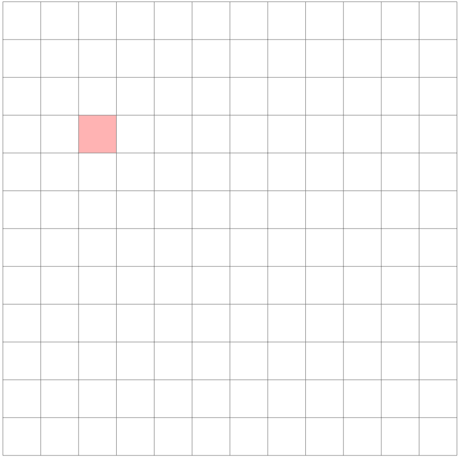
Pooling in Graph Neural Networks

Graph Deep Learning 2021 - Lecture 4

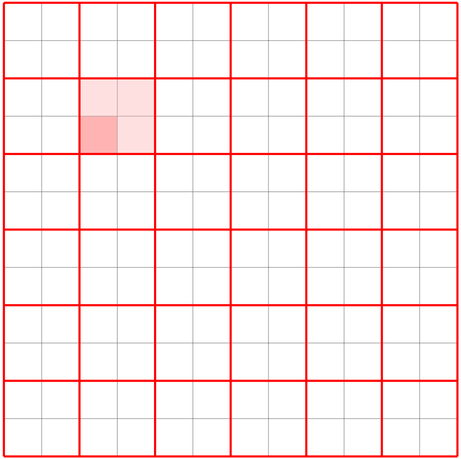
Daniele Grattarola

March 15, 2021

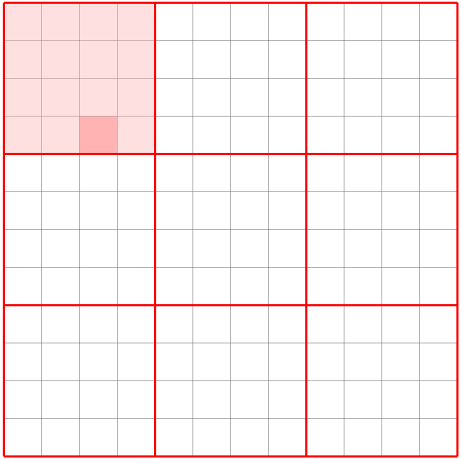
Pooling in CNNs



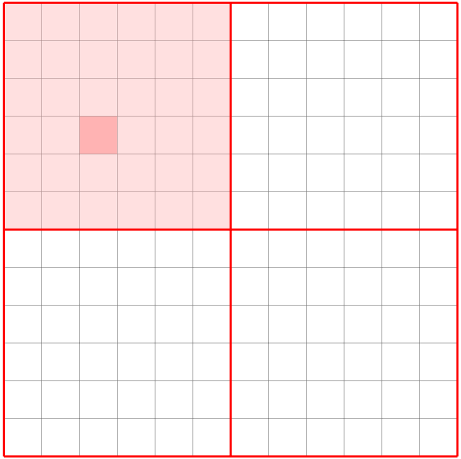
Pooling in CNNs



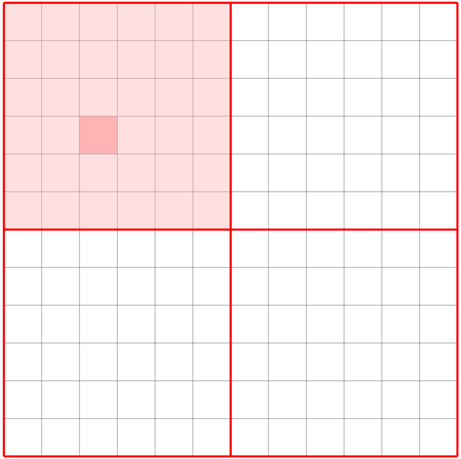
Pooling in CNNs

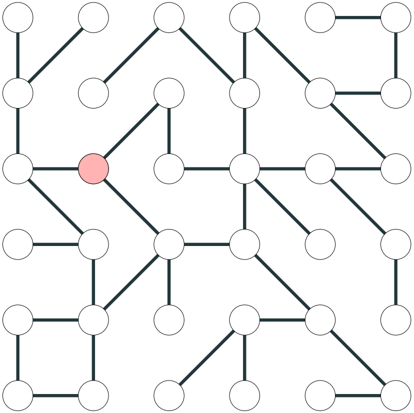


Pooling in CNNs

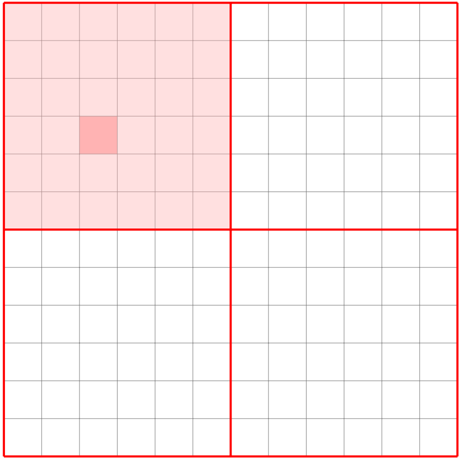


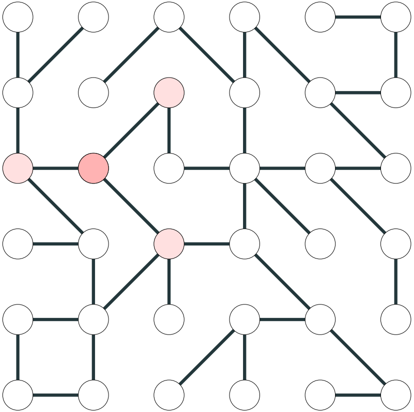
Pooling in CNNs



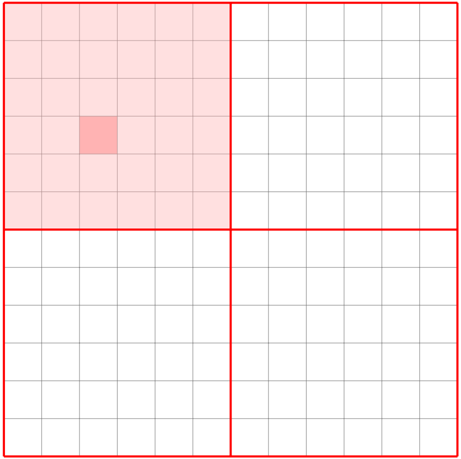


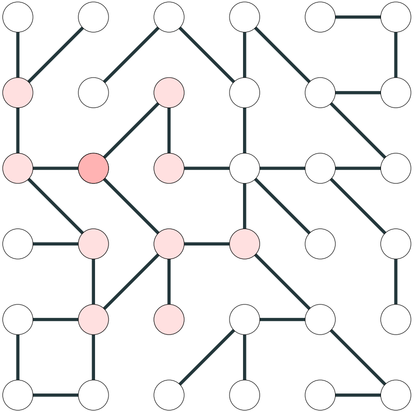
Pooling in CNNs



