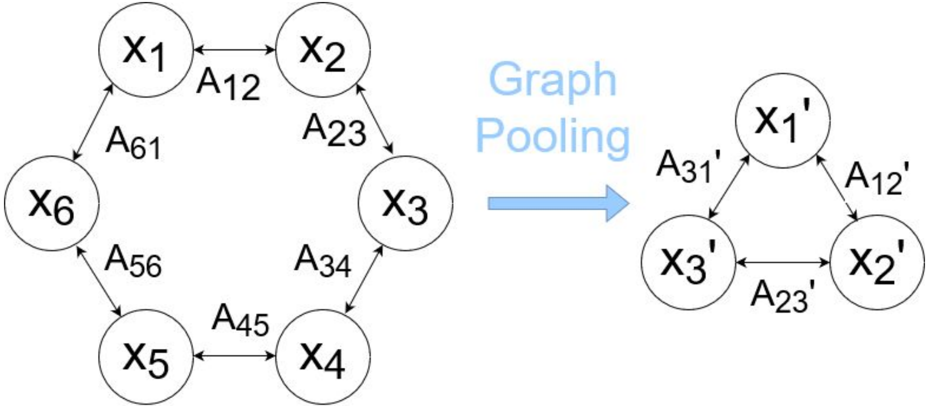
Graph Pooling

Cao Yu
Sep. 3, 2020

What is graph pooling?

Graph neural network (GNN) has been widely used in message propagating between nodes in graph data, obtaining topology-aware node representation.

But under some circumstances (e.g. graph classification), we need to obtain the representation of the whole graph (or higher-level) instead of each raw node and edge. Thus the graph need to be downsampled gradually and finally into representation with smaller scale, which is called **graph pooling**, just like pooling in images.



Two typical kinds of approaches

1) Sorting-based method: ranking nodes and only remain partial of them in each time of pooling

Representative methods: **DiffPool** (Stanford University, NIPS 2018), **MinCutPool** (Politecnico di Milano, ICML 2020)

2) Clustering-based method: cluster original graph into several subgraphs in each time of pooling

Representative methods: **Top-k Pool** (Texas AM University, ICML 2019), **SAGPool** (Korea University, ICML 2019)

Hierarchical Graph Representation Learning with Differentiable Pooling (DiffPool)

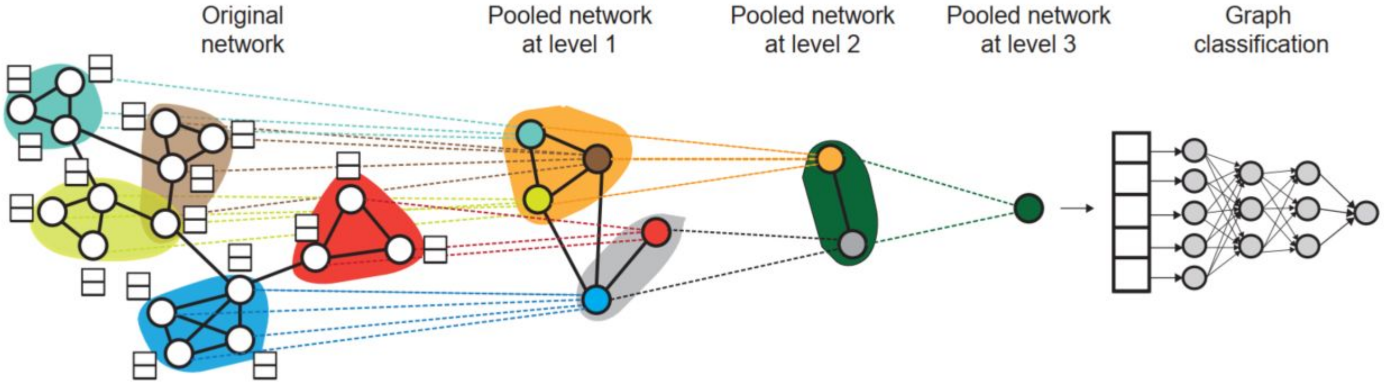
Rex Ying, Jiaxuan You, Christopher Morris, et al.

