Australian National University

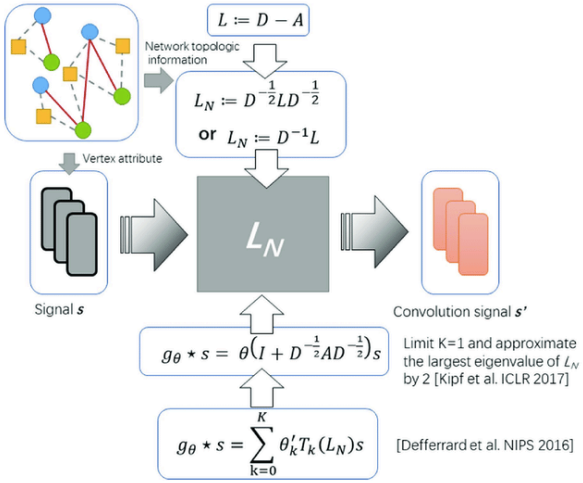
Simple Spectral Graph Convolution

Hao Zhu, Piotr Koniusz

Problems

- What is oversmoothing?
- Spectral Analysis of oversmoothing is right?
- How to relieve/avoid that?

Chebshev Net



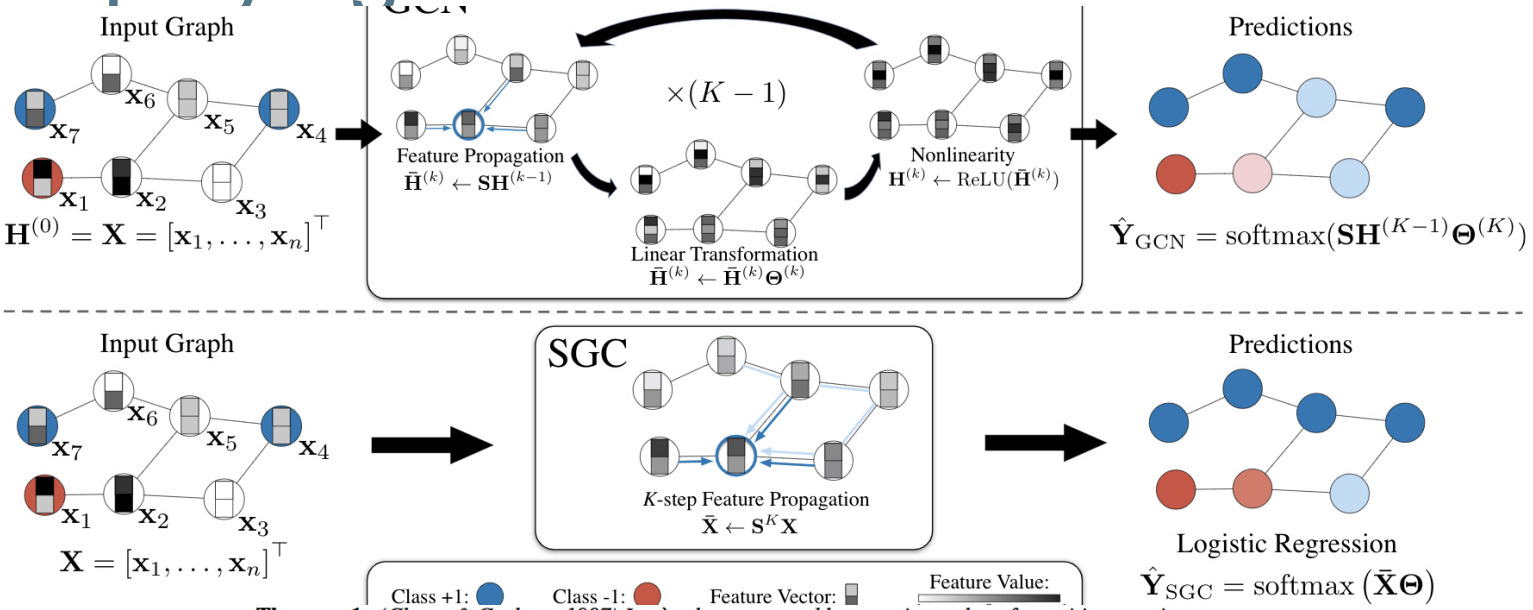
Spectral Graph Convolution (Defferrard et al., 2016). We consider spectral convolutions on graphs defined as the multiplication of a signal $\mathbf{x} \in \mathbb{R}^n$ with a filter g_θ parameterized by $\boldsymbol{\theta} \in \mathbb{R}^n$ in the Fourier domain:

