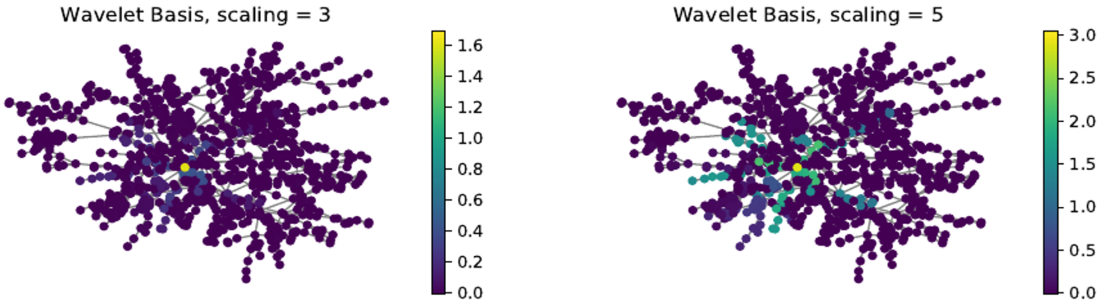
Deep Learning on Graphs
Part 2

Michael Kenning, Stavros Georgousis

Spectral Convolution cont.

- Same objective as spatial convolution
  - (Theoretically) no problem with neighbourhood selection.
  - Signal diffused from central node.
- Cons -> Computationally expensive



Source: Xu et al. 2019

Spectral Convolution cont.

Idea of spectral convolution:

1. Project graph signal to spectral domain

   $$\tilde{f} = \mathbf{\Phi}^\top f$$

2. Filter in spectral domain.
3. Get a new feature representation in vertex (spatial) domain.

   $$f = \mathbf{\Phi}\tilde{f}$$

Example: Spectral Convolution (Bruna et. al., 2014)

- Fourier transform -> U = eigenvectors of L: $\hat{\mathbf{L}} = \mathbf{U}\hat{\mathbf{\Lambda}}\mathbf{U}^\top$
- Convolution defined as:

$$f \star \mathbf{g} = \mathbf{U}\left(\left(\mathbf{U}^\top f\right) \odot \left(\mathbf{U}^\top \mathbf{g}\right)\right)$$

- Layer implementation: $f_{l+1,j} = h\left(\mathbf{U}\sum_{i=0}^{c-1}\left(\mathbf{\Theta}_{l,i,j}\mathbf{U}^\top f_{l,i}\right)\right)$

- Computationally expensive: O(n^2)

Example: ChebNet (Defferard et al., 2016)

- Calculation of eigenvalue decomposition computationally expensive
- Solution -> approximate the Fourier transformed signal (Chebyshev polynomials)

Approximate !

$$\mathbf{g}_\theta(\mathbf{\Lambda}) \approx \sum_{i=0}^{k-1} \theta_i T_i(\bar{\mathbf{\Lambda}}),$$

$$\mathbf{g}_\theta(\mathbf{L}) \approx \sum_{i=0}^{k-1} \theta_i T_i(\bar{\mathbf{L}}) f$$

$$f_{l+1,j} = h\left(\sum_{i=0}^{c-1} \mathbf{g}_\theta(\bar{\mathbf{L}}) f_{l,i}\right)$$

Example: Graph Convolution Network (GCN) (Kipf et al., 2017)

Where we left off:

$$\mathbf{g}_\theta(\mathbf{L}) \approx \sum_{i=0}^{k-1} \theta_i T_i(\bar{\mathbf{L}}) f$$

Approximate !

$$f \star \mathbf{g} \approx \theta_0' f + \theta_1'(\mathbf{L} - \mathbf{I}_n) f = \theta_0' f - \theta_1' \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2} f.$$

$$f \star \mathbf{g} \approx \theta \left(\mathbf{I}_n + \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2} \right) f = \theta \mathbf{L} f$$

Renormalisation trick

$$\mathbf{I}_n + \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2} \rightarrow \bar{\mathbf{D}}^{-1/2} \bar{\mathbf{A}} \bar{\mathbf{D}}^{-1/2}$$

$$\bar{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n,$$

$$f_{l+1} = h\left(\bar{\mathbf{D}}^{-1/2} \bar{\mathbf{A}} \bar{\mathbf{D}}^{-1/2} f_l \boldsymbol{\Theta}_l \right)$$

- Computationally efficient (well… relatively)
- Neighbourhood order: 1
- Basically the baseline standard
- Curious case (could be said it isn't spectral)

Graph Pooling

Global pooling (readout):

