrard et al., Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, NeurIPS 2016]

Graph Convolutional Networks (GCN)

- **Main intuition:** Design a scalable architecture with **first-order approximation** of spectral graph convolution

$$g * x \approx \theta_0 x + \theta_1 (L - I_N) x = \theta_0 x - \theta_1 D^{-1/2} W D^{-1/2} x$$

$$\theta = \theta_0 = -\theta_1$$

$$g * x \approx \theta (I + D^{-1/2} W D^{-1/2}) x$$

$$\tilde{W} = W + I_N$$

$$g * x \approx \theta \tilde{D}^{-1/2} \tilde{W} \tilde{D}^{-1/2} x$$

Adjacency/Weight matrix

Degree matrix

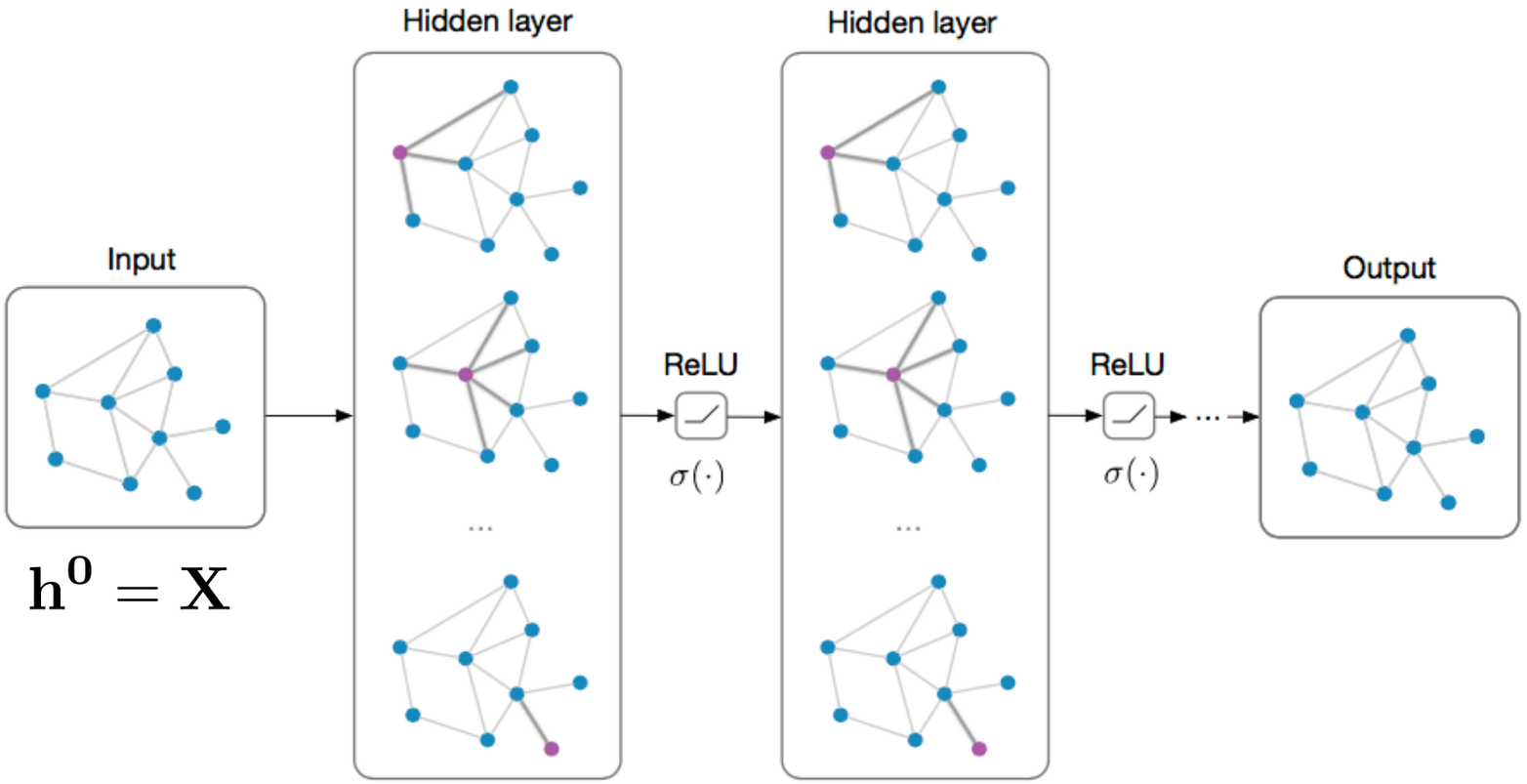
- A convolutional layer is defined as:

$$h^{l+1} = \sigma(\tilde{D}^{-1/2} \tilde{W} \tilde{D}^{-1/2} h^l \theta^{l+1})$$