$$g_\theta(\mathbf{L}) * x = \mathbf{U} g_\theta^*(\mathbf{\Lambda}) \mathbf{U}^\top \mathbf{x}, \tag{1}$$

where the parameter $\boldsymbol{\theta} \in \mathbb{R}^n$ is a vector of spectral filter coefficients. We can understand g_θ as a function operating on eigenvalues of $\mathbf{L}$, that is $g_\theta^*(\mathbf{\Lambda})$. To avoid eigendecomposition, $g_\theta(\mathbf{\Lambda})$ can be approximated by a truncated expansion in terms of Chebyshev polynomials $T_k(\mathbf{\Lambda})$ up to the K-th order (Defferrard et al., 2016):

$$g_\theta^*(\mathbf{\Lambda}) \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\mathbf{\Lambda}}), \tag{2}$$

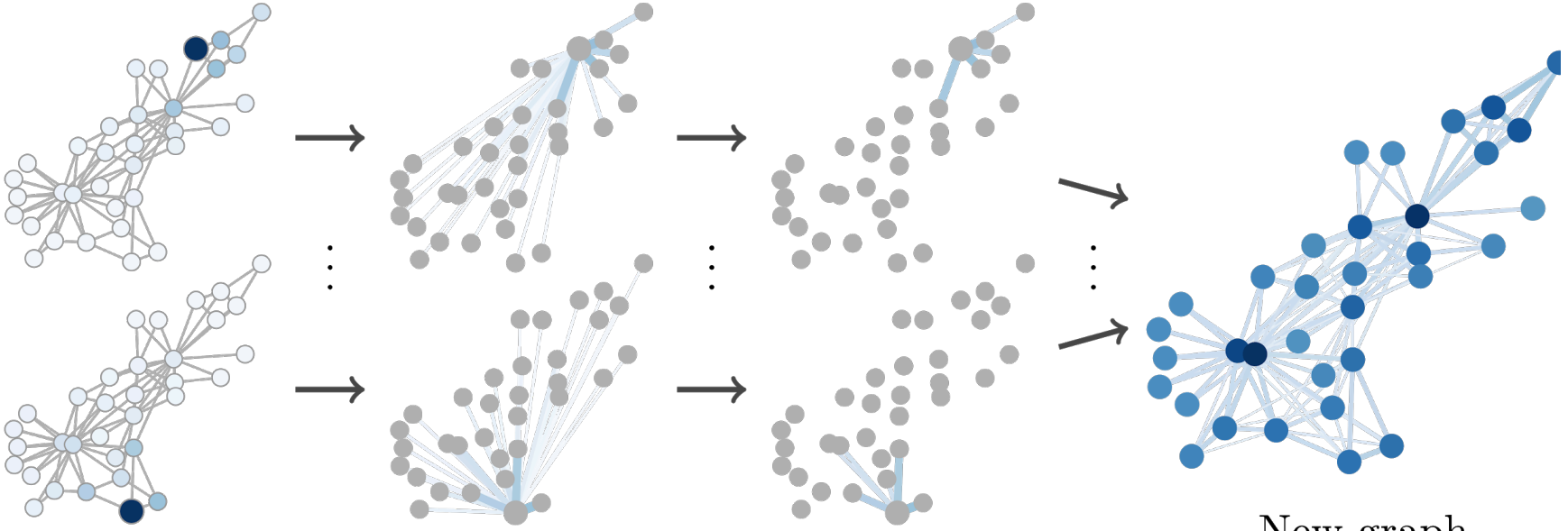
Simplifying GCNs



Theorem 1. *(Chung & Graham, 1997) Let λ_2 denote second largest eigenvalue of transition matrix $\tilde{\mathbf{T}} = \mathbf{D}^{-1}\mathbf{A}$ of a non-bipartite graph, $\mathbf{p}(t)$ be the probability distribution vector and $\boldsymbol{\pi}$ the stationary distribution. If walk starts from the vertex i, $p_i(0) = 1$, then after t steps for every vertex:*