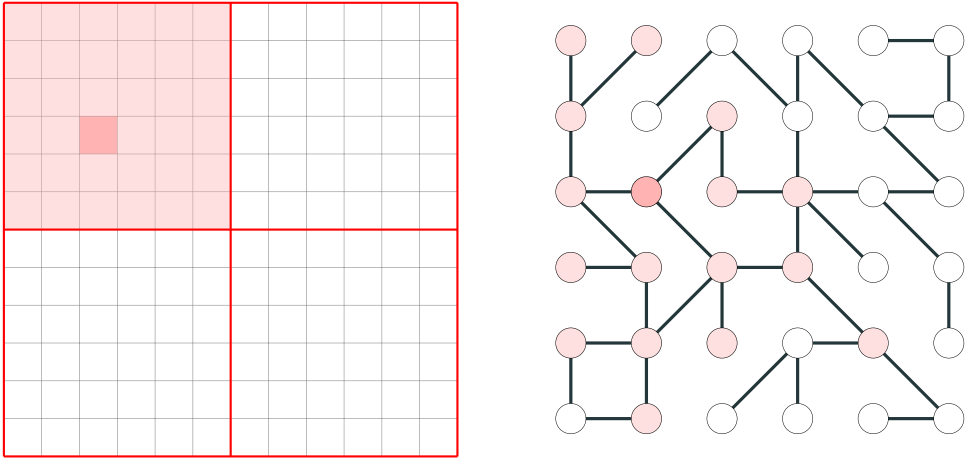


Pooling in CNNs





Pooling in CNNs



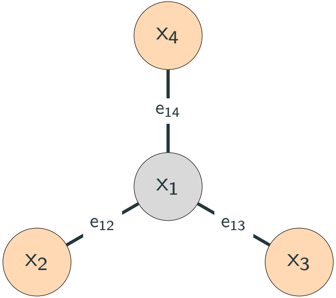
Roadmap

Things we are going to cover:

- A “message passing” for pooling
- Methods
- Global pooling
- Open questions

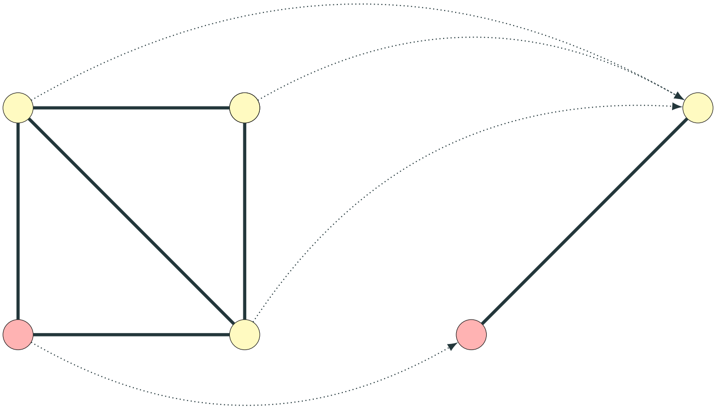
Notation recap

- Graph: nodes connected by edges;
- A, $N \times N$ adjacency matrix;
- $\mathrm{D} = \mathrm{diag}([d_1, \dots, d_N])$, diagonal degree matrix;
- $\mathrm{X} = [\mathrm{x}_1, \dots, \mathrm{x}_N]$, $\mathrm{x}_i \in \mathbb{R}^F$, node attributes or "graph signal";
- $\mathrm{e}_{ij} \in \mathbb{R}^S$, edge attribute for edge $i \to j$;