Learned parameters

GCN architecture

- Very often, it consists of two GCN layers

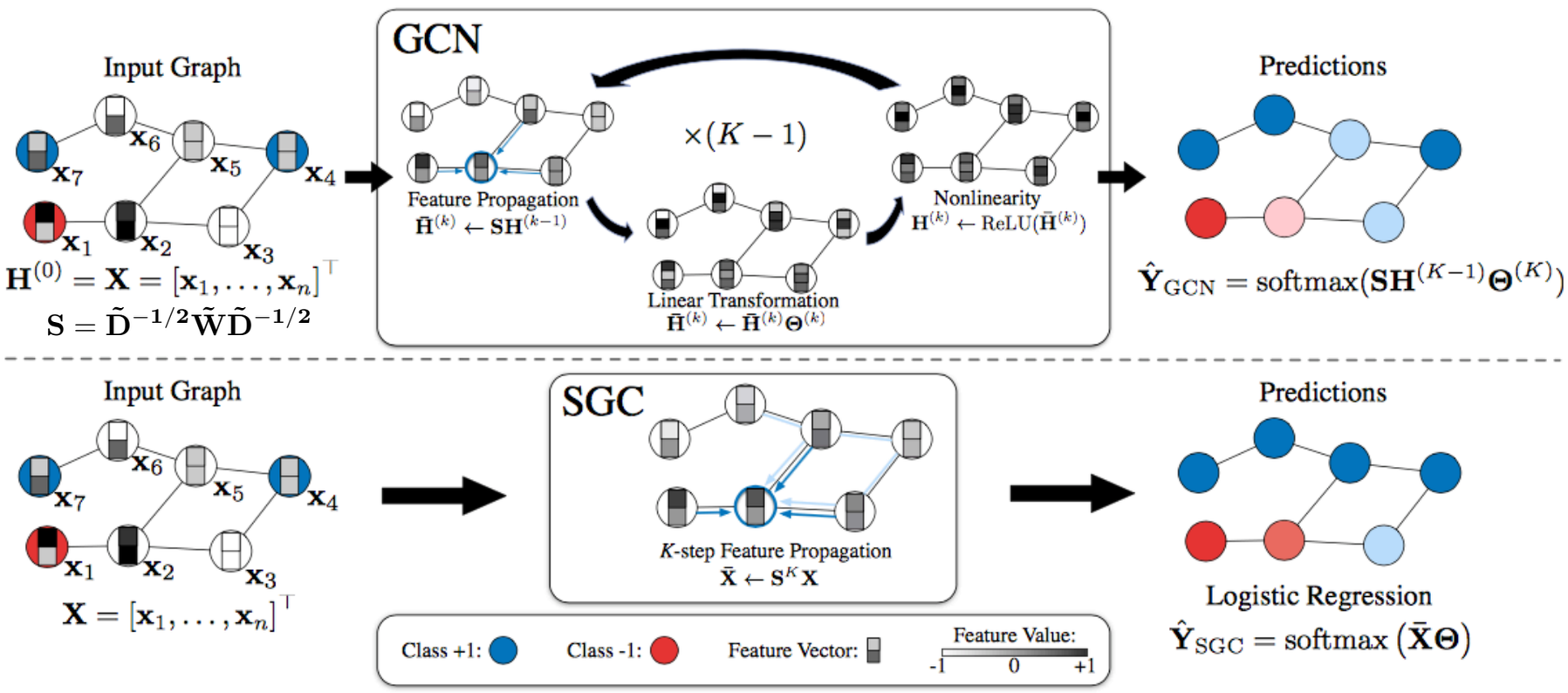


$$\mathbf{h^{l+1}} = \sigma(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{W}}\tilde{\mathbf{D}}^{-1/2}\mathbf{h^l}\theta^{l+1})$$

[Kipf et al., Semi-Supervised Classification with Graph Convolutional Networks, ICLR, 2017]

Simple Graph Convolution (SGC)

- **Main intuition:** Reduce further the complexity of GCN by removing nonlinearities



- Reduces the procedure to a simple low pass filtering
- No significant loss in the performance

[Wu et al., Simplifying Graph Convolutional Networks, ICML 2019]

Summary of spectral methods

- **Pros:**
  - Clear intuition from the spectral domain
  - They take into account the global structure of the graph