$$|p_j(t) - \pi_j| \leq \sqrt{\frac{d_j}{d_i}}\lambda_2^t, \tag{6}$$

Graph Diffusion Convolution



Computational and storage Cost

Personalized Propagation of Neural Prediction (PPNP)

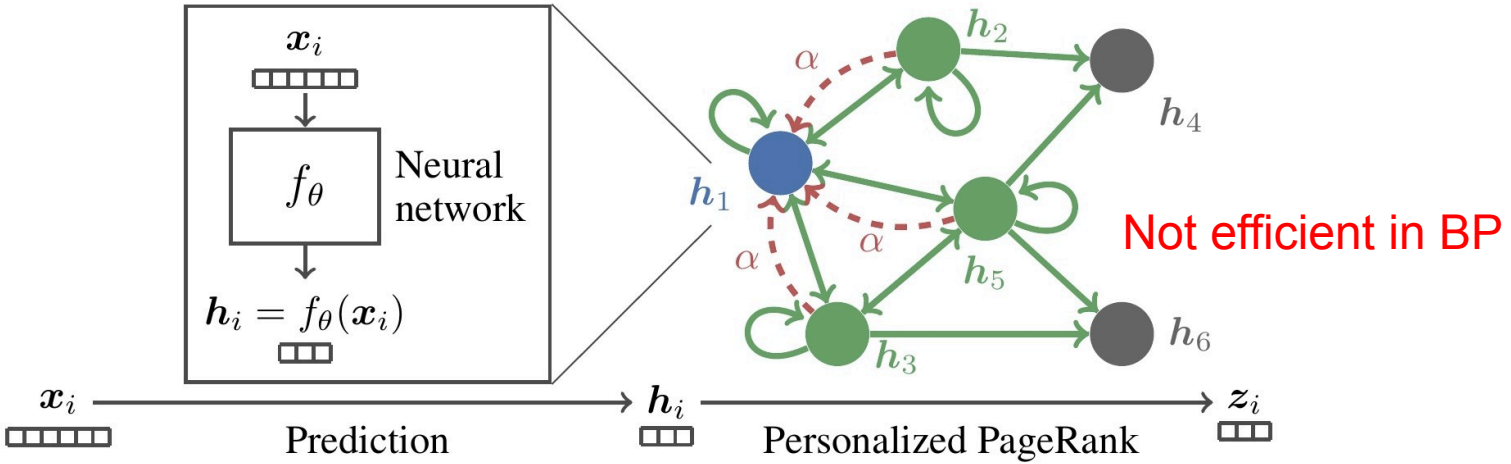


Figure 1: Illustration of (approximate) personalized propagation of neural predictions (PPNP, APPNP). Predictions are first generated from each node's own features by a neural network and then propagated using an adaptation of personalized PageRank. The model is trained end-to-end.

Markov Diffusion Kernel

Claim I. Our filter, by design, will give the highest weight to the closest neighborhood of a node as neighborhoods $\mathcal{N}$ of diffusion steps $k = 0, \cdots, K$ obey: $\mathcal{N}(\widetilde{\mathbf{T}}^0) \subseteq \mathcal{N}(\widetilde{\mathbf{T}}^1) \subseteq \cdots \subseteq \mathcal{N}(\widetilde{\mathbf{T}}^K) \subseteq \mathcal{N}(\widetilde{\mathbf{T}}^\infty)$. That is, smaller neighbourhoods belong to larger neighbourhoods too.