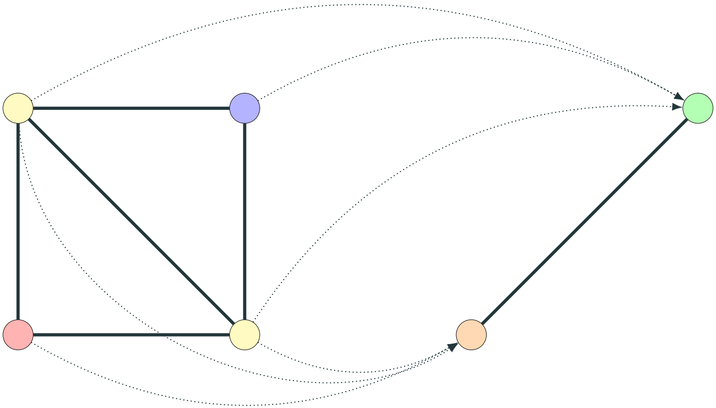
Graph coarsening by example

Strategy 1: aggregate same attributes (Candy Crush pooling).



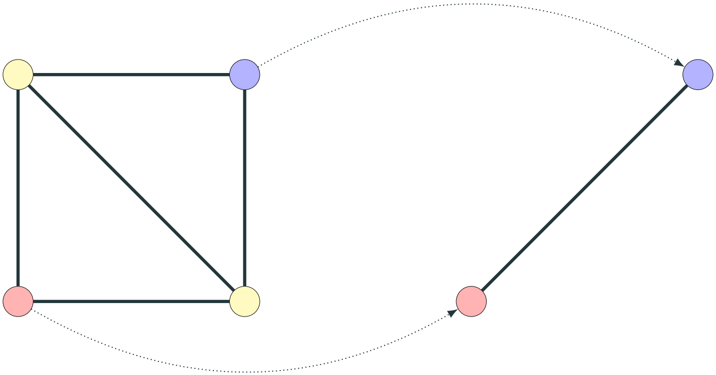
Graph coarsening by example

Strategy 2: aggregate cliques.



Graph coarsening by example

Strategy 3: keep only some types/colors.



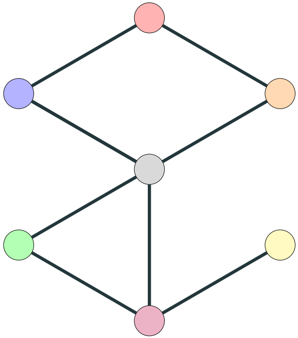
Three main questions [1]

1. How to identify **groups of related nodes**?
2. How to get **new node attributes** from the groups?
3. How to **connect** the new nodes?

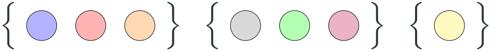
[1] D. Grattarola *et al.*, "Understanding Pooling in Graph Neural Networks," 2021 (In preparation).

Step 1: Select

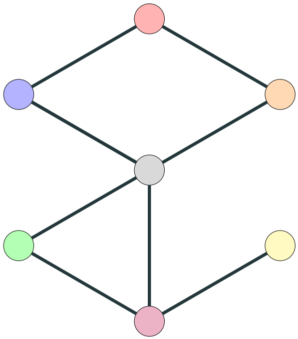
Selecting nodes



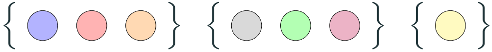
Example 1: partition.



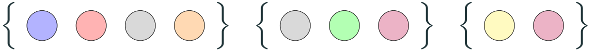
Selecting nodes



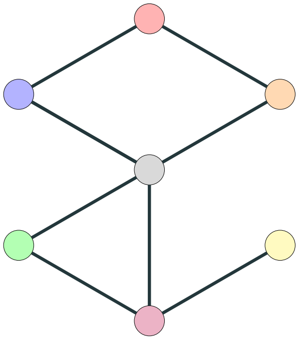
Example 1: partition.



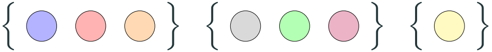
Example 2: cover (possible overlaps).



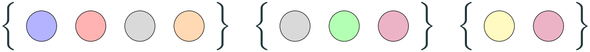
Selecting nodes



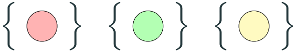
Example 1: partition.



Example 2: cover (possible overlaps).



Example 3: sparse.



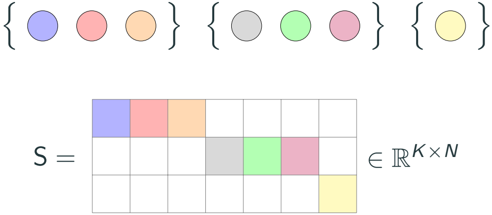
Selecting nodes

The **selection** stage computes K **supernodes**:

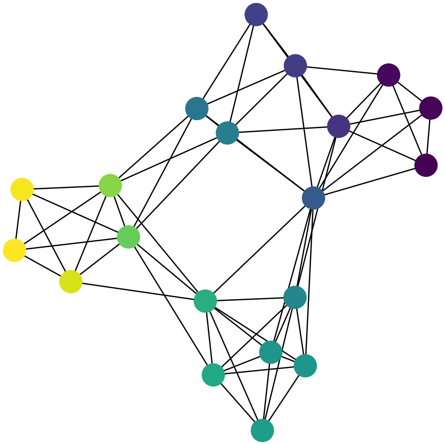
$$\mathrm{Sel} : \mathcal{G} \mapsto \mathcal{S} = \{\mathcal{S}_1, \dots, \mathcal{S}_K\}.$$