- **Cons:**
  - Generalization is an open research question

Outline

- State-of-the-art GNN architectures

  - Spectral approaches
  - **Spatial approaches**

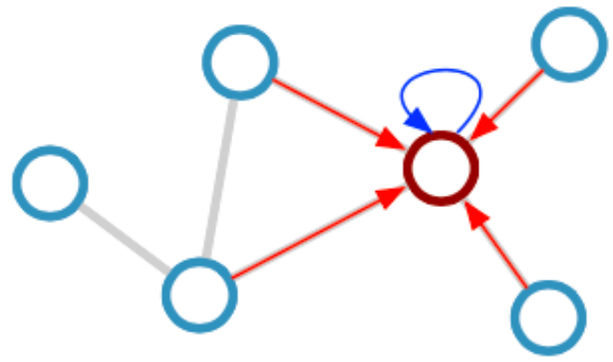
Spatial approaches in one slide

- Generate node embeddings based on local neighborhoods
  - Nodes aggregate information from their neighbors using a permutation-invariant function
  - The feature vector of each node is updated based on its current values and the aggregated neighbourhood representation

- Feed the embeddings into a loss function (usually task specific)

- **Key difference between architectures:** How nodes aggregate information across layers

$$h_i^{l+1} = \sigma(U^l h_i^l + M^l \sum_{j \in \mathcal{N}_i} h_j^l)$$



Spatial approaches in one slide

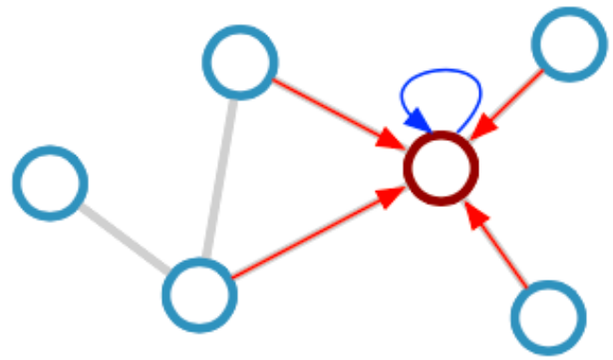
- Generate node embeddings based on local neighborhoods
  - Nodes aggregate information from their neighbors using a permutation-invariant function
  - The feature vector of each node is updated based on its current values and the aggregated neighbourhood representation

- Feed the embeddings into a loss function (usually task specific)

- **Key difference between architectures:** How nodes aggregate information across layers

$$h_i^{l+1} = \sigma(U^l h_i^l + M^l \sum_{j \in \mathcal{N}_i} h_j^l)$$

Trainable parameters



Message Passing Neural Network (MPNN)

- **Main intuition:** Each node exchange messages with its neighbors and update its representations based on these messages

- The message passing scheme runs for T time steps and updates the representation of each vertex based on its previous representation and the representation of its neighbors

$$m_i^{l+1} = \sum_{j \in \mathcal{N}_i} M_l(h_i^l, h_j^l, e_{ij})$$

Message function

$$h_i^{l+1} = U_l(h_i^l, m_i^{l+1})$$

Vertex update function

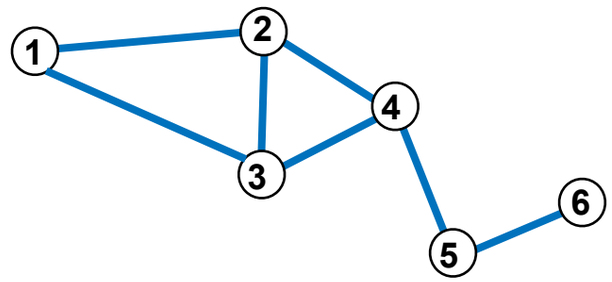
Learned differentiable functions!

[Gilmer et al, Neural Message Passing for Quantum Chemistry, ICML, 2017]

MPNN - Example

- At each iteration, the embeddings are updated as follows:

$$
\begin{aligned}
h_1^{l+1} &= \theta_0^l h_1^l + \theta_1^l h_2^l + \theta_1^l h_3^l \\
h_2^{l+1} &= \theta_0^l h_2^l + \theta_1^l h_1^l + \theta_1^l h_3^l + \theta_1^l h_4^l \\
h_3^{l+1} &= \theta_0^l h_3^l + \theta_1^l h_1^l + \theta_1^l h_2^l + \theta_1^l h_4^l \\
h_4^{l+1} &= \theta_0^l h_4^l + \theta_1^l h_2^l + \theta_1^l h_3^l + \theta_1^l h_5^l \\
h_5^{l+1} &= \theta_0^l h_5^l + \theta_1^l h_4^l + \theta_1^l h_6^l \\
h_6^{l+1} &= \theta_0^l h_6^l + \theta_1^l h_5^l
\end{aligned}
$$