NIPS2018

Main idea

Using a **learnable assignment matrix** in each layer to transform the representation of all current nodes and edges into a smaller size, in other words, coarsen the graph representation whose node amount is smaller than current node amount.

GNN, who takes information from all nodes and edges, will be stacked together with pooling layer to make sure that each node can obtain the information of whole topological and other nodes.


Original network
Pooled network at level 1
Pooled network at level 2
Pooled network at level 3
Graph classification

DiffPool Pooling Layer

A graph can be represented by G=(A,X), $A \in \{0,1\}^{n\times n}$ is the adjacency matrix and $X \in \mathbb{R}^{n\times d}$ is the node feature matrix.

GNN takes A and X as input and Graph Convolutional Networks (GCNs) can be represented as Z=GNN(A,X)= $\mathrm{ReLU}(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(k-1)}W^{(k-1)})$.

At layer l, an assignment matrix is $S^{(l)} \in \mathbb{R}^{n_l \times n_{l+1}}$, in which n_l and n_{l+1} are nodes (cluster) amount in layer l and layer $l+1$. Then the feature matrix Z after GNN and adjacency matrix can be clustered by $S^{(l)}$

$$\textsc{DiffPool}(A^{(l)}, Z^{(l)}) \quad \Rightarrow \quad \begin{aligned} X^{(l+1)} &= {S^{(l)}}^T Z^{(l)} \in \mathbb{R}^{n_{l+1}\times d}, \\ A^{(l+1)} &= {S^{(l)}}^T A^{(l)} S^{(l)} \in \mathbb{R}^{n_{l+1}\times n_{l+1}} \end{aligned}$$

DiffPool Pooling Layer

To generate the assignment matrix, another GNN is used on the feature matrix and adjacency matrix, along with softmax function.

$$S^{(l)} = \mathrm{softmax}\left(\mathrm{GNN}_{l,\mathrm{pool}}(A^{(l)}, X^{(l)})\right)$$

To better train the pooling layer, an auxiliary link prediction objective is added, encoding the intuition that nearby nodes should be pooled together using Frobenius norm.

$$L_{\mathrm{LP}} = ||A^{(l)}, S^{(l)} {S^{(l)}}^{T}||_F$$

Another auxiliary loss is entropy of cluster assignment so that lower entropy means each cluster is more clearly defined.

$$L_{\mathrm{E}} = \frac{1}{n} \sum_{i=1}^{n} H(S_i)$$

These two loss will be added to the final training loss.

Spectral Clustering with Graph Neural Networks for Graph Pooling (MinCutPool)

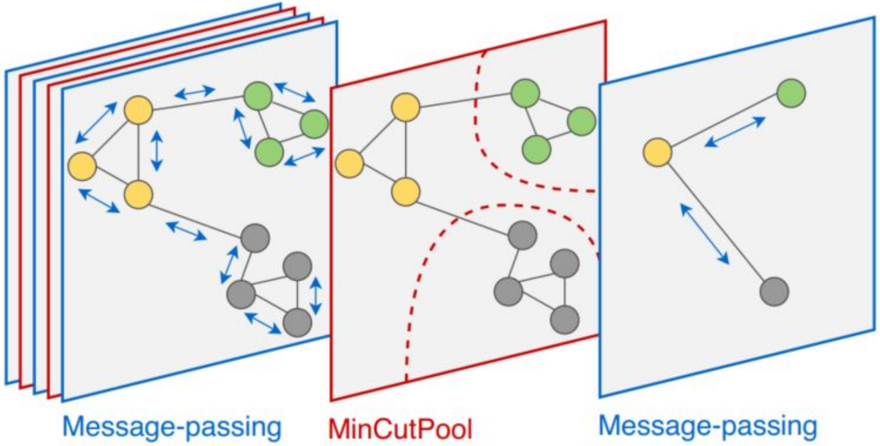
Filippo Maria Bianchi, Daniele Grattarola, Cesare Alippi