Each supernode is a set of nodes associated with a score:

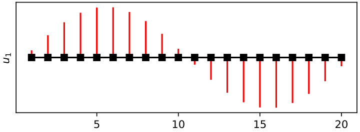
$$\mathcal{S}_k = \{(\mathrm{x}_i, \mathrm{s}_i) \mid \mathrm{s}_i \in \mathbb{R}_{>0}\},$$



Spectral clustering [3]



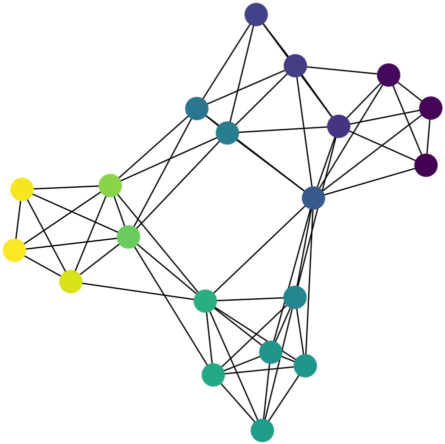
The low-frequency eigenvectors naturally cluster the nodes.



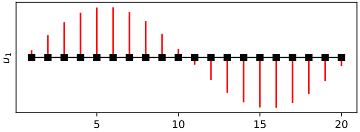
[2] J. Shi *et al.*, "Normalized cuts and image segmentation," 2000.

[3] U. Von Luxburg, "A tutorial on spectral clustering," 2007.

Spectral clustering [3]



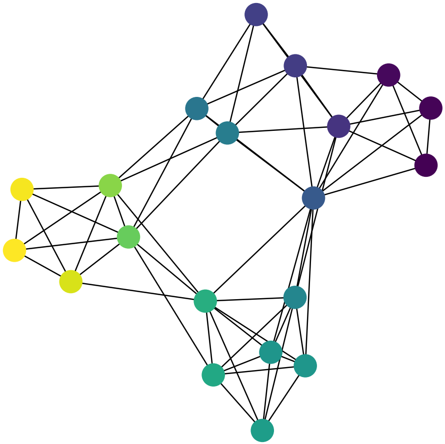
The low-frequency eigenvectors naturally cluster the nodes.



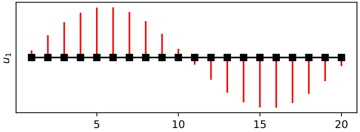
Idea: run k-means clustering (or similar) using the first few eigenvectors.

[2] J. Shi *et al.*, "Normalized cuts and image segmentation," 2000.
[3] U. Von Luxburg, "A tutorial on spectral clustering," 2007.

Spectral clustering [3]



The low-frequency eigenvectors naturally cluster the nodes.



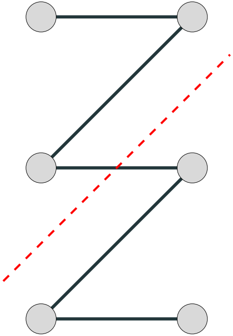
Idea: run k-means clustering (or similar) using the first few eigenvectors.

Equivalent to finding the **minimum normalized k-cut** [2].

---

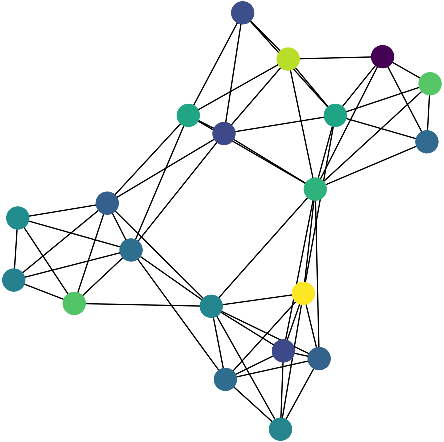
[2] J. Shi *et al.*, "Normalized cuts and image segmentation," 2000.
[3] U. Von Luxburg, "A tutorial on spectral clustering," 2007.

Mincut vs. Maxcut

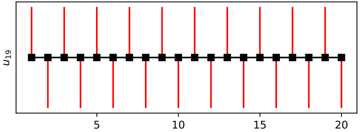


Mincut

Node decimation [5]



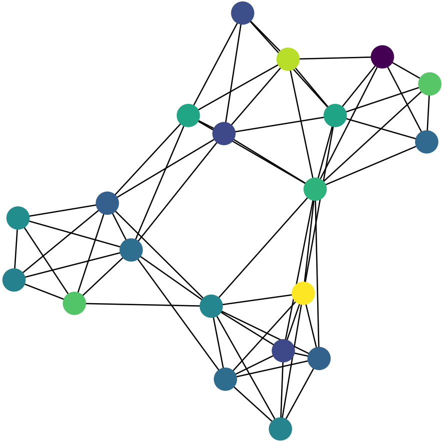
Alternative: use the highest-frequency eigenvector to do something similar to a regular subsampling.



