Graph Neural Networks

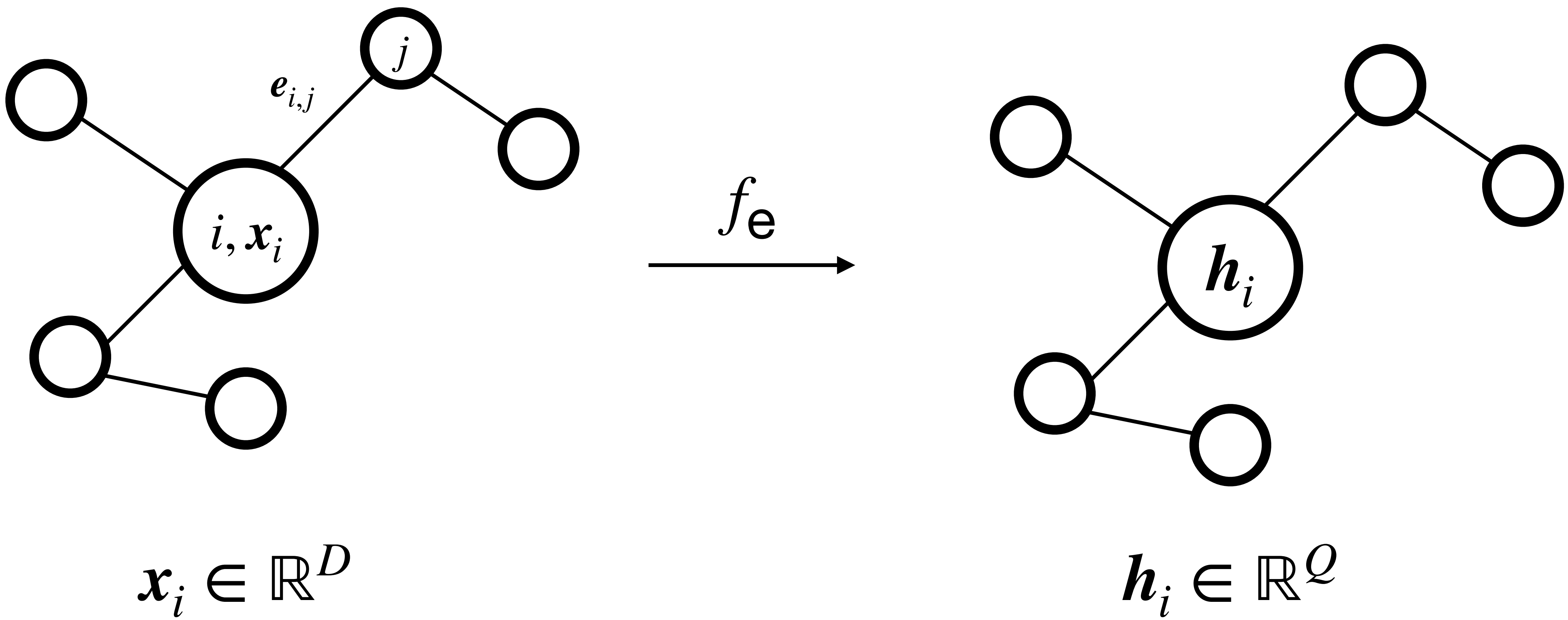
Rajbir Singh Nirwan

References

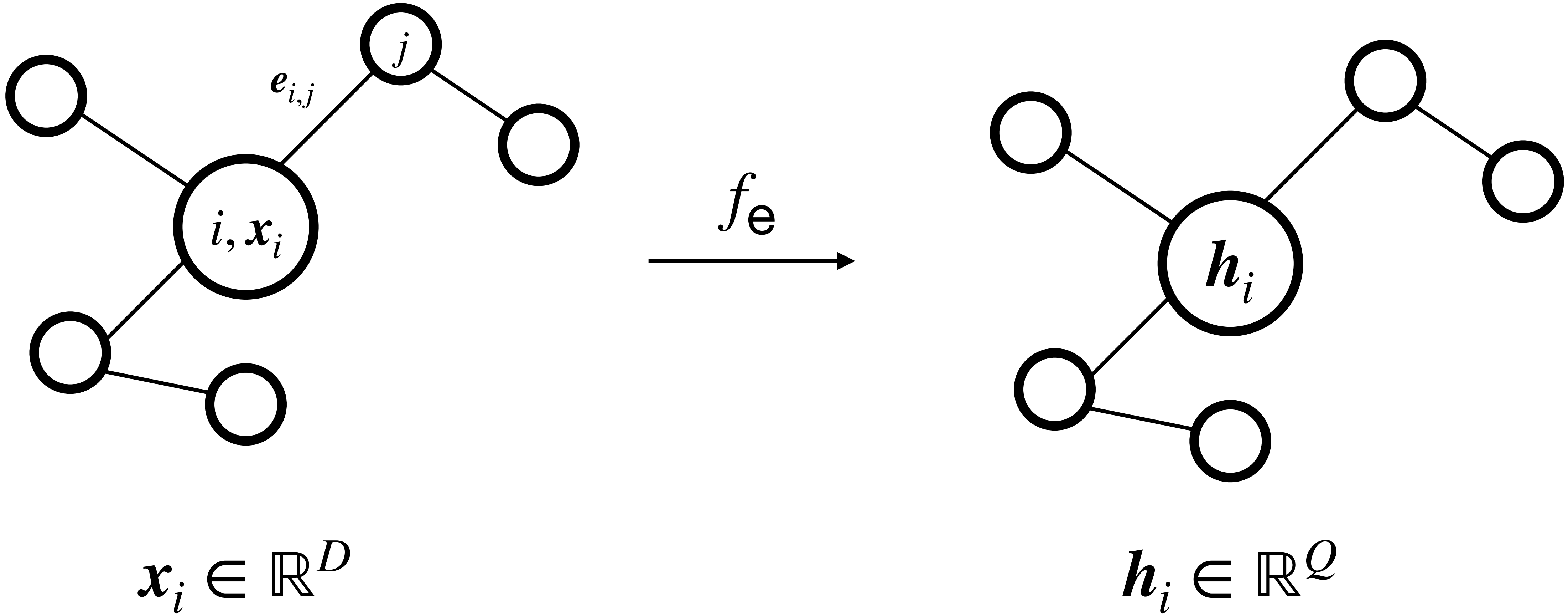
CS224W: Machine Learning with Graphs (Jure Leskovec)

Book: Graph Representation Learning (William L. Hamiltion)

Blog: Best NN Architectures (Sergios Karagiannakos)