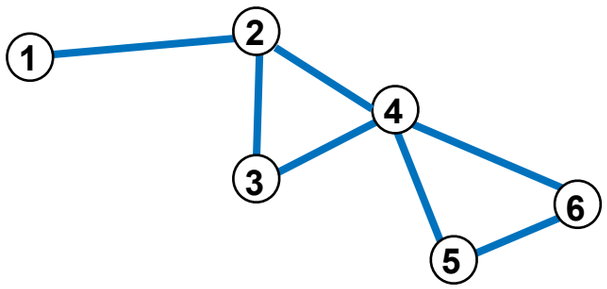
- The output of the message passing is:

$$\{h_1^{l_{max}}, h_2^{l_{max}}, h_3^{l_{max}}, h_4^{l_{max}}, h_5^{l_{max}}, h_6^{l_{max}}\}$$

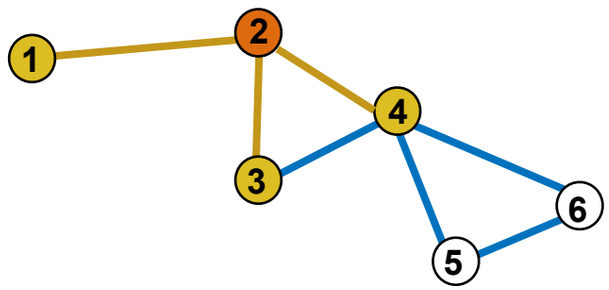
A local view of MPNNs

- MPNNs focus on recursive neighbourhood aggregation: they propagate messages on the edges for a fixed number of iterations layers

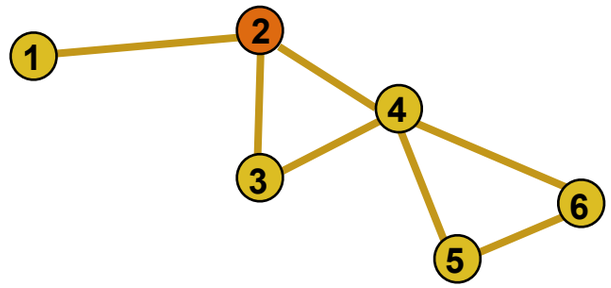
- After k iterations (or layers) the embedding of each node is affected by the node's k-hop neighborhood, i.e., the subgraph containing k-hops nodes



Initial graph



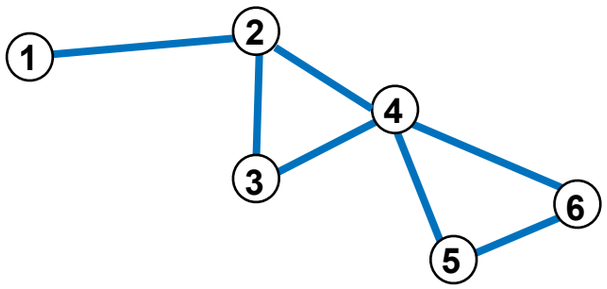
Receptive field at layer 1

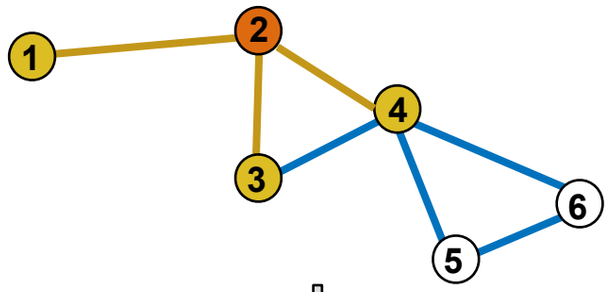


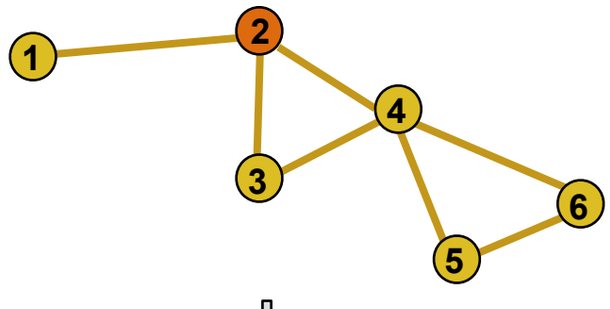
Receptive field at layer 2

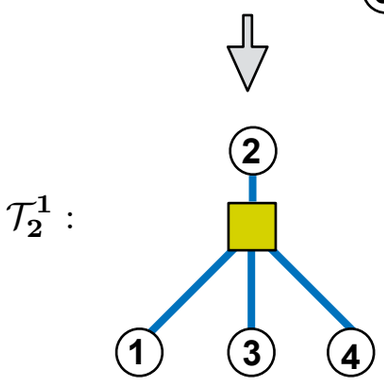
From graphs to rooted subtrees

- Each subgraph can be mapped to a rooted subtree or a subtree pattern
- The maximum depth of the subtree is defined by the number of layers

