Summary Questions of the lecture

- Explain the key aspects of GAT: Graph Attention Networks.

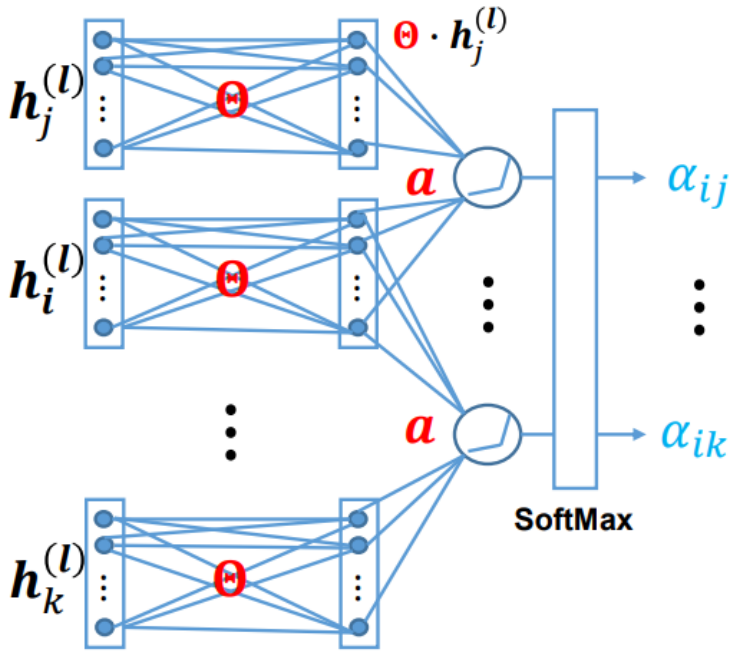
→ GAT aggregates the transformed feature vectors of neighbor nodes $\Theta h_j^{(l)}$ by weighting them with attention. The attention weights are generated by applying softmax across the compatibility scores of neighbor nodes: $LeakyReLU(a^T[\Theta h_i^{(l)} || \Theta h_j^{(l)}])$. Note that a and Θ are learnable parameters.

Aggregation:

$$\boldsymbol{h_i^{(l+1)} = \sigma\left(\sum_{v_j \in N(v_i)} \alpha_{ij}\, \Theta \cdot h_j^{(l)}\right)}$$

$$\alpha_{ij} = \frac{exp\left(LeakyReLU\left(a^T\left[\Theta \cdot h_i^{(l)} \,\|\, \Theta \cdot h_j^{(l)}\right]\right)\right)}{\sum_{v_k \in N(v_i)} exp\left(LeakyReLU\left(a^T\left[\Theta \cdot h_i^{(l)} \,\|\, \Theta \cdot h_k^{(l)}\right]\right)\right)}$$

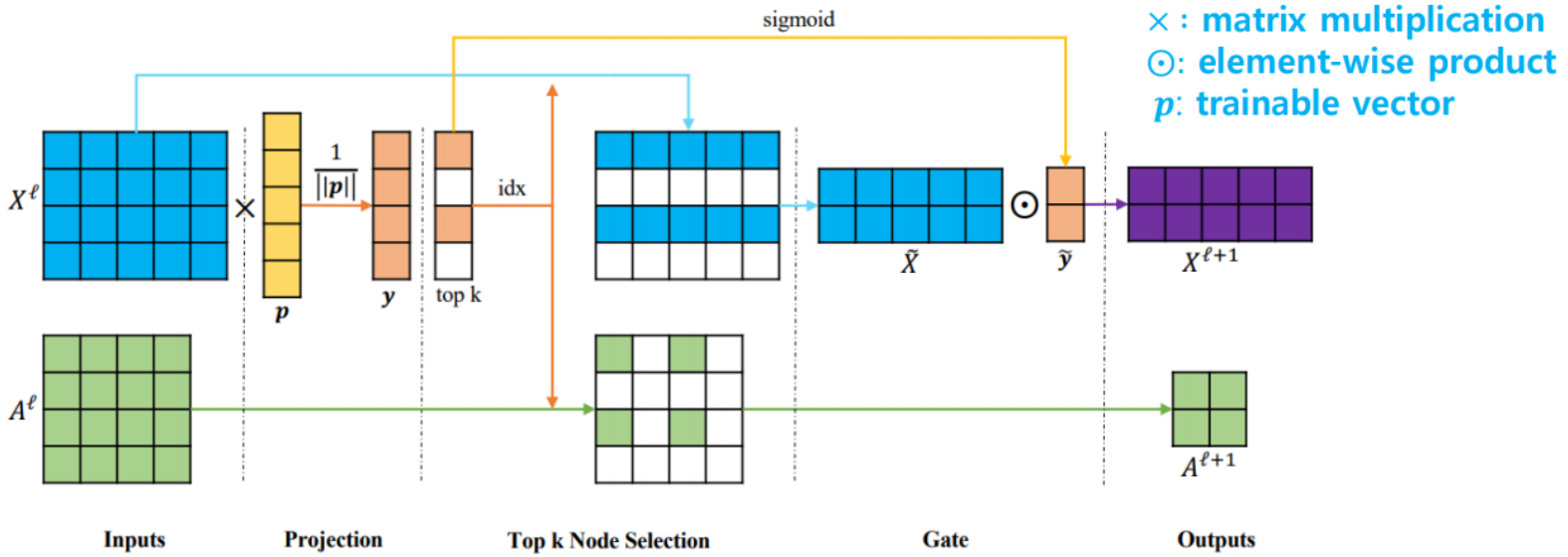
a, Θ: parameters of a single layer network