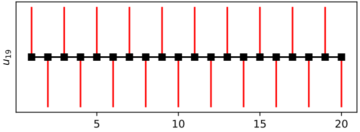
[4] L. Palagi *et al.*, "Computational approaches to max-cut," 2012.

[5] F. M. Bianchi *et al.*, *Hierarchical Representation Learning in Graph Neural Networks with Node Decimation Pooling*, 2019.

Node decimation [5]



Alternative: use the highest-frequency eigenvector to do something similar to a regular subsampling.

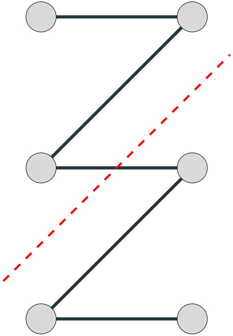


Equivalent to finding the **maximum 2-cut** [4].

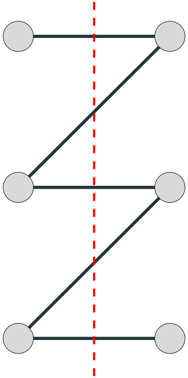
[4] L. Palagi *et al.*, "Computational approaches to max-cut," 2012.

[5] F. M. Bianchi *et al.*, *Hierarchical Representation Learning in Graph Neural Networks with Node Decimation Pooling*, 2019.

Mincut vs. Maxcut

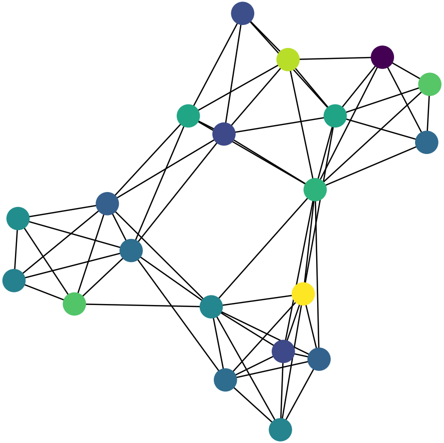


Mincut



Maxcut

Some problems



Problems with spectral methods:

- Computing eigenvectors is **expensive** ($O(N^3)$);
- They do not consider **attributes**.

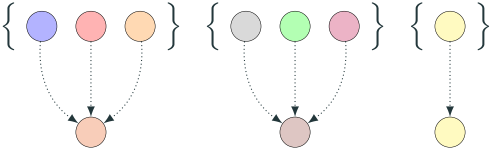
But we get the general idea...

Step 2: Reduce

Reducing supernodes

The **reduction** stage aggregates the supernodes in a **permutation-invariant** way:

$$\text{Red} : \mathcal{G}, \mathcal{S}_k \mapsto \mathrm{x}'_k$$



Typical approach is to take a **weighted sum** (weights given by the scores in the supernodes):

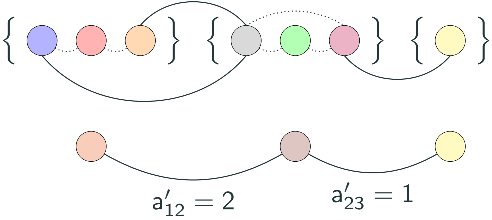
$$\mathrm{X}' = \mathrm{S}\mathrm{X} \quad (\in \mathbb{R}^{K \times F})$$

Step 3: Connect

Connecting supernodes

The **connection** function decides whether two supernodes are connected (and, in case, computes the associated attributes):

$$\text{Con} : \mathcal{G}, \mathcal{S}_k, \mathcal{S}_l \mapsto e'_{kl}$$



Typical approach is again to take a **weighted sum** of edges between two supernodes:

$$A' = SAS^\top \quad (\in \mathbb{R}^{K \times K})$$

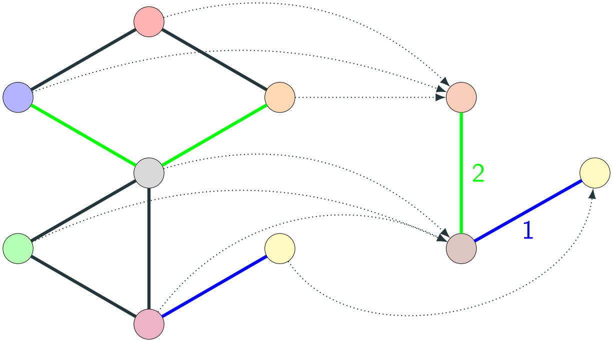
Select, Reduce, Connect [1]

Putting everything together:

$$\underbrace{\mathcal{S} = \{\mathcal{S}_k\}_{k=1:K} = \mathrm{Sel}(\mathcal{G});}_{\text{Selection}}$$

$$\underbrace{\mathcal{X}' = \{\mathrm{Red}(\mathcal{G}, \mathcal{S}_k)\}_{k=1:K};}_{\text{Reduction}}$$

$$\underbrace{\mathcal{E}' = \{\mathrm{Con}(\mathcal{G}, \mathcal{S}_k, \mathcal{S}_l)\}_{k,l=1:K};}_{\text{Connection}}$$



[1] D. Grattarola *et al.*, "Understanding Pooling in Graph Neural Networks," 2021 (In preparation).