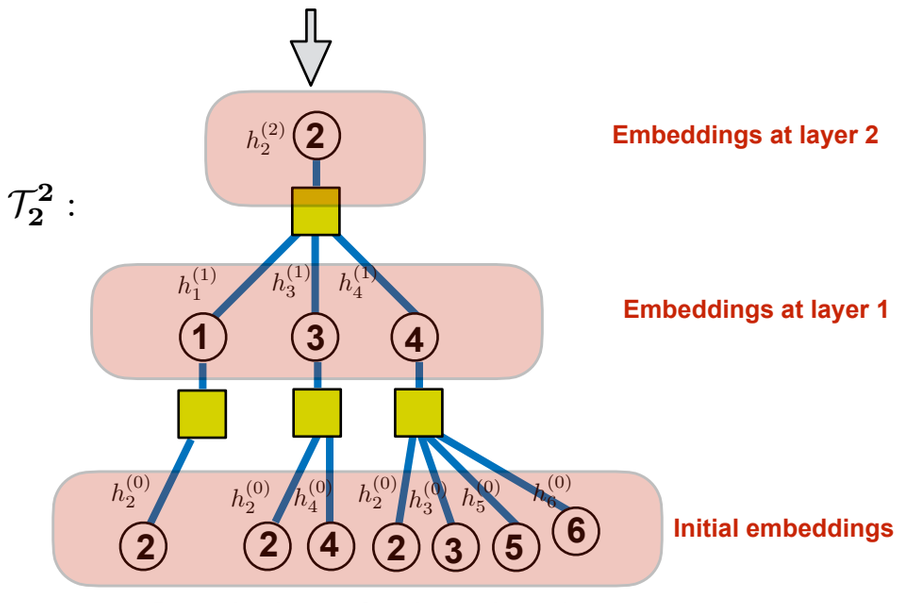








Rooted subtree of node 2 at layer 1

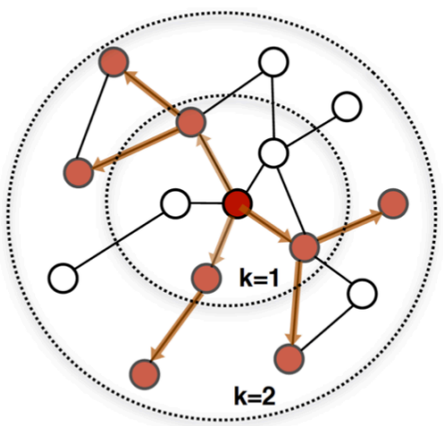


Rooted subtree of node 2 at layer 2

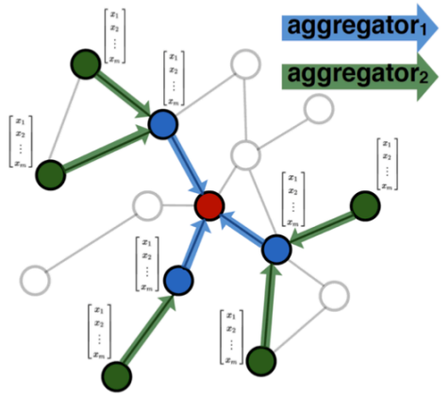
Different rooted subtrees should be assigned different node embeddings!

GraphSAGE

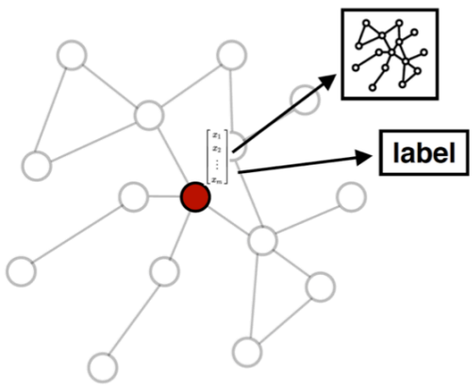
- **Main intuition:** Subsample the neighborhood for better scalability; Node's neighborhood defines a computational graph
- Aggregate the feature information from sample neighbors
- Different loss functions: Cross entropy, Hinge,…
- Usually 2-3 layers deep