ICML2020

Main idea

Also based on cluster approach, it solves the clustering by regarding it as a K-way normalized MinCut problem, in which splitting the graph into K disjoint subgraphs by removing the minimum volume of edges. It is equivalent to

$$\frac{1}{K}\sum_{k=1}^{K}\frac{\text{links}(\mathcal{V}_k)}{\text{degree}(\mathcal{V}_k)} = \frac{1}{K}\sum_{k=1}^{K}\frac{\sum_{i,j\in\mathcal{V}_k}\mathcal{E}_{i,j}}{\sum_{i\in\mathcal{V}_k,j\in\mathcal{V}\backslash\mathcal{V}_k}\mathcal{E}_{i,j}}$$



MinCut Problem

A graph can be represented by G=($\mathbf{A}$, $\mathbf{X}$) in which $\mathbf{A} \in \mathbb{R}^{N\times N}$ is adjacency matrix and $\mathbf{X} \in \mathbb{R}^{N\times F}$ is node feature matrix. Given a cluster matrix assignment matrix $\mathbf{C} \in \{0,1\}^{N\times K}$, the MinCut problem is expressed as

$$\text{maximize} \quad \frac{1}{K}\sum_{k=1}^{K} \frac{\mathbf{C}_k^T \mathbf{A} \mathbf{C}_k}{\mathbf{C}_k^T \mathbf{D} \mathbf{C}_k},$$

$$\text{s.t.} \quad \mathbf{C} \in \{0,1\}^{N\times K}, \quad \mathbf{C}\mathbf{1}_K = \mathbf{1}_N$$

A near-optimal can be obtained by

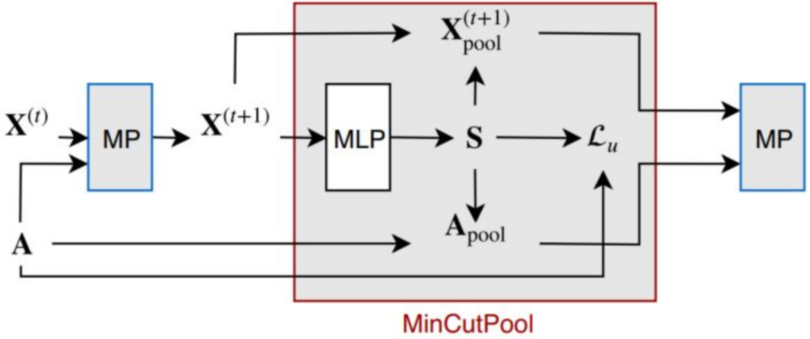
$$\underset{\mathbf{Q}\in\mathbb{R}^{N\times K}}{\text{arg max}} \quad \frac{1}{K}\sum_{k=1}^{K} \mathbf{Q}_k^T \mathbf{A} \mathbf{Q}_k$$

$$\text{s.t.}\ \mathbf{Q} = \mathbf{C}(\mathbf{C}^T\mathbf{D}\mathbf{C})^{-\frac{1}{2}},\ \mathbf{Q}^T\mathbf{Q} = \mathbf{I}_K$$

Such problem is still no-convex, but it can be approximated by gradient descent.

MinCutPool Layer

Similar to DiffPool, GNN will be used before pooling $\bar{\mathbf{X}} = \mathrm{GNN}(\mathbf{X}, \tilde{\mathbf{A}}; \mathbf{\Theta}_{\mathrm{GNN}})$



A assignment matrix $\mathbf{S}$ is generated via MLP, $\mathbf{S} = \mathrm{MLP}(\bar{\mathbf{X}}; \mathbf{\Theta}_{\mathrm{MLP}})$, softmax is used to guarantee that $\mathbf{s}_{ij} \in [0, 1]$ and the sum of each row is 1.

