Graph Neural Networks with *Learnable* and *Optimal* Polynomial Bases

Yuhe Guo and **Zhewei Wei** Contact us: zhewei@ruc.edu.cn

ICML International Conference On Machine Learning

RENMIN UNIVERSITY OF CHINA 1937

Notation	Description
$G=(V,E)$	Undirected, connected graph
$\hat{P}$	Symmetric-normalized adjacency matrix of G
$\hat{L}$	Symmetric-normalized Laplacian matrix of G. $\hat{L}=I-\hat{P}$.
$x, z \in \mathbb{R}^N$	Input/Filtered signal on one channel.
$X, Z \in \mathbb{R}^{N\times d}$	Input/Filtered signals on d channels.

Graph Neural Networks
- Spatial GNNs: Graphs as carrier of messages
- Spectral GNNs: Graphs as raw materials to decompose basic *structural frequency components*.
- Polynomial Filters: Polynomial →(*approximate*) filtering function; *Efficiency* makes them dominant in spectral GNNs.
- Other Filters

We are here !

Spectral GNNs:
- Emerges from Graph Signal Processing
- **Node Feature → Multichannel Signals**
- **Laplacian eigenvectors → frequency components**

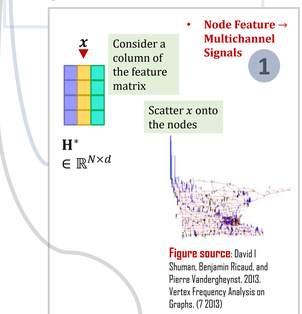
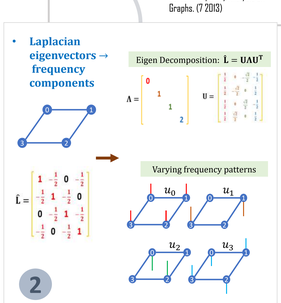
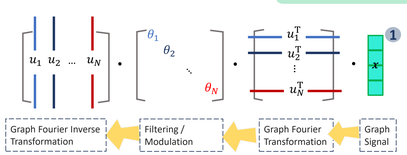


Figure source: David I Shuman, Benjamin Ricaud, and Pierre Vandergheynst. 2013. Vertex Frequency Analysis on Graphs. (7 2013)



BG: Spectral GNNs

$$x \mapsto \mathbf{U}\cdot \mathrm{diag}\{\theta_1,\dots,\theta_N\}\cdot \mathbf{U}^{\mathrm{T}}\cdot x$$



- Back to the nodal / spatial domain
- Frequency Responses on the i-th component adjusted to $\theta_i \mathbf{u}_i^{\mathrm{T}} x$.
- *Laplacian's eigenvectors* used as basic components. 2
- Frequency Responses on the i-th component: $\mathbf{u}_i^{\mathrm{T}} x$.

Shortcoming: *Eigen-decomposition* required; intractable on large graphs.

BG: Polynomial Filtering

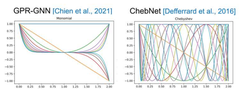
$$x \mapsto \mathbf{U}\cdot \mathrm{diag}\{h(\lambda_1),\dots,h(\lambda_N)\}\cdot \mathbf{U}^{\mathrm{T}}\cdot x$$

$x \mapsto h(\hat{\mathbf{L}})x$ or $b(\tilde{\mathbf{P}})x$ — *Computed Localized !*

$$x \mapsto \Sigma_{k=0}^{K} \alpha_k \; g_k(\tilde{\mathbf{P}})x$$

Arbitrary coefficients — Predetermined polynomial basis

Polynomial bases on the scene !



Motivation

Challenge *this* convention !

- Researchers are trying to applying different basis to polynomial filters.
- Choice of basis impact practical performance.

Examples: **Monomial Basis** (GPRGNN; Chien et al., 2021); **Chebyshev Basis** (Defferrard et al., 2016; He et al., 2021); **Bernstein Basis** (He et al., 2021) ; **Jacobi Basis** (Wang & Zhang, 2022)

- ***Question 1***: Can we ***learn*** orthonormal basis?

Motivation: Existing works choose basis from *famous named polynomial bases*. But the proportion of our knowledge, compare to the vast space of polynomial series, is small. *How to learn from the vast space of all possible orthonormal basis?* 3

- ***Question 2***: Can we ***efficiently*** utilize the ***optimal*** basis?