1. Sample neighborhood



2. Aggregate feature information from neighbors



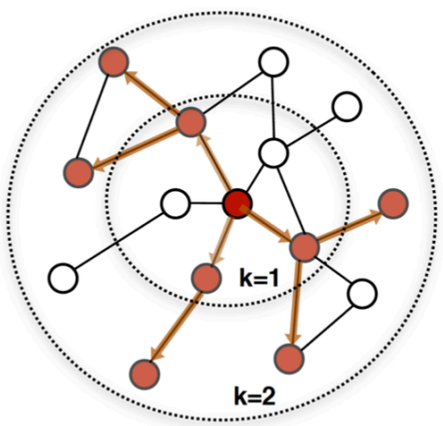
3. Predict graph context and label using aggregated information

$$h_i^{l+1} = ReLU\left(\theta_0^l h_i^l, \bigoplus_{j \in \mathcal{N}_i} ReLU(\theta_1^l h_j^l)\right)$$

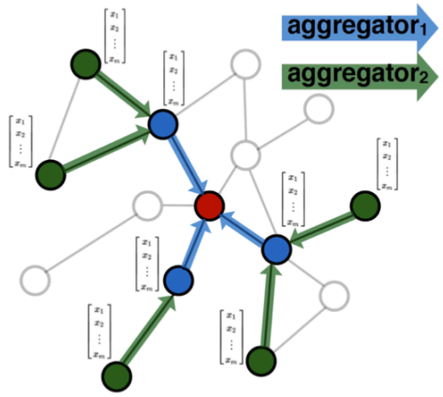
[Hamilton et al, Inductive Representation Learning on Large Graphs, NeurIPS, 2017]

GraphSAGE

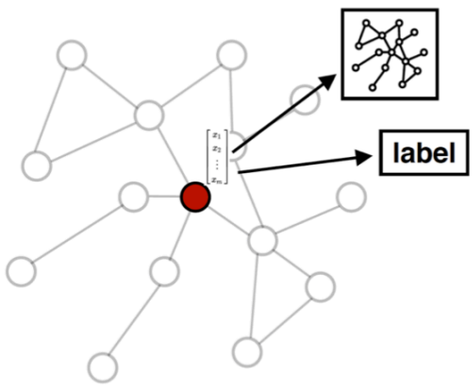
- **Main intuition:** Subsample the neighborhood for better scalability; Node's neighborhood defines a computational graph
- Aggregate the feature information from sample neighbors
- Different loss functions: Cross entropy, Hinge,…
- Usually 2-3 layers deep



1. Sample neighborhood



2. Aggregate feature information from neighbors



3. Predict graph context and label using aggregated information

$$h_i^{l+1} = ReLU\left(\theta_0^l h_i^l \oplus_{j \in \mathcal{N}_i} ReLU(\theta_1^l h_j^l)\right)$$

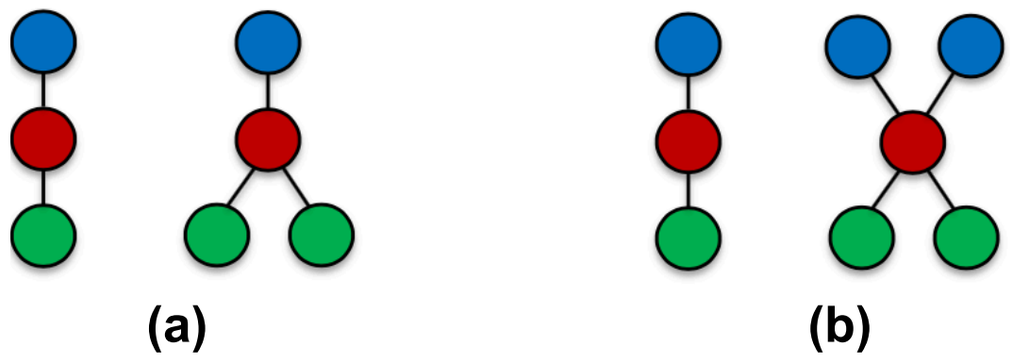
Trained aggregation function: Averaging/max pooling/LSTM

[Hamilton et al, Inductive Representation Learning on Large Graphs, NeurIPS, 2017]

Graph Isomorphism Network (GIN)

- **Main intuition:** Maximize the representation power of GNNs
  - Ensure that different rooted subtrees are mapped into different node embeddings invectively
  - Aggregation and readout functions should be injective, e.g., sum

Which aggregators can discriminate the red node?