Methods

Pooling methods

A few ideas:

1. **Graclus** [6]: visit nodes randomly, merge pairs that maximize $\frac{a_{ij}}{w_i} + \frac{a_{ij}}{w_j}$; [1]

   In [7], they reduce supernodes with element-wise max.

[6] I. S. Dhillon *et al.*, "Weighted graph cuts without eigenvectors a multilevel approach," 2007.
[7] M. Defferrard *et al.*, "Convolutional neural networks on graphs with fast localized spectral filtering," 2016.
[8] E. Luzhnica *et al.*, "Clique pooling for graph classification," 2019.
[9] E. Noutahi *et al.*, "Towards Interpretable Sparse Graph Representation Learning with Laplacian Pooling," 2019.

[1] w_i is a "weight" assigned to the node, *e.g.*, its degree.

Pooling methods

A few ideas:

1. **Graclus** [6]: visit nodes randomly, merge pairs that maximize $\frac{a_{ij}}{w_i} + \frac{a_{ij}}{w_j}$; [1]

   In [7], they reduce supernodes with element-wise max.
2. **Clique Pooling** [8]: merge together cliques.

[6] I. S. Dhillon *et al.*, "Weighted graph cuts without eigenvectors a multilevel approach," 2007.
[7] M. Defferrard *et al.*, "Convolutional neural networks on graphs with fast localized spectral filtering," 2016.
[8] E. Luzhnica *et al.*, "Clique pooling for graph classification," 2019.
[9] E. Noutahi *et al.*, "Towards Interpretable Sparse Graph Representation Learning with Laplacian Pooling," 2019.

[1] w_i is a "weight" assigned to the node, *e.g.*, its degree.

Pooling methods

A few ideas:

1. **Graclus** [6]: visit nodes randomly, merge pairs that maximize $\frac{a_{ij}}{w_i} + \frac{a_{ij}}{w_j}$; [1]

   In [7], they reduce supernodes with element-wise max.
2. **Clique Pooling** [8]: merge together cliques.
3. **LaPool** [9]: select "leaders" that have higher local variation $\|LX\|$ w.r.t. all their neighbors. Create clusters by assigning nodes to nearest leader.

---

[6] I. S. Dhillon *et al.*, "Weighted graph cuts without eigenvectors a multilevel approach," 2007.

[7] M. Defferrard *et al.*, "Convolutional neural networks on graphs with fast localized spectral filtering," 2016.

[8] E. Luzhnica *et al.*, "Clique pooling for graph classification," 2019.

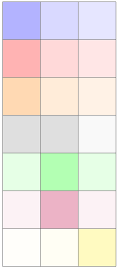
[9] E. Noutahi *et al.*, "Towards Interpretable Sparse Graph Representation Learning with Laplacian Pooling," 2019.

[1] w_i is a "weight" assigned to the node, *e.g.*, its degree.

Learning to pool

Key idea: learn to output $S^\top$ by giving node features X as input to a neural network.

- **DiffPool** [10]: GNN for $S^\top$, regularize with "link prediction" loss;

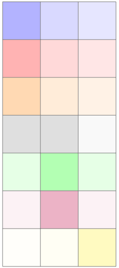
$\phi(X) = S^\top =$  $\in \mathbb{R}^{N \times K}$

[10] R. Ying *et al.*, "Hierarchical Graph Representation Learning with Differentiable Pooling," 2018.
[11] F. M. Bianchi *et al.*, "Mincut pooling in Graph Neural Networks," 2019.
[12] C. Bodnar *et al.*, "Deep Graph Mapper: Seeing Graphs through the Neural Lens," 2020.

Learning to pool

Key idea: learn to output $\mathsf{S}^\top$ by giving node features X as input to a neural network.

- **DiffPool** [10]: GNN for $\mathsf{S}^\top$, regularize with "link prediction" loss;
- **MinCutPool** [11]: MLP for $\mathsf{S}^\top$, regularize with "minimum cut" loss (same objective as spectral clustering);

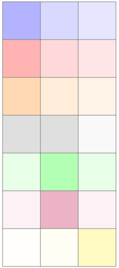
$\phi(\mathsf{X}) = \mathsf{S}^\top =$  $\in \mathbb{R}^{N\times K}$

[10] R. Ying *et al.*, "Hierarchical Graph Representation Learning with Differentiable Pooling," 2018.
[11] F. M. Bianchi *et al.*, "Mincut pooling in Graph Neural Networks," 2019.
[12] C. Bodnar *et al.*, "Deep Graph Mapper: Seeing Graphs through the Neural Lens," 2020.

Learning to pool

Key idea: learn to output $S^\top$ by giving node features X as input to a neural network.

- **DiffPool** [10]: GNN for $S^\top$, regularize with “link prediction” loss;
- **MinCutPool** [11]: MLP for $S^\top$, regularize with “minimum cut” loss (same objective as spectral clustering);
- **Deep Graph Mapper** [12]: combine Mapper [13] and GCN [14] to compute clusters.