Motivation: Is there an optimal basis for given (graph, signal)? Wang&Zhang(ICML'22) raised a criterion, **but habitually believe that this optimal basis is intractable**. *Can we utilize this optimal basis efficiently?*

Orthonormal Polynomial Series/Basis 3

- ***Inner product*** of polynomials
  - $\langle f, g\rangle = \int f(x)g(x)w(x)\,\mathrm{d}x$;
  - $w(x)$: weight function; positive.
- ***Orthonormal polynomial series***
  - $\{p_n\}_{n=0}^{\infty}$, $\boldsymbol{p_n}$ is of order $\boldsymbol{n}$;
  - $\langle p_n, p_m\rangle_{\boldsymbol{w}} = \delta_{mn}$.

Example: Chebyshev basis $\{T_n\}_{n=0}^{\infty}$ is ***orthogonal*** w.r.t. $w(x)=\frac{1}{\sqrt{1-x^2}}$, yet not ***orthonormal***.

Orthonormal polynomials ⟵ ω: 4

- Given a ***weight function***, it is easy to construct an ***orthonormal polynomial series*** by the ***Gram-Schmidt process***.

$$p_{k+1} \leftarrow xp_k - \langle xp_k, p_k\rangle_\omega - \langle xp_k, p_{k-1}\rangle_\omega - \langle xp_k, p_{k-2}\rangle_\omega - \cdots \quad \text{// Orthogonalize}$$
$$p_{k+1} \leftarrow p_{k+1}/|p_{k+1}| \quad \text{// Normalize}$$

We are actually familiar with `Three-term Recurrence' ! 5

- A more general three-term recurrence formula is for ***orthogonal*** polynomial series.
- In fact, the property of three-term recurrences for orthogonal polynomials has been used multiple times in the context of current spectral GNNs to reuse $g_k(\hat{P})x$ and $g_{k-1}(\hat{P})x$ for the calculation of $g_{k+1}(\hat{P})x$.

Example: Chebyshev polynomial series, which is orthogonal w.r.t. $w(x)=1/\sqrt{1-x^2}$, satisfies the following three-term recurrence:

$$T_0(x)=1$$
$$T_1(x)=x$$
$$T_{k+1}(x)=2xT_k(x)-T_{k-1}(x)\ (k\geq 2)$$

Model I: FavardGNN

Learn polynomial basis

Learn orthonormal **polynomial series** of order K ⟺ Learn **parameters** $\{\gamma_i\}$ and $\{\sqrt{\beta_i}\}$

- An ***orthonormal polynomial series*** can be thought of as determined by *some* ***weight function*** ω. 4
- **[Three-term Recurrence]** ***Any*** orthonormal polynomial series satisfies the ***three-term recurrence formula***: 5

$$\sqrt{\beta_{k+1}}\ p_{k+1}(x) = (x-\gamma_k)p_k(x) - \sqrt{\beta_k}\ p_{k-1}(x),$$
$$p_{-1}(x):=0,\ p_0(x)=1/\sqrt{\beta_0},$$
$$\gamma_k\in\mathbb{R},\ \sqrt{\beta_k}\in\mathbb{R}^+,\ k\geq 0$$

- **[Favard's Theorem]** Conversly, ***any*** polynomial series of such a recurrence is deemed to be orthonormal *w.r.t.* some weight function!

Algorithm 1: FavardFiltering

Input: Input signals X with d channels; Normalized graph adjacency $\hat{P}$; Truncated polynomial order K
Learnable Parameters : β, γ, α
Output: Filtered Signals Z

$x_{-1} \leftarrow 0$
for $l = 0$ **to** $d-1$ **do**
  $x \leftarrow X_{:,l}$, $x_0 \leftarrow x/\sqrt{\beta_{0,l}}$, $z \leftarrow \alpha_{0,l}x_0$
  for $k = 0$ **to** K **do**
    $x_{k+1} \leftarrow (\hat{P}x_k - \gamma_{k,l}x_k - \sqrt{\beta_{k,l}}x_{k-1})/\sqrt{\beta_{k+1,l}}$
    $z \leftarrow z + \alpha_{k+1,l}x_{k+1}$
  $Z_{:,l} \leftarrow z$
return Z

Algorithm 2: FavardGNN (For Classification)

