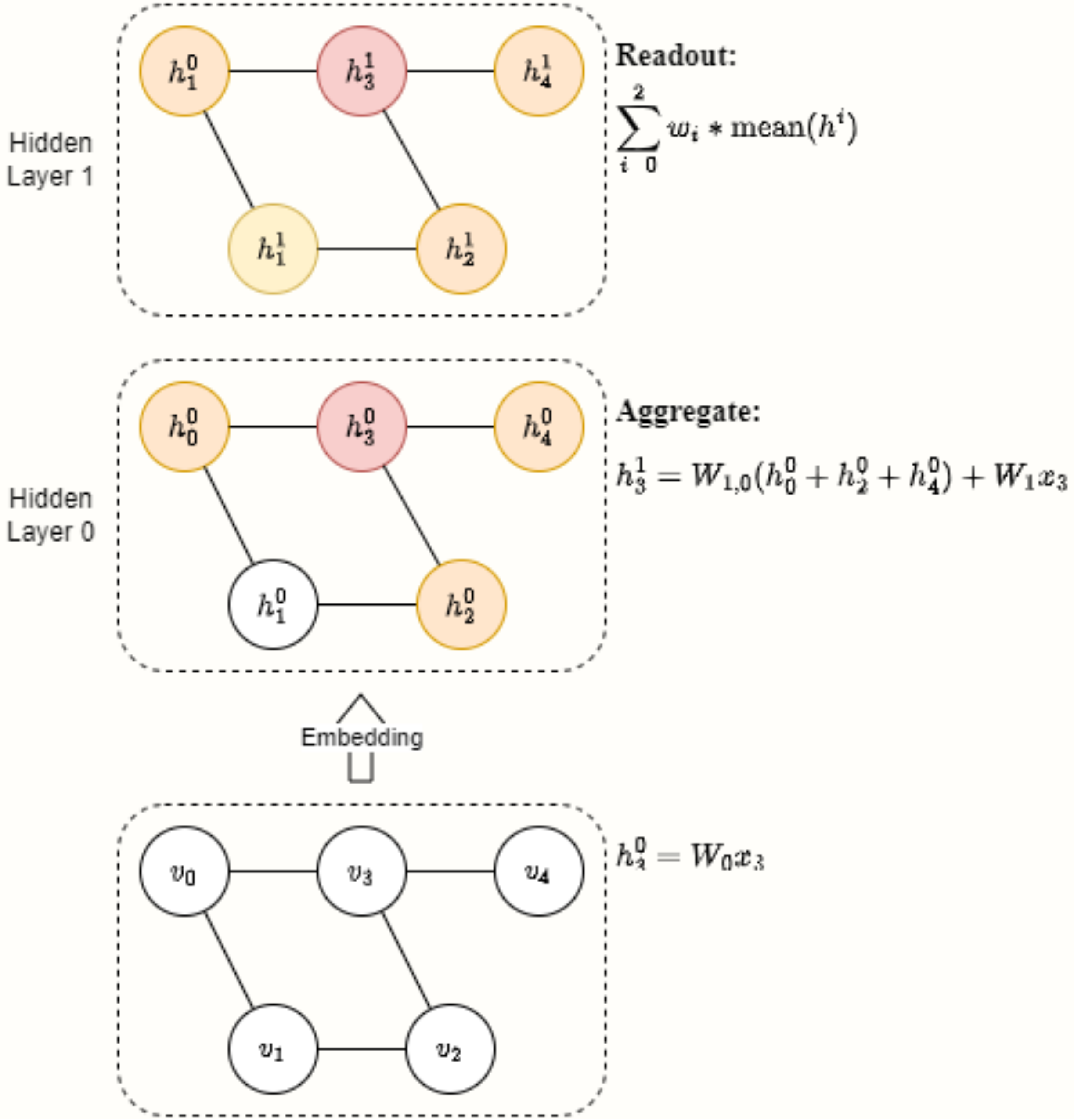
NN4G

Neural Network for Graphs

IEEE TNN 2009

- First **spatial-based** GNN
- With **residual**
- Aggregate by direct sum
- Readout by direct mean sum