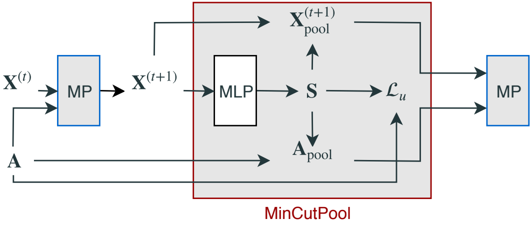
$\phi(X) = S^\top =$  $\in \mathbb{R}^{N \times K}$

[10] R. Ying *et al.*, “Hierarchical Graph Representation Learning with Differentiable Pooling,” 2018.
[11] F. M. Bianchi *et al.*, “Mincut pooling in Graph Neural Networks,” 2019.
[12] C. Bodnar *et al.*, “Deep Graph Mapper: Seeing Graphs through the Neural Lens,” 2020.

MinCut Pooling [11]

- Select: $S^\top = \mathrm{MLP}(X)$
- Reduce: $X' = SX$
- Connect: $A' = SAS^\top$
- MinCut loss: $\mathcal{L}_c = -\frac{Tr(SAS^\top)}{Tr(SDS^\top)}$
- Orthogonality loss:

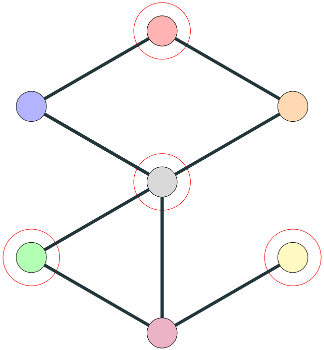
$$\mathcal{L}_o = \left\| \frac{SS^\top}{\|SS^\top\|_F} - \frac{I_K}{\sqrt{K}} \right\|_F$$



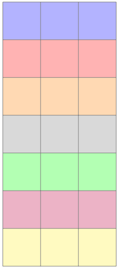
[11] F. M. Bianchi *et al.*, "Mincut pooling in Graph Neural Networks," 2019.

Problem: computing S with neural network is likely to yield a very **dense** matrix.

Can we learn a sparse selection?

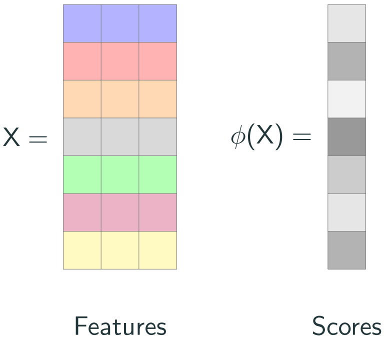


Top-K methods

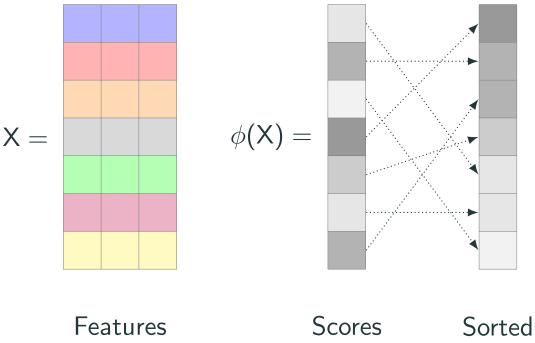
$X =$ 

Features

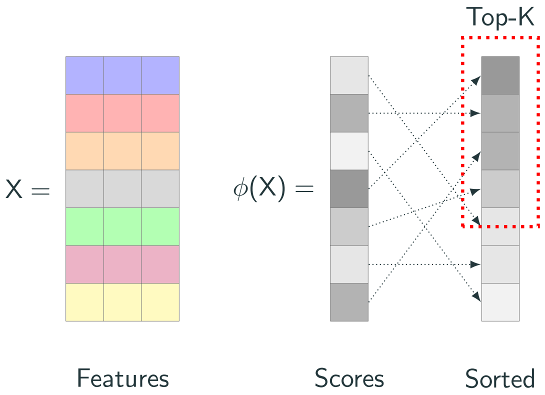
Top-K methods



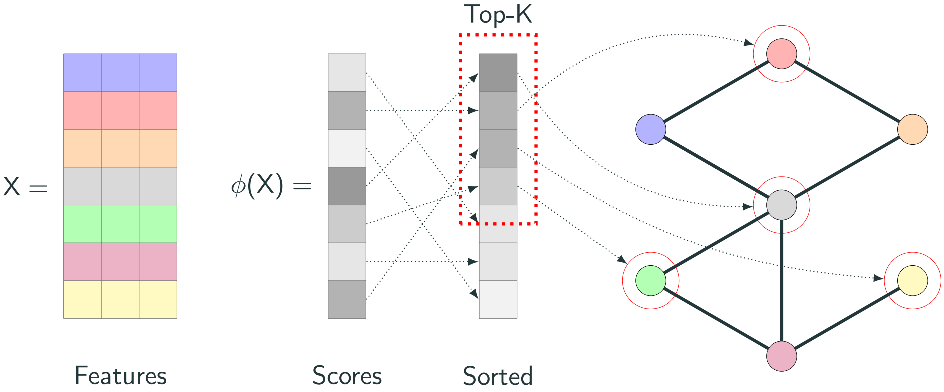
Top-K methods



Top-K methods



Top-K methods



Top-K methods

Different ways of computing the selection indices i:

- Select with a simple linear projection $\theta \in \mathbb{R}^F$ [15];
- Select with a GNN [16];
- Train the selection with a supervised objective (needs ground truth for which nodes to keep) [17].

[15] S. J. Hongyang Gao, "Graph U-Net," 2019.

[16] J. Lee *et al.*, "Self-Attention Graph Pooling," 2019.