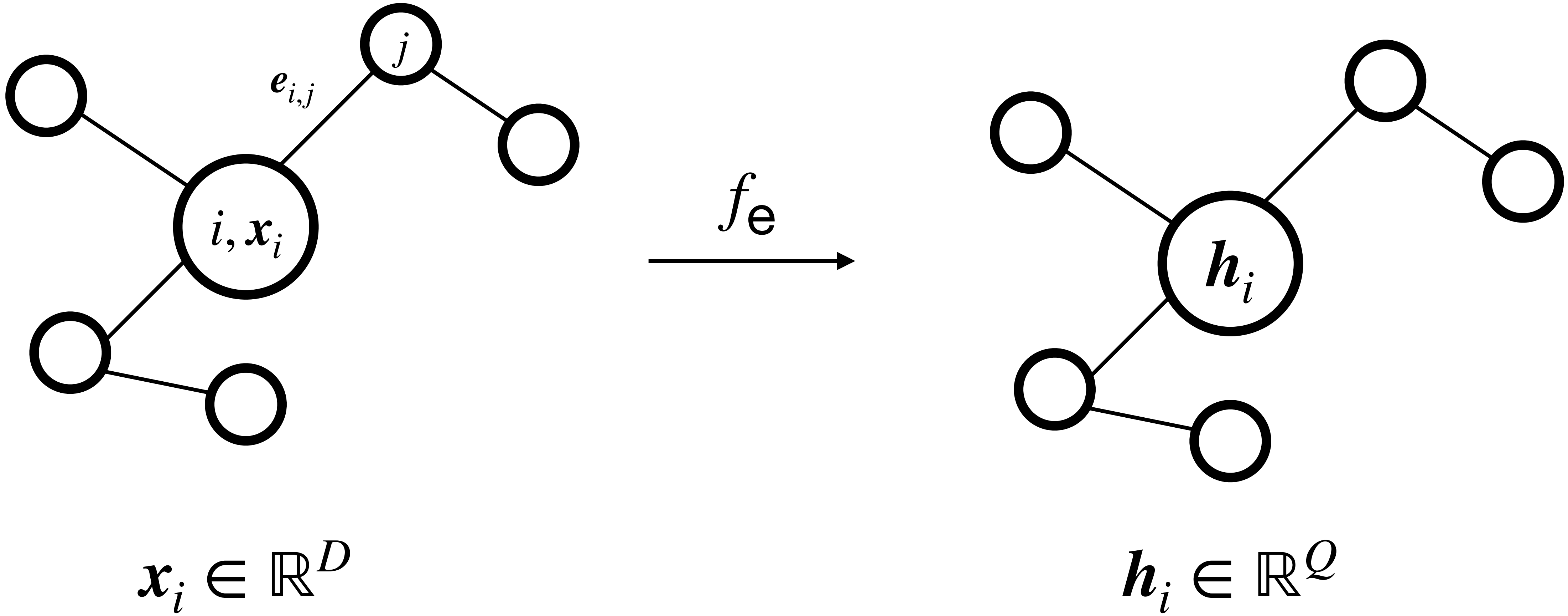
Node Classification	$z_i = f(\boldsymbol{h}_i)$
Edge Classification	$z_{i,j} = f(\boldsymbol{h}_i, \boldsymbol{h}_j, \boldsymbol{e}_{i,j})$
Graph Classification	$z_G = f(\mathrm{AGG}(\{\boldsymbol{h}_i \;\; i \in \nu\}))$



