Revisiting GNNs: Graph Filtering Perspective

Hoang NT [1,2], Takanori Maehara [3], and Tsuyoshi Murata [1]

[1] Tokyo Institute of Technology [2] RIKEN AIP [3] Facebook

Tokyo Tech RIKEN

Graph Neural Network (Vertex)

Input: Graph G = (A, V),
label set Y, and
feature set X: $V \to \mathbb{R}^d$ (optional)

Find: Labeling function f: $V \to Y$

(Perozzi et al, 2014)

Hey, spectral clustering for unsupervised vertex classification is unscalable 😞, let's use random walks and Skipgram! 👍

Awesome! 😮
Can we combine vertex features (X) to get better performace?

(Yang et al, 2015)

Sure we can! Use semi-supervised learning! Let's use two losses: An unsupervised loss to capture the structural similarity and a supervised loss for features similarity. 😎

Something like this:

$$\mathcal{L} = \mathcal{L}_s + \lambda \mathcal{L}_u$$

I see. Tuning λ can be tricky though... 😖

(Kipf & Welling, 2016)

Forget about λ 😉, we can adapt the local filtering techniques in GSP to come up with a function f(X,A) directly predict vertex labels.

I call this Graph Convolutional Network (GCN):

$$\text{label} = \text{softmax}(\hat{A}\sigma(\hat{A}XW_0)W_1)$$

Great performance! But now we are back to the scalability problem. 😟

(Hamilton et al, 2017)

Use GraphSAGE! - Our message passing model. 😇
The neighbors are sampled so we can do batching!

(Chen et al, 2018)

FastGCN can compute GCN fast! 🌟

These schemes are similar to feature propagation...

(Buchnik & Cohen, 2018)

Linear feature propagation has similar performance as GCN! There is a strong connection to PageRank!

(Klicpera et al, 2019)

Predict then propagate decouples feature transformation and prediction making classifcation scalable.

(Xu et al, 2019)

Activation function for GCN might not be useful.

Simple Graph Convolution model is much faster:

$$\text{label} = \text{softmax}(\hat{A}^k XW_0)$$

(Klicpera et al, 2019) and (Xu et al, 2019) are essentially the same. One is bottom-up and the other is top-down.

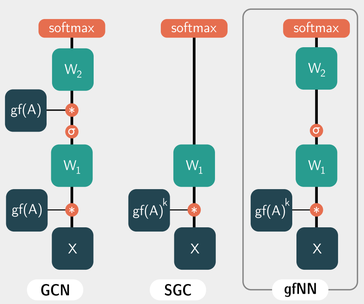
So how to understand the mechanism of these different models?

(Li et al, 2019), ***(NT & Maehara, 2019)***

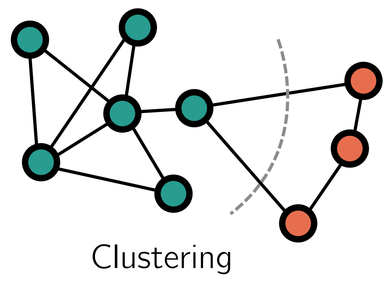
We study these model from the GSP view where the main objects are the signal filters defined from the graph structure (matrix A).

(NT et al., 2020)

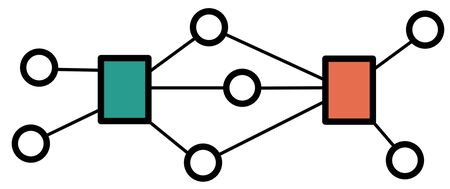
We study under the following model:



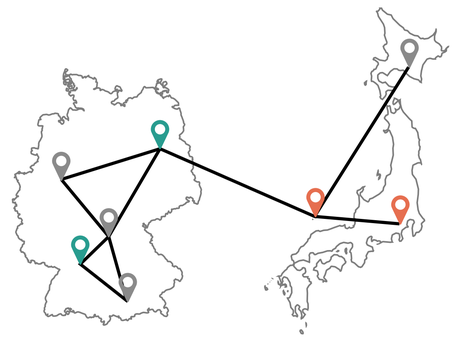
Why classify vertices on a graph?



Clustering



Document Classification



Geolocation minining

Multi-sensor forecast
Traffic management
Scene understanding
Point-cloud segmentation
...

Graph Signal Processing

General Problem	Example	Graph Signal Processing
Graph : $G = (A, V)$	*Citation network*: V = {ordered set of papers}. A[i,j] = 1 if paper "i" cites "j" else 0.	G defines a building block for a signal filter (Laplacian Matrix) or graph shift (Adjacency Matrix)
Feature $X : V \to R^d$	Y = {categories of papers, e.g. ML}.	Y = {signal to be recovered}.
Label set $Y; \ell : V \to Y$	X: Each vertex is associated with a content vector (e.g. word association vector).	X: Partially observed signals.
Task: Approximate the labeling function.	Find the categories for unlabeled papers.	Recover missing signals in Y or X.