Claim II. As $K \to \infty$, the ratio of energies contributed by S^2GC to SGC is 0. Thus, the energy of infinite-dimensional receptive field (largest k) will not dominate the sum energy of our filter. Thus, S^2GC can incorporate larger receptive fields without overbearing contributions of smaller receptive fields. This is substantiated by Table 8 where we achieve $K = 16$ while SGC achieves $K = 4$.

words, two nodes are close if they are in the same cluster which has a consistent local structure. More precisely, the diffusion distance at time K between nodes i and j is defined as follows:

$$d_{ij}(K) = \|\mathbf{x}_i(K) - \mathbf{x}_j(K)\|_2^2, \tag{8}$$

where the average visiting rate $\mathbf{x}_i(K)$ after K steps for a process that started at time $k = 0$ is computed as follows:

$$\mathbf{x}_i(K) = \frac{1}{K} \sum_{k=1}^{K} \mathbf{T}^k \mathbf{x}_i(0). \tag{9}$$

By defining $\mathbf{Z}(K) = \frac{1}{K} \sum_{k=1}^{K} \mathbf{T}^k$, we reformulate Eq. 8 as a metric given as:

$$d_{ij}(K) = \|\mathbf{Z}(K)(\mathbf{x}_i(0) - \mathbf{x}_j(0))\|_2^2. \tag{10}$$

The underlying feature map of Markov Diffusion Kernel (MDK) is given as $\mathbf{Z}(K)\mathbf{x}_i(0)$ for node i.

Simple Spectral Graph Convolution

Based on the aforementioned Markov Diffusion Kernel, we include self-loops and we propose the Simple Spectral Graph Convolution (S^2GC) network with a softmax after a linear layer:

$$\hat{Y} = \text{softmax}\left(\frac{1}{K}\sum_{k=0}^{K}\widetilde{\mathbf{T}}^k\mathbf{X}\mathbf{W}\right). \tag{11}$$

$$\hat{Y} = \text{softmax}\left(\frac{1}{K}\sum_{k=1}^{K}\left((1-\alpha)\,\widetilde{\mathbf{T}}^k\mathbf{X} + \alpha\mathbf{X}\right)\mathbf{W}\right). \tag{13}$$

Relation to other methods

Relation of S^{2}GC to APPNP. Let us define $\mathbf{H}^0 = \mathbf{XW}$ as we use the linear step in our S^{2}GC. Then and only then, for $l = 0$ and $\mathbf{H}^0 = \mathbf{XW}$, APPNP expansion yields $\mathbf{H}^1 = (1-\alpha)\widetilde{\mathbf{T}}\mathbf{XW} + \alpha\mathbf{XW} = ((1-\alpha)\widetilde{\mathbf{T}} + \alpha\mathbf{I})\mathbf{XW}$ which is equal to our $\mathbf{Z}(1)\mathbf{XW} = (\sum_{k=0}^{K}\widetilde{\mathbf{T}}^k)\mathbf{XW} = \widetilde{\mathbf{T}}\mathbf{X} + \mathbf{X} = (\widetilde{\mathbf{T}} + \mathbf{I})\mathbf{XW}$ if $\alpha = 0.5$, $K = 1$, except for scaling (constant) of $\mathbf{H}^1$.

In contrast, for $l = 1$ and general case $\mathbf{H}^0 = f(\mathbf{X}; \mathbf{W})$, APPNP yields $\mathbf{H}^2 = (1-\alpha)^2\widetilde{\mathbf{T}}^2 f(\mathbf{X}; \mathbf{W}) + (1-\alpha)\alpha\widetilde{\mathbf{T}} f(\mathbf{X}; \mathbf{W}) + \alpha f(\mathbf{X}; \mathbf{W})$ from which is is easy to note specific weight coefficients $(1-\alpha)^2$, $(1-\alpha)\alpha$ and α associated with 2-, 1-, and 0-hops. This shows that the APPNP expansion is very different to S^{2}GC expansion in Eq. 13. In fact, S^{2}G and APPNP are only equivalent if $\alpha = 0.5$, $K = 1$ and a linear transformation f is used.