Graph will be corsened using $\mathbf{S}$ $\mathbf{A}^{pool} = \mathbf{S}^T \tilde{\mathbf{A}} \mathbf{S}; \quad \mathbf{X}^{pool} = \mathbf{S}^T \mathbf{X},$

New adjacency matrix is zero-diagonal $\hat{\mathbf{A}} = \mathbf{A}^{pool} - \mathbf{I}_K \mathrm{diag}(\mathbf{A}^{pool}); \quad \tilde{\mathbf{A}}^{pool} = \hat{\mathbf{D}}^{-\frac{1}{2}} \hat{\mathbf{A}} \hat{\mathbf{D}}^{-\frac{1}{2}}$

MinCutPool Layer

The unsupervised loss function is composed of two terms,

$$\mathcal{L}_u = \mathcal{L}_c + \mathcal{L}_o = \underbrace{-\frac{Tr(\mathbf{S}^T\tilde{\mathbf{A}}\mathbf{S})}{Tr(\mathbf{S}^T\tilde{\mathbf{D}}\mathbf{S})}}_{\mathcal{L}_c} + \underbrace{\left\| \frac{\mathbf{S}^T\mathbf{S}}{\|\mathbf{S}^T\mathbf{S}\|_F} - \frac{\mathbf{I}_K}{\sqrt{K}} \right\|_F}_{\mathcal{L}_o},$$

$\mathcal{L}_c$ is cut loss that encourages strongly connected nodes to be clustered together, whose maximum value is 0 when cluster assignments are orthogonal. $\mathcal{L}_o$ is orthogonality loss encourages cluster to be orthogonal and clusters in similar size.

The unsupervised loss from each layer will be added to the original training loss for the specific task.

Graph U-Nets (TopK Pool)

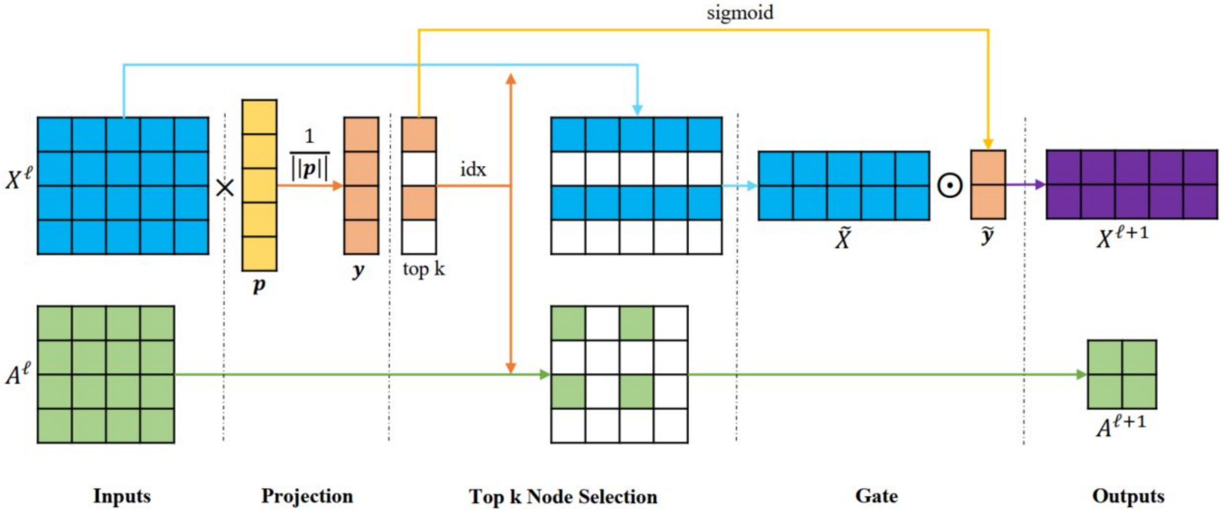
Hongyang Gao, Shuiwang Ji

ICML2019

Main idea

As a sort-based method, it uses a projection vector to transform nodes into corresponding scores in each pooling. Then only nodes with TopK scores along with related edges are remained as the input of next processing.