(a) (b)

(a) max fails; (b) max and mean fails

Sum succeeds in (a) and (b)

- GIN is a maximally-expressive GNN
  - An MLP (universal approximator) can approximate an injective function

$$h_i^{l+1} = \mathrm{MLP}^{l+1}\big((1+\epsilon^l)h_i^l + \sum_{j\in\mathcal{N}_i} h_j^l\big)$$

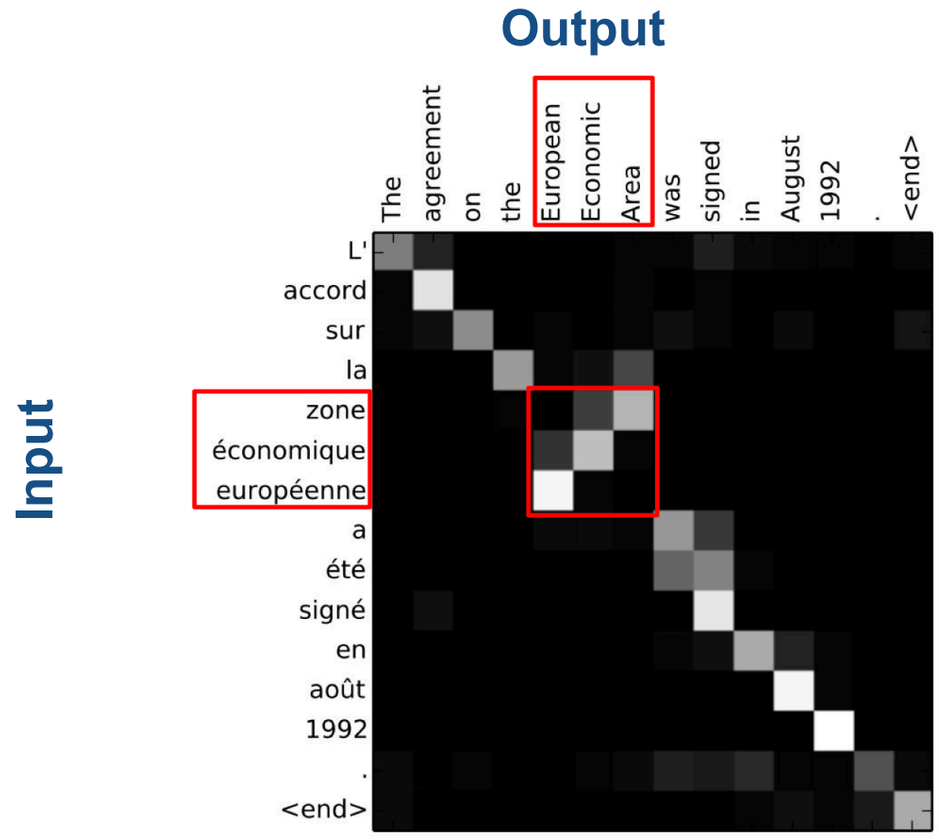
$$h_{\mathcal{G}} = \mathrm{MLP}\big(\sum_{i\in\mathcal{V}} h_i^{l_{max}}\big)$$

[Xu et al., How powerful are graph neural networks, ICLR 2019]

Attention mechanisms on graphs

- Attention is a mechanism through which neural networks dynamically allocate weights to distinct input based on their relevance

- It has shown to be key in tasks such as machine translation and language understanding



Attention weights

How can we extend attention to graph nodes?

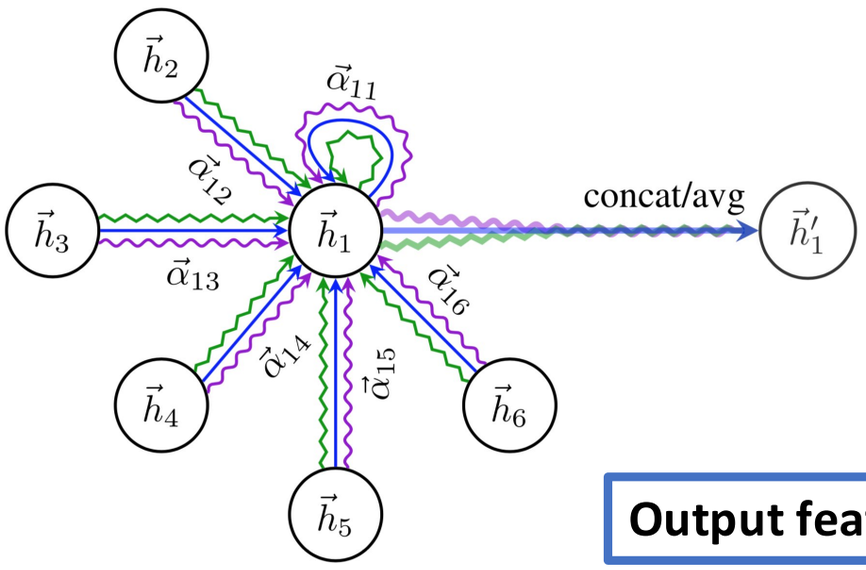
[Bahdanau et al, “Neural machine translation by jointly learning to align and translate”, ICLR 2015]