Supervised Embedding (e.g. Node labels $\boldsymbol{y} = (y_1, \ldots, y_N)^T$ available)

$$\min_{\boldsymbol{\theta}} L\left(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{h}_{\boldsymbol{\nu}})\right)$$

$$L = \sum_i \mathsf{CrossEntropy}(y_i, f_{\boldsymbol{\theta}}(\boldsymbol{h}_i))$$

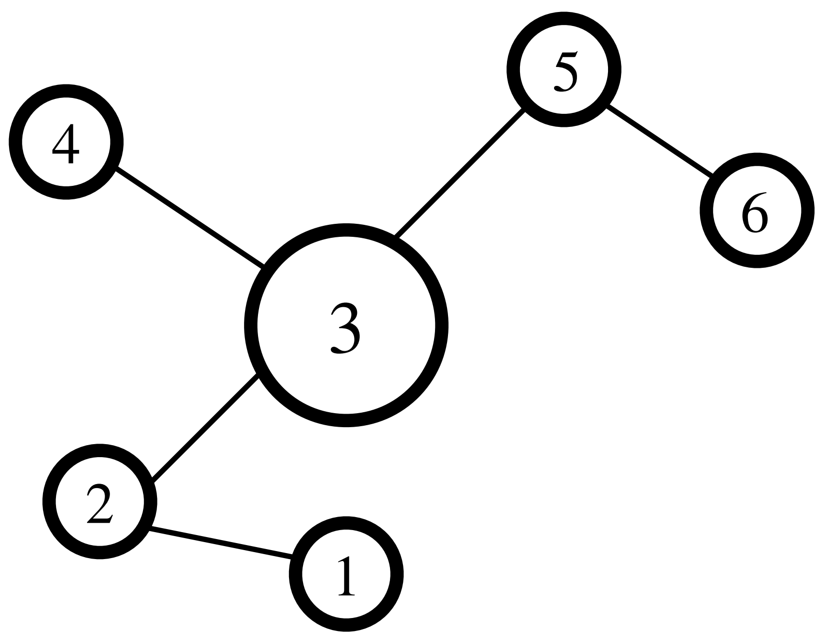


Unsupervised/self-supervised Embedding (use similarity $S : i, j \rightarrow s$)