It should be noted that the score is only based on representation of each node independently.



TopK Pooling Layer

Similarly, given a graph $G=(\mathbf{A}, \mathbf{X})$, a trainable projection vector $\mathbf{p}$ is used to get the scores for each node $\mathbf{y} = X^\ell \mathbf{p}^\ell / \|\mathbf{p}^\ell\|$

Then top-k node index idx will be ranked based on $\mathbf{y}$ $\mathrm{idx} = \mathrm{rank}(\mathbf{y}, k)$

Corresponding node features and edges will be chosen

$$\tilde{X}^\ell = X^\ell(\mathrm{idx}, :) \qquad A^{\ell+1} = A^\ell(\mathrm{idx}, \mathrm{idx})$$

The scores y after activation will be used to weight $\tilde{X}^\ell$ as a gate to obtain the node feature in the next layer $\tilde{\mathbf{y}} = \mathrm{sigmoid}(\mathbf{y}(\mathrm{idx}))$ $X^{\ell+1} = \tilde{X}^\ell \odot (\tilde{\mathbf{y}}\mathbf{1}_C^T)$

Such **gate is essential** which can make the whole procedure differentiable, otherwise the top-k selection will be a discrete operation.

TopK Pooling Layer

Different from cluster-based method, there is no auxiliary loss for sort-based method, the whole model will be trained end-to-end.

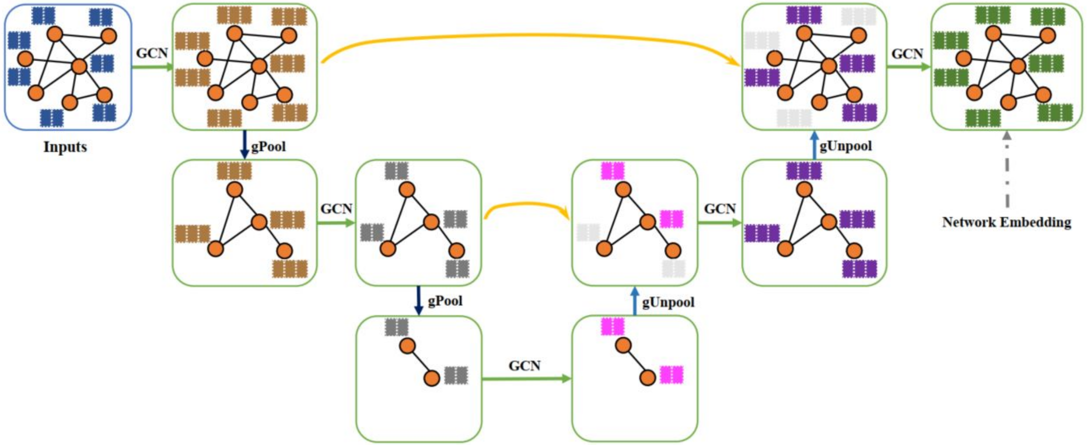
The reason is that there is no significant loss for designing the ranking function.

Unpooling and encoder-decoder architecture

Actually this paper also introduce a graph encoder-decoder frame, in which pooling and unpooling are two important components.