DCNN

Diffusion-Convolution Neural Network

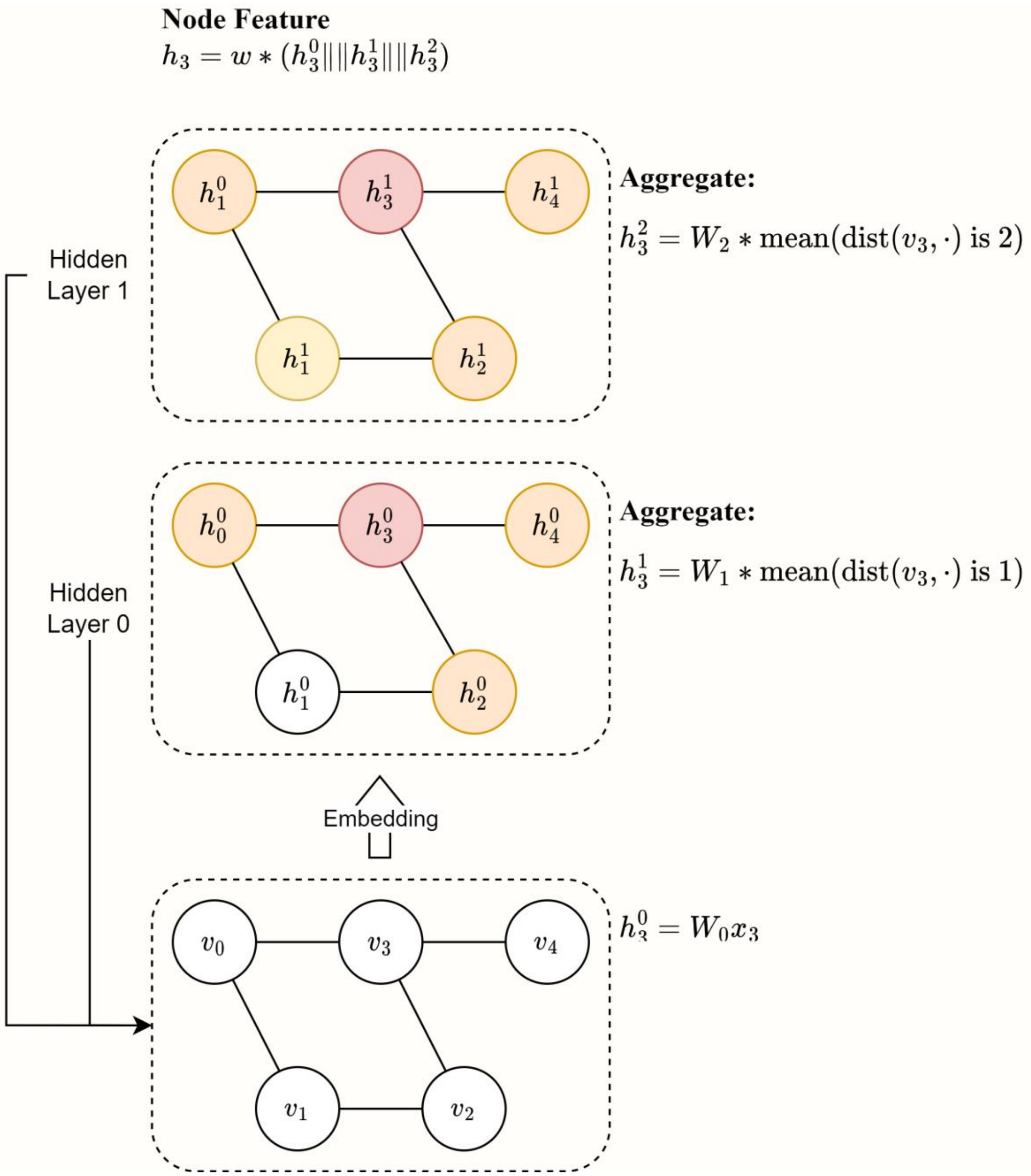
NIPS 2016

- Assume information spreads among nodes with a **fixed probability**(***P***), and it can reach **equilibrium** after several rounds of diffusion.
- ***A*****:** Adjacency Matrix
- ***D*****:** Degree Matrix

$$H^k = W^k P^k x$$

$$P = D^{-1}A$$

$$Y = \mathrm{cat}(H^i)\ W$$



Atwood, James, and Don Towsley. "Diffusion-convolutional neural networks." *Advances in neural information processing systems* 29 (2016).

GraphSAGE

Graph SAmple and aggreGatE

NIPS 2017

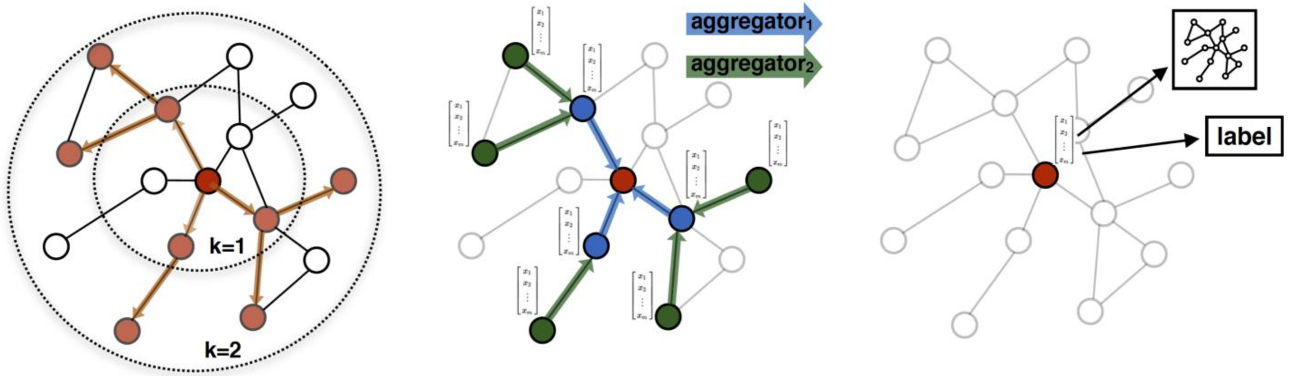
- Sample Neighbors
- Different aggregation function
  1. mean

     $$\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W} \cdot \text{MEAN}(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}).$$