Summary Questions of the lecture

- Explain the key aspects of gPool: Graph U-Nets.

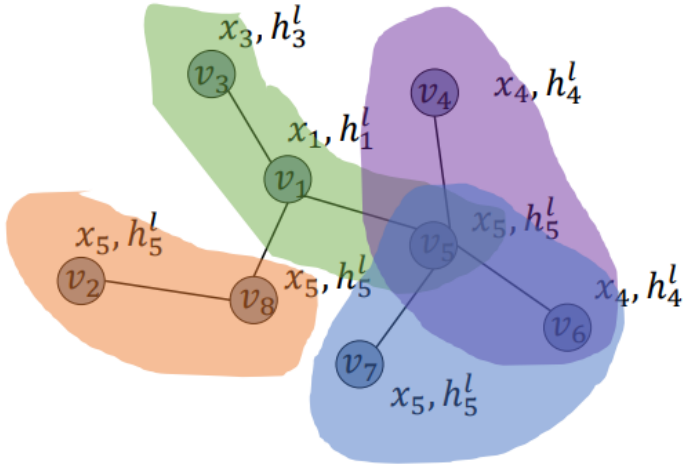
→ Graph U-Nets downsamples the graph using gPool, upsamples it using gUnpool, and transforms graph features with the simplified ChebNet. Specifically, gPool downsamples a given graph (X^l, A^l) by selecting the top k most important nodes. The importance scores y are generated by multiplying a learnable parameter vector (1×1 convolution filter) p to X^l and normalizing it. The scores are also used to gate the graph signals of the selected nodes to generate the output graph signals. gUnpool simply restores the graph back to its previous structure.



Summary Questions of the lecture

- Explain the key aspects of DiffPool: Hierarchical Graph Representation Learning with Differentiable Pooling

→ DiffPool first embeds the input graph X to H through the simplified ChebNet operation. Then, it generates a soft assignment matrix S through another simplified ChebNet operation, but this time its output is normalized with the softmax. Since S provides the probability matrix for each node being assigned to each cluster, the input graph can be pooled with $H_p = S^T H$ and $A_p = S^T AS$.



Assignment Matrix for pooling: $\boldsymbol{S} \in \mathbb{R}^{n \times n_p}$

$$\boldsymbol{S} = \boldsymbol{SoftMax}\left(\tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-1/2}\boldsymbol{X}\boldsymbol{\Theta}_{\boldsymbol{s}}\right) \leftarrow \boldsymbol{GCN}$$

GCN Filtering(node embedding): $\boldsymbol{H} \in \mathbb{R}^{n \times d_p}$

$$\boldsymbol{H} = \boldsymbol{ReLU}\left(\tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-1/2}\boldsymbol{X}\boldsymbol{\Theta}_{\boldsymbol{h}}\right) \leftarrow \boldsymbol{GCN}$$

DiffPool layer:

$$\boldsymbol{H_p} = \boldsymbol{S^T H} \in \mathbb{R}^{n_p \times d_p}$$

$$\boldsymbol{A_p} = \boldsymbol{S^T AS} \in \{0,1\}^{n_p \times n_p}$$

Summary Questions of the lecture

- Explain the key aspects of EigenPooling: Graph Convolutional Networks with EigenPooling
- EigenPooling first clusters the input graph by directly adopting Laplacian clustering. Then, within each cluster's sub-graph, the node features are smoothed by applying a hard low pass filter in the spectral domain. The resulting sub-graph feature vectors are averaged to obtain the feature vector of a representative node for each sub-graph.