Input: Raw features X_{raw}; Normalized graph adjacency $\hat{P}$; Truncated polynomial order K
Learnable Parameters : $W_0, b_0, W_1, b_1, \beta, \gamma, \alpha$
Output: Label predictions $\hat{Y}$

$X \leftarrow X_{\text{raw}}W_0 + b_0$
$Z \leftarrow$ FavardFiltering$(X, \hat{P}, K, \beta, \gamma, \alpha)$
$\hat{Y} \leftarrow$ Softmax$(ZW_1 + b_1)$

Model II: OptBasisGNN

Efficiently Solved ***Optimal*** Basis

A Review of previous work

- ***Definition*** of Optimal Basis (Derived by Wang&Zhang, ICML'22)

(Summary of **Definition 4.1 & Proposition 4.2**) For a given graph signal x, polynomial basis $\{g_k\}_{k=0}^K$ is ***optimal in convergence rate*** when $H_{k_1k_2} = x^{\mathrm{T}}g_{k_2}(\hat{P})g_{k_1}(\hat{P})x = \delta_{k_1k_2}$. The **exact weight function** of $\{g_k\}_{k=0}^K$ is also, relying on Rienman sum and Eigen-decomposition of $\hat{P}$.

- The authors believe ***habitually*** that intractable eigen-decomposition is unavoidable. Thus the optimal basis ***cannot be utilized***.

Our Method

Solving optimal polynomial basis ⟺ Solving optimal vector basis

,then the polynomial is solved in an accompanying way.

1. Consider *optimal vectors, instead of polynomials* (Alg. 4)
   - **Condition 1**: $v_{k_1}v_{k_2} = \delta_{k_1k_2}$ (**orthonormality**).
   - **Condition 2**: $\forall k, \exists g_k,\ v_k = g_k(\hat{P})x$. (**Existence of *accompanying polynomial***, which exactly would be the *optimal basis polynomial*)
   - => Filtered signal: $x \mapsto \Sigma_{k=0}^K \alpha_k v_k$
   - => Complexity: $O(K|V|^2 + K|E|)$
2. Use *Proposition 4.4* to reduce complexity to $O(K|V| + K|E|)$. (Plug Alg.5 into Alg.4).
3. The ***nature of this method***:
   - shifting from *defining the optimal polynomial series via weight function* to defining it via *three-term recurrences.*
   - *Discussed in Section 4.3*

Algorithm 4: OptBasisFiltering

1. In the comment, we write the implicitly undergoing process of obtaining the accompanying optimal polynomial basis.
2. Steps 1-3 will be further substituted by Algorithm 5 after the derivative of Proposition 4.4.

Input: Input signals X with d channels; Normalized graph adjacency $\hat{P}$; Order K
Learnable Parameters : α
Output: Filtered signals Z

for $l = 0$ **to** $d-1$ **do**
  $x \leftarrow X_{:,l}$
  $v_0 \leftarrow x/\|x\|$; // $g_0(\mu) = 1/\|x\|$
  $z \leftarrow \alpha_{0,l}v_0$
  for $k = 0$ **to** K **do**
    Step 1: $v_{k+1}^* \leftarrow \hat{P}v_k$; // $g_{k+1}^*(\mu) := \mu g_k(\mu)$
    Step 2: $v_{k+1}^\perp \leftarrow v_{k+1}^* - \sum_{i=0}^k \langle v_{k+1}^*, v_i\rangle v_i$; // $g_{k+1}^\perp(\mu) := g_{k+1}^*(\mu) - \sum_{i=0}^k \langle v_{k+1}^*, v_i\rangle g_i(\mu)$
    Step 3: $v_{k+1} \leftarrow v_{k+1}^\perp/\|v_{k+1}^\perp\|$; // $g_{k+1}(\mu) := g_{k+1}^\perp(\mu)/\|v_{k+1}^\perp\|$
  $z \leftarrow z + \alpha_{k+1,l}v_{k+1}$
  $Z_{:,l} \leftarrow z$
return Z

Proposition 4.4 In Algorithm 4, v_{k+1}^* is only dependent with v_k and v_{k-1}.