Matries

$$L = D - A$$
$$\mathcal{L} = D^{-1/2} L D^{-1/2}$$

Graph Fourier Transformation

$$\mathcal{L} = U \Lambda U^\top$$
$$\hat{X} = U^\top X$$
$$X = U\hat{X}$$

Low-Pass Filtering by top-k

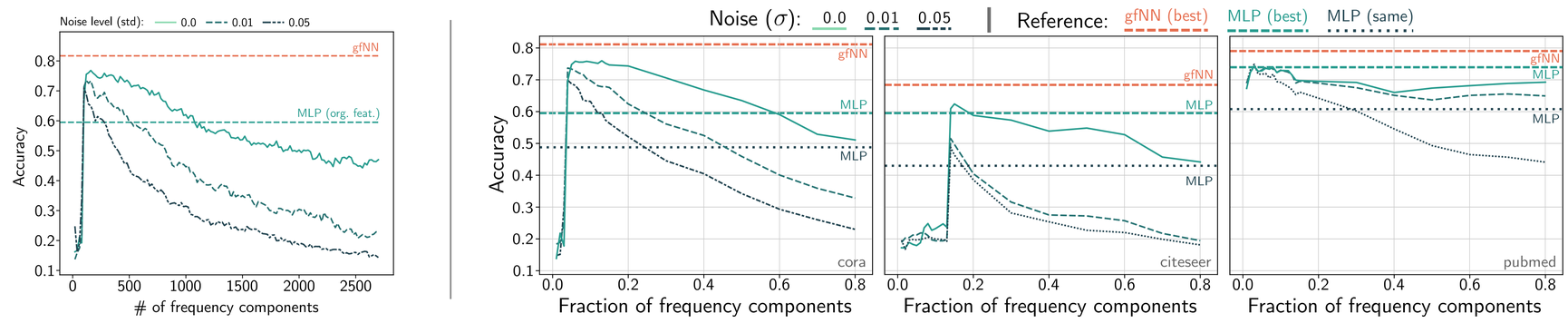
$$X_k = U[:, :k]U^\top[:k, :]X$$

Low-Pass Filtering by matrix

AugNorm
$$\bar{X} = (D+I)^{-1/2}(A+I)(D+I)^{-1/2}X$$
LeftNorm
$$\bar{X} = D^{-1/2}LX$$
Bilateral
$$\bar{X} = (\alpha I - L)^\top(\alpha I - L)X$$

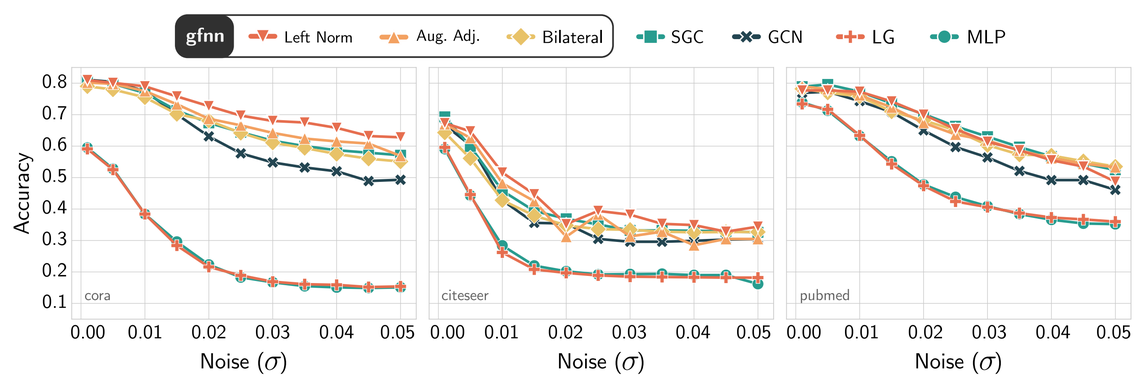
Low-Pass Filtering

Observation: In popular benchmark datasets, using more and more graph frequency components only hurts test accuracies.



Claim: SOTA models simply perform low-pass filtering on signal X and such transformation makes learning easier when X and Y are "low-frequency" with respect to the graph structure.

This view implies models like GCN would be prone to noisy data.



Lemma

Assume input signal X consist of true signals and noise. The true signals have sufficient information for the machine learning tasks.

Then, outcomes of gfNN (also SGC, GCN) are similar to those of the corresponding NNs using true features.