For unpooling (upsampling) on the same data, a distribute function is used in which the graph structure before pooling is remained and only node representation who are selected by TopK are remained in corresponding position, while features of other nodes are zero

$$X^{\ell+1} = \text{distribute}(0_{N\times C}, X^{\ell}, \text{idx})$$



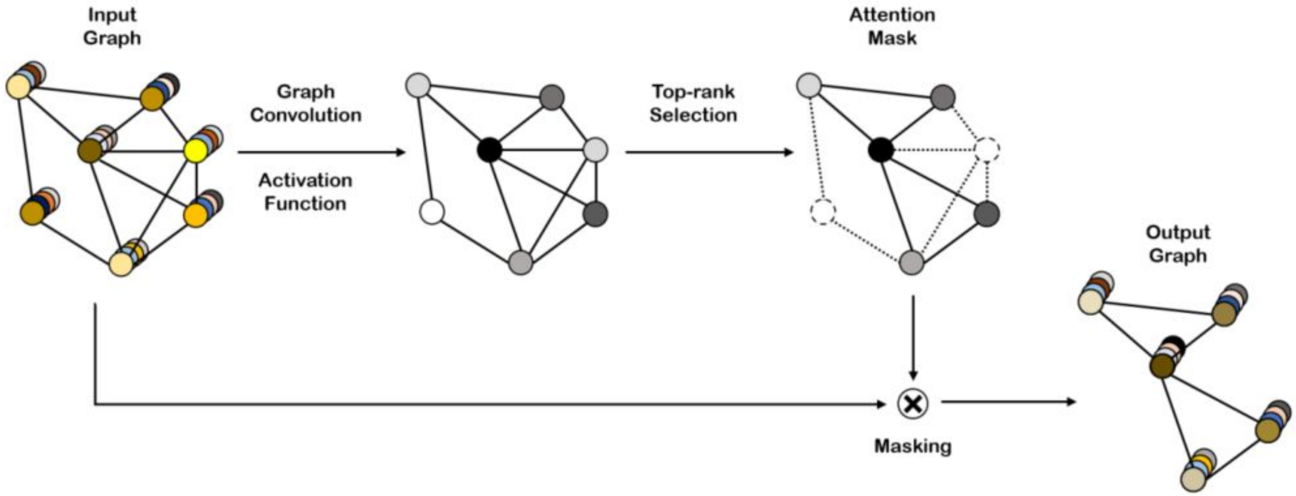
Self-Attention Graph Pooling (SAG Pool)

Junhyun Lee, Inyeop Lee, Jaewoo Kang

ICML2019

Main idea

As a sort-based method, to sort the nodes, it obtains ranking scores based on GNN who considers the topology of graph rather than barely independent node features as TopK Pool.



SAG Pooling Layer

Given a graph G=(**A, X**) in, the self-attention score $Z \in \mathbb{R}^{N\times 1}$is calculated by GCN

$$Z = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta_{att})$$

Similarly, indices of nodes with high scores are selected based on a ratio k

$$\mathrm{idx} = \text{top-rank}(Z, \lceil kN \rceil), \quad Z_{mask} = Z_{\mathrm{idx}}$$

These scores for selected nodes are also gating weights for node features after filtering, such procedure is the same as TopK Pooling.

$$X' = X_{\mathrm{idx},:}, \; X_{out} = X' \odot Z_{mask}, \; A_{out} = A_{\mathrm{idx},\mathrm{idx}}$$

SAG Pooling Layer

There are also variants for calculating the ranking scores.