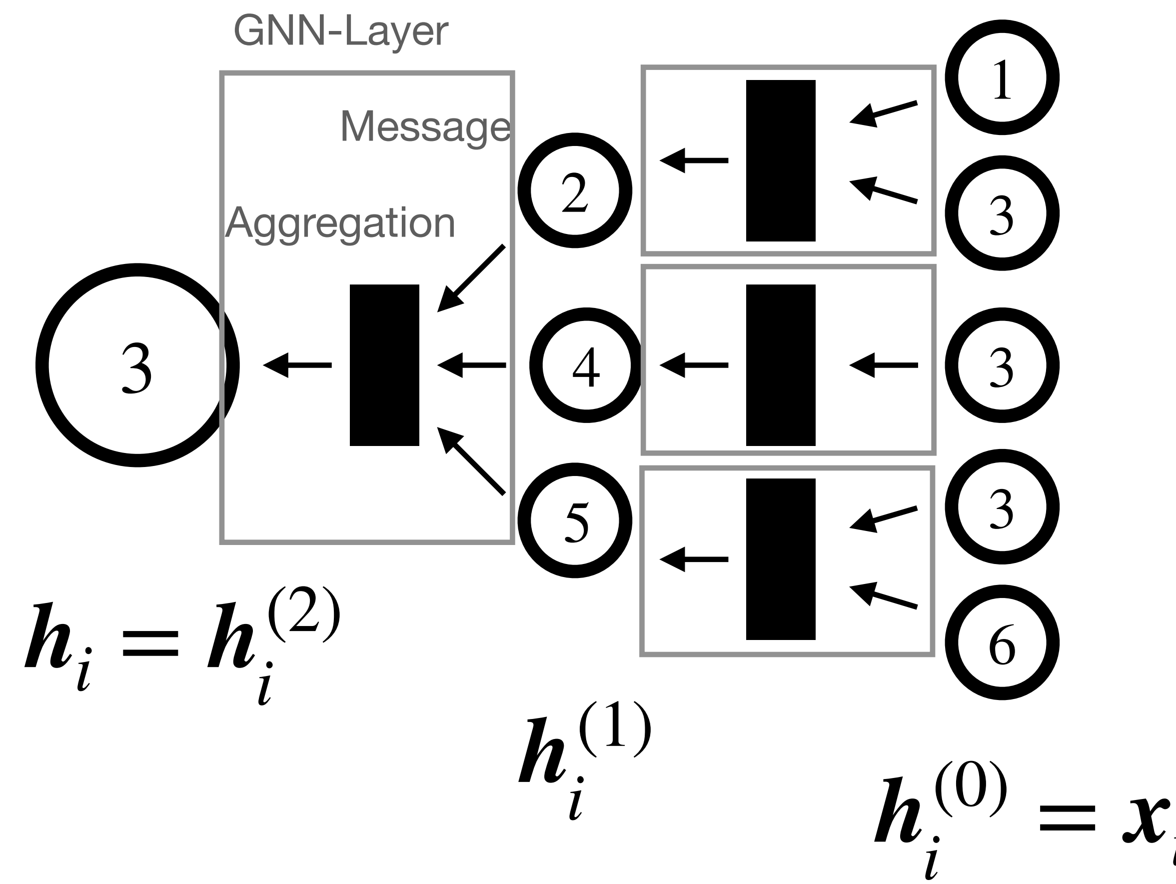
$$L = \sum_{i,j} \text{CrossEntropy}(S(i,j), \boldsymbol{h}_i^T \boldsymbol{h}_j)$$

S can be an indicator function for connected nodes or some type of random walk or …

Graph



Computation graph for node 3



$$\boldsymbol{h}_i^{(l+1)} = \mathrm{update}\left(f(\boldsymbol{h}_i^{(l)}), g(\mathrm{agg}(\{\mathrm{msg}(\boldsymbol{h}_j^{(l)})\ \ j \in \mathcal{N}(i)\}))\right)$$

$$\boldsymbol{h}_i^{(l+1)} = \underbrace{f_u}_{\text{non-Linearity}}\left(\boldsymbol{h}_i^{(l)}, \underbrace{\sum_{j\in\mathcal{N}(i)}}_{\text{Agg}} \underbrace{c_{i,j}}_{\text{Weights}}\ \underbrace{m(\boldsymbol{h}_i^{(l)}, \boldsymbol{h}_j^{(l)})}_{\text{Message}}\right)$$

GCN (Kipf, Welling ICLR 2017)

$$\boldsymbol{h}_i^{(l+1)} = \overset{\text{non-Linearity}}{\sigma}\left(\overbrace{\boxed{\sum_{j\in\mathcal{N}(i)}}}^{\text{Agg}}\ \overbrace{\boxed{\boldsymbol{W}^{(l+1)}\frac{\boldsymbol{h}_j^{(l)}}{\underbrace{N(j)}_{\text{Weights}}}}}^{\text{Message}}\right)$$

GraphSAGE (Hamilton et. al. NeurIPS 2017)

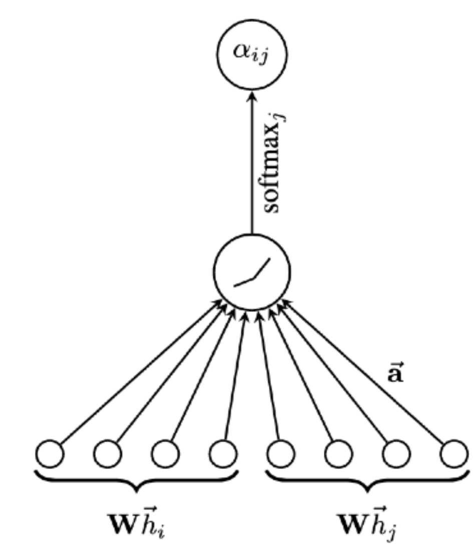
$$\boldsymbol{h}_i^{(l+1)} = \sigma\left(\boldsymbol{W}^{(l+1)}\text{Concat}\left(\boldsymbol{h}_i^{(l)}, \text{Agg}(\{\boldsymbol{h}_j^{(l)}\ \ j \in \mathcal{N}(i)\})\right)\right)$$