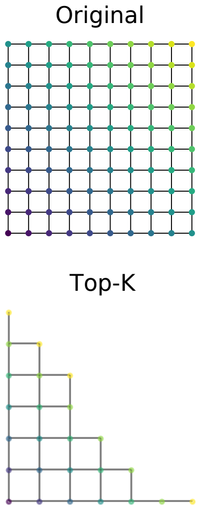
[17] B. Knyazev *et al.*, "Understanding attention in graph neural networks," 2019.

Top-K methods

$$\text{Reduce: } X' = X_i \text{ - Connect: } A' = A_{i,i}$$

Problems:

- Top-k selection is **non-differentiable** (no way of training ϕ). Solved by **gating** (multiplying) the node attributes with the scores.
- Graph is likely to be **disconnected** or simply **cut off** (like in the image on the right). Not really solvable...



Main properties of pooling operators

- **Dense vs. Sparse**: how many nodes are selected for the supernodes (O(N) vs. O(1));
- **Fixed vs. Adaptive**: how many supernodes does the selection compute (*e.g.*, DiffPool/MinCut are fixed);
- **Model-free vs. Model-based**: learn to pool from data or based on a *model* of how to pool;

Global pooling

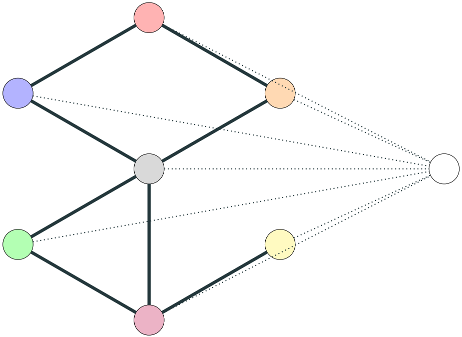
Global Pooling

In CNNs, after convolution, we usually **flatten** out the images to give a vector as input to a MLP:

1	2	3
4	5	6
7	8	9

1	2	3	4	5	6	7	8	9

The graph equivalent must be **invariant to permutations** of the nodes:



Global Pooling

Once again, there are many ways to do this:

- Sum, average, product, max;
- Weighted sum with attention [18];
- Sum and then apply a neural network [19];

---

[18] Y. Li *et al.*, "Gated graph sequence neural networks," 2015.

[19] N. Navarin *et al.*, "Universal readout for graph convolutional neural networks," 2019.

Open questions

- Does pooling really work? [20]
    - Which task benefit from it, *a priori*?
- Can we make a pooling layer that is dense, model-free, and adaptive?

[20] D. Mesquita *et al.*, "Rethinking pooling in graph neural networks," 2020.

[1] D. Grattarola, D. Zambon, F. M. Bianchi, and C. Alippi, "Understanding pooling in graph neural networks,", 2021 (In preparation).

[2] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[3] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[4] L. Palagi, V. Piccialli, F. Rendl, G. Rinaldi, and A. Wiegele, "Computational approaches to max-cut," in *Handbook on semidefinite, conic and polynomial optimization*, Springer, 2012, pp. 821–847.

[5] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi, *Hierarchical representation learning in graph neural networks with node decimation pooling*, 2019. arXiv: 1910.11436 [cs.LG].

[6] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 11, pp. 1944–1957, 2007.

[7] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, 2016, pp. 3844–3852.

[8] E. Luzhnica, B. Day, and P. Lio, "Clique pooling for graph classification," *International Conference of Learning Representations (ICLR) – Representation Learning on Graphs and Manifolds workshop*, 2019.

[9] E. Noutahi, D. Beani, J. Horwood, and P. Tossou, "Towards interpretable sparse graph representation learning with laplacian pooling," *arXiv preprint arXiv:1905.11577*, 2019.

[10] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," *arXiv preprint arXiv:1806.08804*, 2018.

[11] F. M. Bianchi, D. Grattarola, and C. Alippi, "Mincut pooling in graph neural networks," *CoRR*, vol. abs/1907.00481, 2019. arXiv: 1907.00481. [Online]. Available: http://arxiv.org/abs/1907.00481.

[12] C. Bodnar, C. Cangea, and P. Liò, "Deep graph mapper: Seeing graphs through the neural lens," *arXiv preprint arXiv:2002.03864*, 2020.

[13] G. Singh, F. Mémoli, and G. E. Carlsson, "Topological methods for the analysis of high dimensional data sets and 3d object recognition.," in *SPBG*, 2007, pp. 91–100.

[14] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2016.

[15] S. J. Hongyang Gao, "Graph u-net," *Submitted to ICLR*, 2019.

[16] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," *arXiv preprint arXiv:1904.08082*, 2019.

[17] B. Knyazev, G. W. Taylor, and M. R. Amer, "Understanding attention in graph neural networks," *CoRR*, vol. abs/1905.02850, 2019. arXiv: 1905.02850. [Online]. Available: http://arxiv.org/abs/1905.02850.

[18] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[19] N. Navarin, D. Van Tran, and A. Sperduti, "Universal readout for graph convolutional neural networks," in *2019 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2019, pp. 1–7.

[20] D. Mesquita, A. Souza, and S. Kaski, "Rethinking pooling in graph neural networks," *Advances in Neural Information Processing Systems*, vol. 33, 2020.