Graph attention networks (GAT)

- Previous message passing GNNs use fixed weights for incoming information

$$h_i^{l+1} = \sigma\big(\theta_0^l h_i^l + \sum_{j \in \mathcal{N}_i} \theta_1^l h_j^l\big)$$

- In attentional GNNs, weights depend on the neighborhood's features
  - Different weights to different nodes in a neighbourhood
  - Remove dependence on the global graph structure



Output features

Updated embedding

$$h_i^{l+1} = \sigma\big(\alpha_{ii}^l \theta_0^l h_i^l + \sum_{j \in \mathcal{N}_i} \alpha_{ij}^l \theta_1^l h_j^l\big)$$

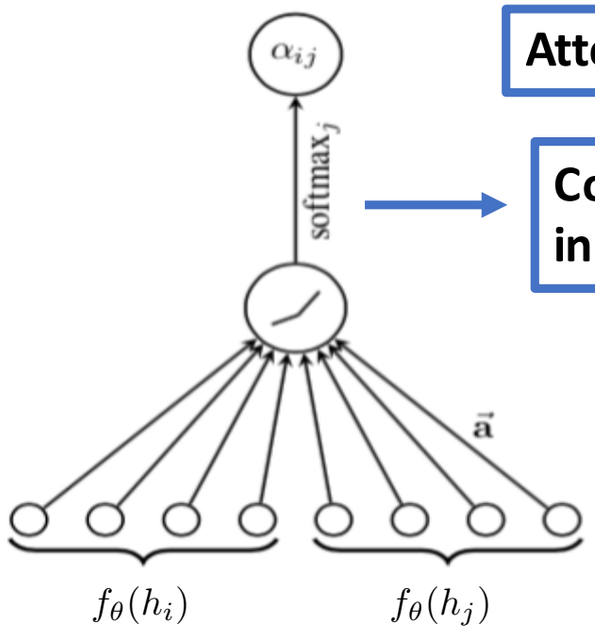
[Velickovic et al., Graph Attention Networks, ICLR 2018]

Graph attention networks (GAT)

- Previous message passing GNNs use fixed weights for incoming information

$$h_i^{l+1} = \sigma\big(\theta_0^l h_i^l + \sum_{j \in \mathcal{N}_i} \theta_1^l h_j^l\big)$$

- In attentional GNNs, weights depend on the neighborhood's features
  - Different weights to different nodes in a neighbourhood
  - Remove dependence on the global graph structure



Attention weights

Comparison of input features

Input features of nodes i, j

Updated embedding

$$h_i^{l+1} = \sigma\big(\alpha_{ii}^l \theta_0^l h_i^l + \sum_{j \in \mathcal{N}_i} \alpha_{ij}^l \theta_1^l h_j^l\big)$$

$$\alpha_{ij}^{l+1} = \alpha\big(f_\theta(h_i^{l+1}), f_\theta(h_j^{l+1})\big)$$

Updated edge attention

[Velickovic et al., Graph Attention Networks, ICLR 2018]

Summary of spatial methods

- **Pros:**
  - Intuitive
  - Easy to implement
  - Generalize to inductive settings

- **Cons:**
  - Lack of interpretation in the spectral domain
  - Require many message passing iterations if the size of the graph is large

Useful resources

- **Toolboxes**
  - https://github.com/rusty1s/pytorch_geometric
  - https://github.com/dmlc/dgl
  - https://github.com/deepmind/jraph
  - https://github.com/tensorflow/gnn
- **Datasets**
  - DGL datasets: https://docs.dgl.ai/api/python/dgl.data.html
  - PyG datasets: https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html
  - OGB datasets: https://ogb.stanford.edu
  - https://chrsmrrs.github.io/datasets/
  - https://chrsmrrs.github.io/datasets/

Summary

- Graph neural networks: A very active area of research
- Different architecture designs, most of them can be categorized as convolutional, message passing, attentional
- The expressive power of GNN architectures will be discussed in the next lecture!

References

1. Graph representation learning (chap 5), William Hamilton
   - https://www.cs.mcgill.ca/~wlh/grl_book/files/GRL_Book.pdf
2. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges, Bronstein et al, 2022
   - https://arxiv.org/abs/2104.13478
3. A Comprehensive Survey on Graph Neural Networks, Wu et al, 2021
   - https://arxiv.org/abs/1901.00596
4. Graph neural networks: A review of methods and applications, Zhou et al, 2020.
   - https://arxiv.org/pdf/1812.08434.pdf