Moreover, APPNP assumes $\mathbf{H}^0 = f(\mathbf{X}; \mathbf{W}) = \mathrm{ReLU}(\mathbf{XW})$, thus their optimizer has to back-propagate through $f(\mathbf{X}; \mathbf{W})$ to obtain $\frac{\partial f}{\partial \mathbf{W}}$ and multiply this with the above expansion *e.g.*, $\frac{\partial \mathbf{H}^2}{\partial \mathbf{W}} = (1-\alpha)^2\widetilde{\mathbf{T}}^2 f'(\mathbf{X}; \mathbf{W}) + (1-\alpha)\alpha\widetilde{\mathbf{T}} f'(\mathbf{X}; \mathbf{W}) + \alpha f'(\mathbf{X}; \mathbf{W})$.

In contrast, we use the linear function $\mathbf{XW}$. Thus, $\frac{\partial \mathbf{XW}}{\partial \mathbf{W}}$ yields $\mathbf{X}$. Thus, the multiplication of our expansion with $\mathbf{X}$ for the backprop step is in fact obtained in the forward pass which makes our approx very fast for large graphs.

Computational and storage Costs

Table 1: Computational and storage complexities $\mathcal{O}(\cdot)$.

Stage	Complexity	APPNP	GDC	SGC	S^2GC
Forward Propagation	Computation Cost	$K\|E\|d + Knd$	$\approx K\|E\|n$	$K\|E\|d$	$K\|E\|d + Knd$
	Storage Cost	$nd + \|E\|$	$\approx n^2$	$nd + \|E\|$	$nd + \|E\|$
Backward Propagation	Computation Cost	$K\|E\|d$	0	0	0
	Storage Cost	$nd + \|E\|$	0	0	0

Node Clustering

Table 3: Clustering performance with three different metrics on four datasets.

Methods	Input	Cora			Citeseer			Pubmed			Wiki		
		Acc%	NMI%	F1%	Acc%	NMI%	F1%	Acc%	NMI%	F1%	Acc%	NMI%	F1%
k-means	Feature	34.65	16.73	25.42	38.49	17.02	30.47	57.32	29.12	57.35	33.37	30.20	24.51
Spectral-f	Feature	36.26	15.09	25.64	46.23	21.19	33.70	59.91	32.55	58.61	41.28	43.99	25.20
Spectral-g	Graph	34.19	19.49	30.17	25.91	11.84	29.48	39.74	3.46	51.97	23.58	19.28	17.21
DeepWalk	Graph	46.74	31.75	38.06	36.15	9.66	26.70	61.86	16.71	47.06	38.46	32.38	25.74
GAE	Both	53.25	40.69	41.97	41.26	18.34	29.13	64.08	22.97	49.26	17.33	11.93	15.35
VGAE	Both	55.95	38.45	41.50	44.38	22.71	31.88	65.48	25.09	50.95	28.67	30.28	20.49
ARGE	Both	64.00	44.90	61.90	57.30	35.00	54.60	59.12	23.17	58.41	41.40	39.50	38.27
ARVGE	Both	62.66	45.28	62.15	54.40	26.10	52.90	58.22	20.62	23.04	41.55	40.01	37.80
AGC	Both	68.92	53.68	65.61	67.00	41.13	62.48	69.78	31.59	68.72	47.65	45.28	40.36
S^2GC	Both	**69.60**	**54.71**	**65.83**	**69.11**	**42.87**	**64.65**	**70.98**	**33.21**	**70.28**	**52.67**	**49.62**	**44.31**

Node Classification

Table 4: Test Micro F1 Score (%) averaged over 10 runs on Reddit. Performance of other models are cited from their original papers.