  2. pool

     $$h_{N(v)}^k = max(\{\sigma(W_{pool} h_{ui}^k + b)\}, \forall u_i \in N(v))$$
     $$h_v^k = \sigma(W^k \cdot CONCAT(h_v^{k-1}, h_{N(u)}^{k-1}))$$

  3. LSTM



1. Sample neighborhood
2. Aggregate feature information from neighbors
3. Predict graph context and label using aggregated information

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \to 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

```
h_v^0 ← x_v, ∀v ∈ V ;
for k = 1...K do
    for v ∈ V do
        h^k_N(v) ← AGGREGATE_k({h_u^{k-1}, ∀u ∈ N(v)});
        h_v^k ← σ(W^k · CONCAT(h_v^{k-1}, h^k_N(v)))
    end
    h_v^k ← h_v^k / ||h_v^k||_2, ∀v ∈ V
end
z_v ← h_v^K, ∀v ∈ V
```

Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." *Advances in neural information processing systems* 30 (2017).

GAT

Graph Attention Networks

ICLR 2018

- Energy (Attention) of node ***j*** to ***i***

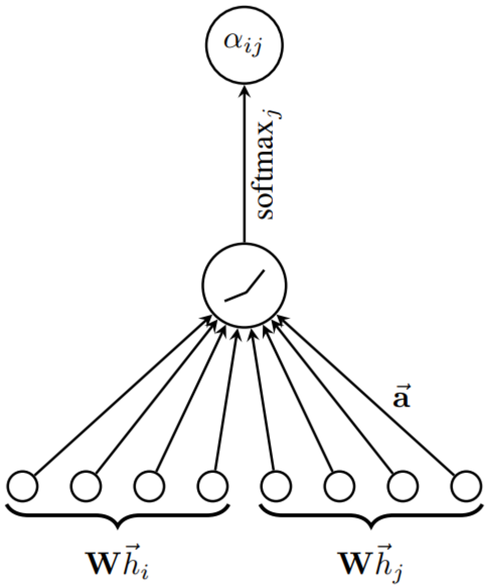
$$e_{ij} = a(\mathbf{W}h_i, \mathbf{W}h_j),$$

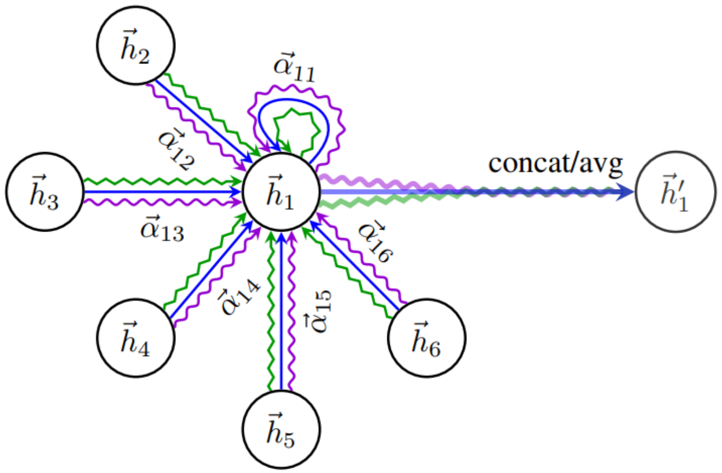
$$\text{where} \begin{cases} a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \to \mathbb{R} & \text{attentional mechanism is FF} \\ \mathbf{W} \in \mathbb{R}^{F' \times F} & \text{weight matrix} \end{cases}$$

- Coefficient (Weigh)

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}(\mathbf{a}^\top[\mathbf{W}h_i||\mathbf{W}h_j])\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}(\mathbf{a}^\top[\mathbf{W}h_i||\mathbf{W}h_j])\right)}$$

$$\text{where, } \mathbf{a} \in \mathbb{R}^{2F'} \text{ weight vector}$$





- Output

$$h'_i = \Big\Vert_{k=1}^{K} \sigma\Big(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k h_j\Big)$$

$$\text{where} \begin{cases} \alpha_{ij}^k : \text{normalized attention coefficients} \\ \mathbf{W}^k : \text{corresponding input linear transformation's weight matrix} \end{cases}$$

$$h'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k\mathbf{W}^k h_j\right)$$

Velickovic, Petar, et al. "Graph attention networks." *stat* 1050 (2017): 20.