Algorithm 5: ObtainNextBasisVector

Input: Normalized graph adjacency $\hat{P}$; **Two** solved basis vectors v_{k-1}, v_k $(k \geq 0)$
Output: v_{k+1}

Step 1: $v_{k+1}^* \leftarrow \hat{P}v_k$
Step 2: $v_{k+1}^\perp \leftarrow v_{k+1}^* - \langle v_{k+1}^*, v_k\rangle v_k - \langle v_{k+1}^*, v_{k-1}\rangle v_{k-1}$
Step 3: $v_{k+1} \leftarrow v_{k+1}^\perp/\|v_{k+1}^\perp\|$
return v_{k+1}

Experiments

Node Classification Tasks

- **Medium sized datasets** (2k ~ 20k Nodes)
  - Geom-GCN datasets (Heterophilious)
  - Citation Networks (Homophilious)
- **LINKX Datasets**
  - Penn94, Genius, Twitch-Gamer (40k ~ 1m Nodes)
  - Pokec, Wiki (1m ~ 2m Nodes)
- **OGBN Datasets**
  - Papers100M (100m Nodes)

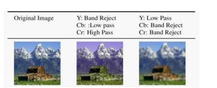
Outstanding — High ranks — Scalability

Table 1. Please check Table 2 & 3 for FavardGNN and OptBasisGNN's performances on other node classification tasks.

Dataset	Chameleon	Squirrel	Actor	Citeseer	Pubmed
$\lvert V\rvert$	2,277	5,201	7,600	3,327	19,717
$\mathcal{H}(G)$	.23	.22	.22	.74	.80
MLP	46.59 ± 1.84	31.01 ± 1.18	40.18 ± 0.55	76.52 ± 0.89	86.14 ± 0.25
GCN	60.81 ± 2.95	45.87 ± 0.8	33.26 ± 1.15	79.85 ± 0.78	86.79 ± 0.31
ChebNet	50.51 ± 1.25	40.81 ± 0.42	37.42 ± 0.58	79.33 ± 0.57	87.82 ± 0.24
ARMA	60.21 ± 1.00	36.27 ± 0.62	37.67 ± 0.54	80.04 ± 0.55	86.93 ± 0.24
APPNP	52.15 ± 1.79	35.71 ± 0.78	39.76 ± 0.49	80.47 ± 0.73	88.13 ± 0.33
GPRGNN	67.49 ± 1.38	50.43 ± 1.89	39.91 ± 0.62	80.13 ± 0.84	88.46 ± 0.31
BernNet	68.53 ± 1.68	51.39 ± 0.92	41.71 ± 1.12	80.08 ± 0.75	88.51 ± 0.39
ChebNetII	71.37 ± 1.01	57.72 ± 0.59	41.75 ± 1.07	80.53 ± 0.79	88.93 ± 0.29
JacobiConv	74.20 ± 1.03	57.38 ± 1.25	41.17 ± 0.64	80.78 ± 0.79	89.62 ± 0.41
FavardGNN	72.32 ± 1.90	63.49 ± 1.47	43.05 ± 0.53	81.89 ± 0.63	90.90 ± 0.27
OptBasisGNN	74.26 ± 0.74	63.62 ± 0.76	42.39 ± 0.52	80.58 ± 0.82	90.30 ± 0.19

Multichannel Regression Task

- **Task**: min $\frac{1}{2}|Z-Y|_F^2$
  - X: Input signals
  - Z: Filtered signals. $Z_{:,l} = \Sigma_{k=0}^K \alpha_k\, g_k(\hat{\mathbf{P}})X_{:,l}$
  - Y: Signals filtered by *true* filters (Synthetic)



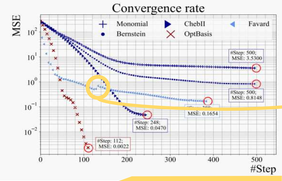
- **Synthetic datasets** (60 samples):
  - **Three channels**: Y, Cb and Cr channels in computer vision
  - **100x100 grid graphs**
  - ***True* filters**: Chosen from Band-Reject / Low-pass / High-pass/

Table 5. Experimental results of the multichannel filtering learning task. MSE loss ± standard errors of the 60 samples achieved by different bases are exhibited.

BASIS	OptBasis	ChebII	Bernstein	Favard	Monomial
MSE	**0.0058**	0.1501	0.4231	0.3175	3.9076
± STDV	± **0.0157**	± 0.2433	± 0.4918	± 0.2840	± 2.9263

OptBasis: Smallest error

Convergence rate



OptBasis: Most rapid convergence !

- **Worth-noting:** **FavardGNN** show large *bumps, indicating bad convergence property.*