1) Condering two-hop neighbors by adding the square of adjacency matrix

$$Z = \sigma(\mathrm{GNN}(X, A + A^2))$$

2) Stacking GNN layers for indirect aggregation of multi-hop nodes

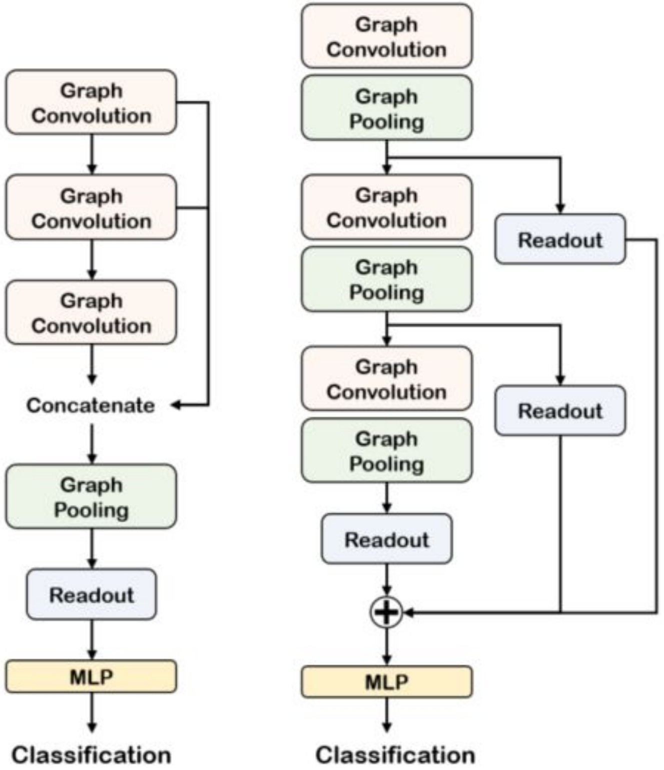
$$Z = \sigma(\mathrm{GNN}_2(\sigma(\mathrm{GNN}_1(X, A)), A))$$

3) Average attention score among M GNNs (like ensemble)

$$Z = \frac{1}{M} \sum_{m} \sigma(\mathrm{GNN}_m(X, A))$$

Model Architecture

It also proposes two architectures, one is **global pooling** in which there is only one pooling layer after several stacked GCN layer, and the other one is hierarchical pooling in which a pooling layer is stacked together with a GCN layer.



Performance comparison of methods

The most common approach is using graph classification task, each graph will be transformed into a fixed-length feature then using MLP to make classification.

Some statistics of common datasets are shown as below

Dataset	samples	classes	avg. nodes	avg. edges	node labels
DD	1178	2	284.32	715.66	yes
PROTEINS	1113	2	39.06	72.82	yes
NCI1	4410	2	29.87	32.30	yes
NCI109	4127	2	29.68	32.13	yes
Mutagencity	4337	2	30.32	30.77	yes
COLLAB	5000	3	74.49	2457.78	no
Reddit-binary	2000	2	429.63	497.75	no

Performance comparison of methods

Classification accuracy of all above models and baseline avg-Pool (average pooling after the same number of GCNs)

Generally speaking, clustering-based method is superior to sort-based ones, with the cost of higher complexity.
Avg method is comparable when graph scale is small, but it fails when graph is large.

Methods	DD	PROTEINS	NCI1	NCI109	Mutagenicity	COLLAB	Reddit-binary
AvgPool	73.05%	71.55%	70.89%	69.62%	**79.63%**	70.62%	82.41%
DiffPool	79.30%	72.70%	-	-	77.60%	81.80%	80.80%
MincutPool	**79.56%**	**75.88%**	**76.77%**	**74.97%**	79.24%	**82.89%**	**83.35%**
TopK Pool	75.01%	71.10%	67.02%	66.12%	73.67%	77.56%	74.70%
SAG Pool	76.45%	71.86%	67.45%	67.86%	74.52%	79.20%	73.90%

Complexity comparison

The node number in original graph and new graph after pooling is N and K respectively, d is node feature dimension of current layer.

DiffPool: space $O(Kd)$ (GNN), time $O(NK(N+K+d)+N^2(2N+d))$ (GNN to obtain S and cluster edges).

MinCutPool: space $O(NK)$ (matrix $\mathbf{S}$), time $O(NK(N+K))$ (loss term L_c).

TopK Pool: space $O(d)$ (projection vector $\mathbf{p}$), time $O(Nd+NlogN+Kd)$.

SAG Pool: space space $O(Kd)$ (GNN), time $O(2N^3)$ (GNN to obtain the ranking scores).

Thanks and QA