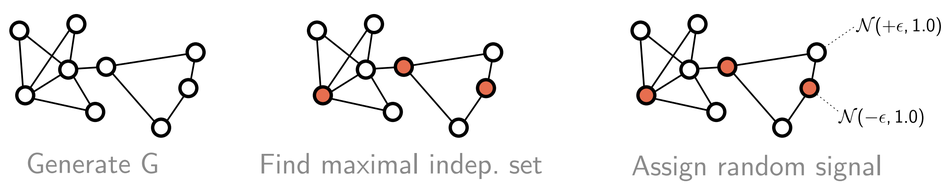
Theorem 7
$$\|h_{\text{MLP}}(\bar{X}|W_1, W_2) - h_{\text{MLP}}(^2X|W_1, W_2)\|_D = \tilde{O}(\sqrt{\epsilon})[\|Z\|_D]\rho(W_1)\rho(W_2)$$

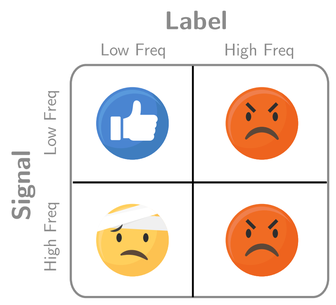
Theorem 8
$$\|h_{\text{MLP}}(\bar{X}|W_1, W_2) - h_{\text{GCN}}(X|W_1, W_2)\|_D \le \tilde{O}(\sqrt[4]{\epsilon})[\|Z\|_D]\rho(W_1)\rho(W_2)$$

Conclusion: Graph Neural Networks (GCN, SGC, etc.) work well when the labels and features are smooth with respect to the graph structure. It is more beneficial to separate feature transformation and learning (gfNN, SGC, PPNP) for scalability and noise tolerrance.

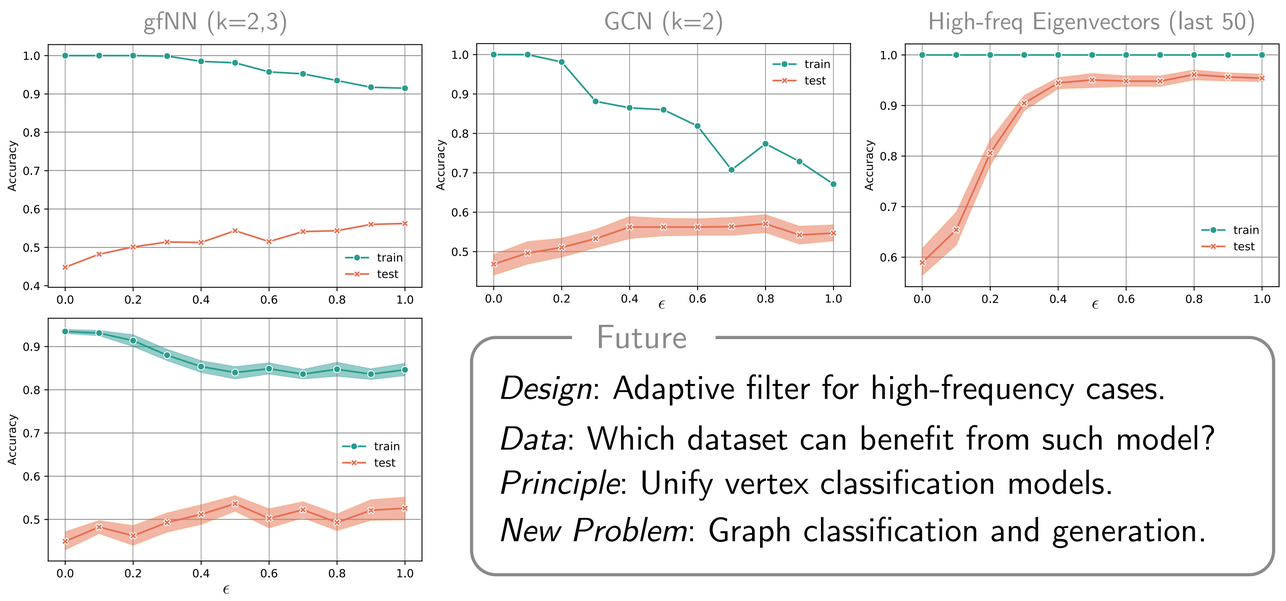
High-Frequency Cases

Observation: State of the art methods do not work well under high-frequency synthetic settings.



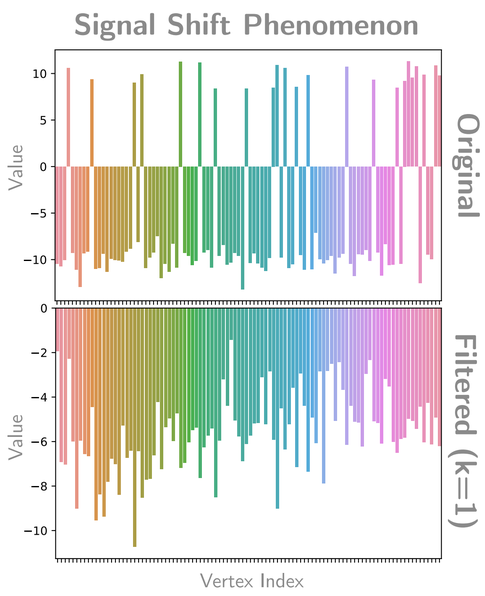


Experiment: Synthetic BA(100,10) graph with maximal independent set labels.



Future

Design: Adaptive filter for high-frequency cases.
Data: Which dataset can benefit from such model?
Principle: Unify vertex classification models.
New Problem: Graph classification and generation.



Our paper: https://arxiv.org/pdf/1905.09550.pdf Github: https://github.com/gear/gfnn