Setting	Model	Test F1
Supervised	SAGE-mean	95.0
	SAGE-LSTM	**95.4**
Unsupervised	SAGE-GCN	93.0
	FastGCN	93.7
No Learning	SAGE-GCN	90.8
	DGI	94.0±0.001
	SGC	94.9±0.001
	S^2GC	**95.3±0.001**

Table 5: Test accuracy (%) averaged over 10 runs on citation networks.

methods	Cora	Citeseer	Pubmed
GCN	81.4± 0.4	70.9± 0.5	79.0± 0.4
GAT	83.3± 0.7	72.6± 0.6	78.5± 0.3
FastGCN	79.8± 0.3	68.8± 0.6	77.4± 0.3
GIN	77.6± 1.1	66.1± 0.9	77.0± 1.2
DGI	82.5± 0.7	71.6± 0.7	78.4± 0.7
SGC	81.0± 0.02	71.9± 0.08	78.9± 0.03
MixHop	81.8±0.6	71.4±0.8	80.0±1.1
APPNP	**83.3±0.5**	71.7±0.6	80.1±0.2
Chebynet	78.0± 0.4	70.1± 0.5	78.0± 0.4
AR filter	80.8± 0.02	69.3± 0.15	78.1± 0.01
Ours	83.0± 0.02	**73.6± 0.09**	**80.4± 0.02**

Table 6: Test accuracy (%) averaged over 10 runs on the large-scale OGB node property prediction.

methods	Products	Mag	Arxiv
MLP	61.06±0.08	26.92±0.26	55.50±0.23
GCN	75.64±0.21	30.43±0.25	71.74±0.29
GraphSage	**78.29±0.16**	31.53±0.15	71.49±0.27
Softmax	47.70±0.03	24.13±0.03	52.77±0.56
SGC	68.87± 0.01	29.47±0.03	68.78±0.02
S^2GC	70.22± 0.01	32.47±0.11	70.15±0.13
S^2GC+MLP	74.84±0.20	**32.72±0.23**	**72.01±0.25**

Text Classification

Table 7: Test accuracy on the document classification task.

Model	20NG	R8	R52	Ohsumed	MR
Text GCN	87.9 ± 0.2	97.0 ± 0.2	93.8 ± 0.2	68.2 ± 0.4	76.3 ± 0.3
SGC	88.5 ± 0.1	97.2 ± 0.2	94.0 ± 0.2	$\mathbf{68.5 \pm 0.3}$	75.9 ± 0.3
S^2GC	$\mathbf{88.6 \pm 0.1}$	$\mathbf{97.4 \pm 0.1}$	$\mathbf{94.5 \pm 0.2}$	$\mathbf{68.5 \pm 0.1}$	$\mathbf{76.7 \pm 0.0}$

APPNP failed in this task

Parameter Analysis

Table 8: Summary of classification accuracy (%) results with various depths. In the linear model, filter parameter K is equivalent with the number of layers.

Dataset	Method	Layers (K)					
		2	4	8	16	32	64
Cora	GCN	81.1	80.4	69.5	64.9	60.3	28.7
	SGC	80.8	81.5	80.7	79.0	75.9	66.8
	S^2GC	76.5	79.8	82.5	**83.0**	82.2	80.0
Citeseer	GCN	70.8	67.6	30.2	18.3	25.0	20.0
	SGC	71.9	72.6	73.1	72.2	70.6	69.2
	S^2GC	70.9	72.7	72.7	**73.4**	73.1	73.2
Pubmed	GCN	79.0	76.5	61.2	40.9	22.4	35.3
	SGC	79.2	79.7	78.4	76.4	71.6	68.6
	S^2GC	77.6	78.7	79.4	**80.6**	78.0	74.9

Table 9: Classification accuracy (%) results with different α.

Dataset	0.0	0.05	0.1	0.15
cora	82.9	83.0	82.6	81.9
citeseer	73	73.4	73.3	72.9
pubmed	80.4	80.6	79.7	79.0

Future works

- Linear GNNs (SGC and ours) and explainable models
- Down Stream Tasks
- Large Scale Graph Convolution
- Learnable SSGC

Thanks for Your Attention!