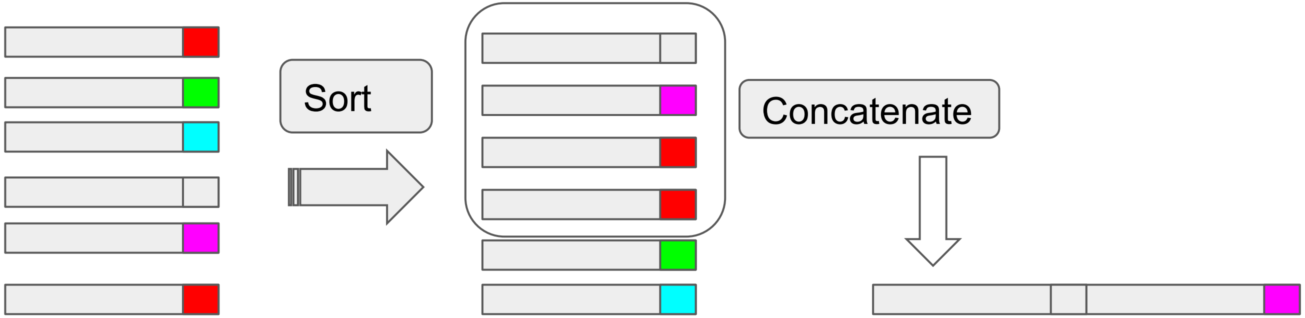
- Summarises features for graph level tasks
- No equivalent in CNNs

Hierarchical pooling:

- Analogous to standard pooling in CNNs
- Dilates receptive field
- Can improve performance using hierarchical representation

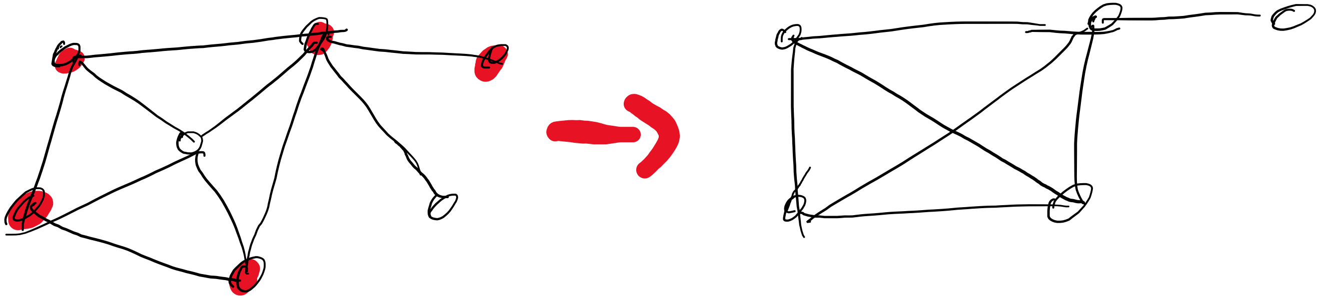
Global pooling (readout layer)

- Min, max, mean, sum pooling
- Combinations of above mean + max pool (Ying et al., 2018)
- Interesting example: SortPool (Zhang et al., 2019)



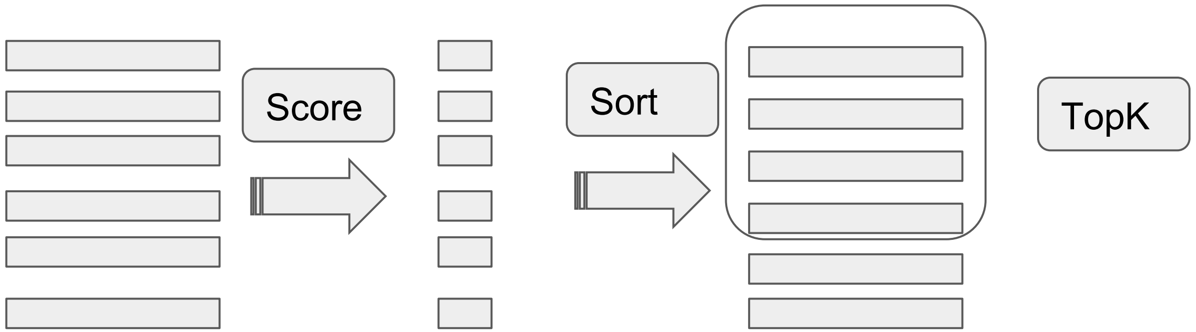
Hierarchical Pooling

- Subsampling based
  - Pros -> Efficient
  - Cons -> Discard information, potentially isolated vertices
- Clustering based
  - Common caveat: pooled graph dense



Subsampling based pooling

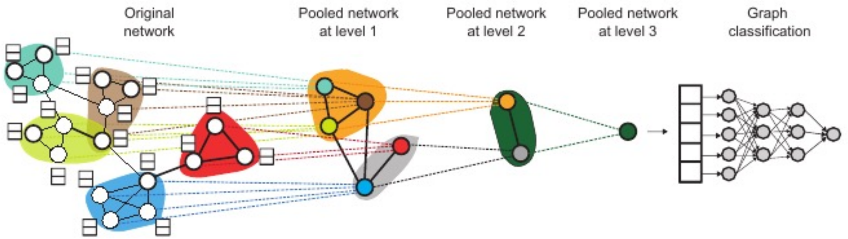
Examples: Top-K pooling (Graph U-Nets, SAGPool)



Clustering based pooling

Common approach learning a clustering assignment matrix, S

$$\mathbf{A}_{\text{pool}} = \mathbf{S}^{\top}\mathbf{A}\mathbf{S}$$
$$\mathbf{A}_{\text{unpool}} = \mathbf{S}\mathbf{A}_{\text{pool}}\mathbf{S}^{\top}$$



Source: Ying et al. 2019

S → truncated matrix of weights from original nodes to supernodes

Example: Edge Pooling

Selects edges to be contracted based on edge scores.

- Pros: Retains sparsity