Agg: mean, LSTM and pooling aggregators
L2-Normalization at the end to the output

$$\boldsymbol{h}_i^{(l+1)} = \underbrace{f_u}_{\text{non-Linearity}} \left(\boldsymbol{h}_i^{(l)}, \underbrace{\sum_{j \in \mathcal{N}(i)}}_{\text{Agg}} \underbrace{c_{i,j}}_{\text{Weights}} \underbrace{m(\boldsymbol{h}_i^{(l)}, \boldsymbol{h}_j^{(l)})}_{\text{Message}} \right)$$

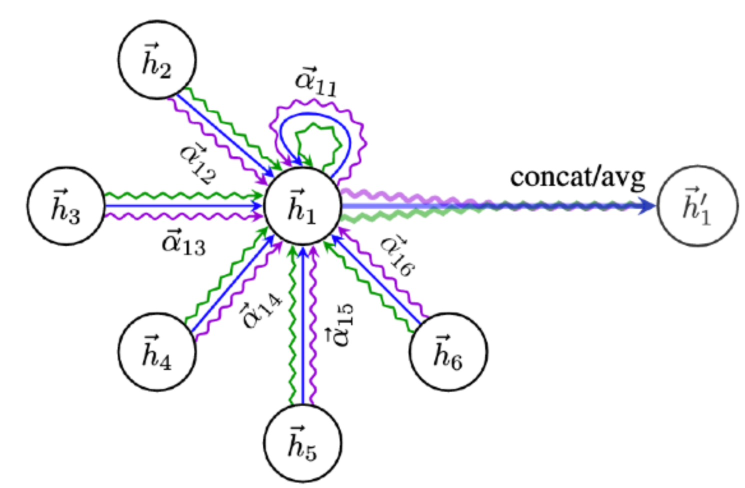
GAT (Velickovic et al, ICLR 2018)

$a_{i,j} = \text{softmax}(a(\boldsymbol{W}\boldsymbol{h}_i, \boldsymbol{W}\boldsymbol{h}_j))$



$$\boldsymbol{h}_i^{(l+1)} = \sigma \left(\sum_{j \in \mathcal{N}(i)} a_{i,j} \boldsymbol{W}^{(l+1)} \boldsymbol{h}_j^{(l)} \right)$$

$$\boldsymbol{h}_i^{(l+1)} = \sigma \left(\text{AGG}_k \left(\sum_{j \in \mathcal{N}(i)} a_{i,j}^k \boldsymbol{W}^{(l+1)} \boldsymbol{h}_j^{(l)} \right) \right)$$



$$\boldsymbol{h}_i^{(l+1)} = f_u\left(\boldsymbol{h}_i^{(l)}, \sum_{j\in\mathcal{N}(i)} c_{i,j}\; m(\boldsymbol{h}_i^{(l)}, \boldsymbol{h}_j^{(l)})\right)$$

non-Linearity Agg Weights Message

GraphSAGE (Hamilton et. al. NeurIPS 2017) (Neighborhood Sampling)

$$\boldsymbol{h}_i^{(l+1)} = \sigma\left(\boldsymbol{W}^{(l+1)}\text{Concat}\left(\boldsymbol{h}_i^{(l)}, \text{Agg}(\{\boldsymbol{h}_j^{(l)}\;\; j \sim p_{\mathcal{N}(j)}(j)\})\right)\right)$$

$$L(\boldsymbol{h}_i) = -\log\left(\sigma(\boldsymbol{h}_i^T\boldsymbol{h}_j)\right) - Q\mathbb{E}_{j\sim p(j)}\log\left(\sigma(-\boldsymbol{h}_i^T\boldsymbol{h}_j)\right)$$

PinSage (Ying et al 2018)

- Efficient neighborhood sampling using random walk
- Engineering excellence
- Scaling up to billions of data points

$$\boldsymbol{h}_i^{(l+1)} = f_u\left(\boldsymbol{h}_i^{(l)}, \sum_{j \in \mathcal{N}(i)} c_{i,j}\ m(\boldsymbol{h}_i^{(l)}, \boldsymbol{h}_j^{(l)})\right)$$

non-Linearity Agg Weights Message

GNNInf (Satorras et al.) - Multivariate Time Series Forecasting with Latent Graph Inference