Rational Neural Networks for Approximating Graph Convolution Operator on Jump Discontinuities

Zhiqian Chen*, Feng Chen†, Rongjie Lai‡, Xuchao Zhang* and Chang-Tien Lu*
*Computer Science Department, Virginia Tech
Email:{czq,xuczhang,ctlu}@vt.edu
†Department of Computer Science, University at Albany - SUNY
Email:fchen5@albany.edu
‡Department of Mathematics, Rensselaer Polytechnic Institute
Email:lair@rpi.edu

***Abstract*—For node level graph encoding, a recent important state-of-art method is the graph convolutional networks (GCN), which nicely integrate local vertex features and graph topology in the spectral domain. However, current studies suffer from several drawbacks: (1) graph CNNs rely on Chebyshev polynomial approximation which results in oscillatory approximation at jump discontinuities; (2) Increasing the order of Chebyshev polynomial can reduce the oscillations issue, but also incurs unaffordable computational cost; (3) Chebyshev polynomials require degree Ω(poly(1/ϵ)) to approximate a jump signal such as $|x|$, while rational function only needs $\mathcal{O}$(poly log(1/ϵ)) [1], [2]. However, it is non-trivial to apply rational approximation without increasing computational complexity due to the denominator.**

In this paper, the superiority of rational approximation is exploited for graph signal recovering. RatioanlNet is proposed to integrate rational function and neural networks. We show that the rational function of eigenvalues can be rewritten as a function of graph Laplacian, which can avoid multiplication by the eigenvector matrix. Focusing on the analysis of approximation on graph convolution operation, a graph signal regression task is formulated. Under graph signal regression task, its time complexity can be significantly reduced by graph Fourier transform. To overcome the local minimum problem of neural networks model, a relaxed Remez algorithm is utilized to initialize the weight parameters. Convergence rate of RatioanlNet and polynomial based methods on a jump signal is analyzed for a theoretical guarantee. The extensive experimental results demonstrated that our approach could effectively characterize the jump discontinuities, outperforming competing methods by a substantial margin on both synthetic and real-world graphs.

I. Introduction

Effective information analysis generally boils down to the geometry of the data represented by a graph. Typical applications include social networks [3], transportation networks [4], spread of epidemic disease [5], brain's neuronal networks [6], gene data on biological regulatory networks [7], telecommunication networks [8], knowledge graph [9], which are lying on non-Euclidean graph domain. To describe the geometric structures, graph matrices such as adjacency matrix or graph Laplacian can be employed to reveal latent patterns.

In recent years, many problems are being revisited with deep learning tools. Convolutional neural networks(ConvNets) emerging in recent years are at the heart of deep learning, and the most prominent strain of neural networks in research. ConvNets have revolutionized computer vision [10], natural language processing [11], computer audition [12], reinforcement learning [13], [14], and many other areas. However, ConvNets are designed for grid data such as image, which belongs to the Euclidean domain. Graph data is non-Euclidean which makes it difficult to employ typical ConvNets. To bridge the gap, Bruna et al. [15] [16] generalized spectral convolutional operation which requires expensive steps of spectral decomposition and matrix multiplication. Hammond et al. [17] first introduced truncated Chebyshev polynomial for estimating wavelet in graph signal processing. Based on this polynomial approximation, Defferrard et al. [18] designed ChebNet which contains a novel neural network layer for the convolution operator in the spectral domain. Kipf and Welling [19] simplified ChebNet by assuming the maximum of eigenvalues is 2 and fixing the order to 1, which boosts both effectiveness and efficiency. Li et al. [20] found that this simplified ChebNet is an application of Laplacian smoothing, which implies that current studies are only effective on the smooth signal.

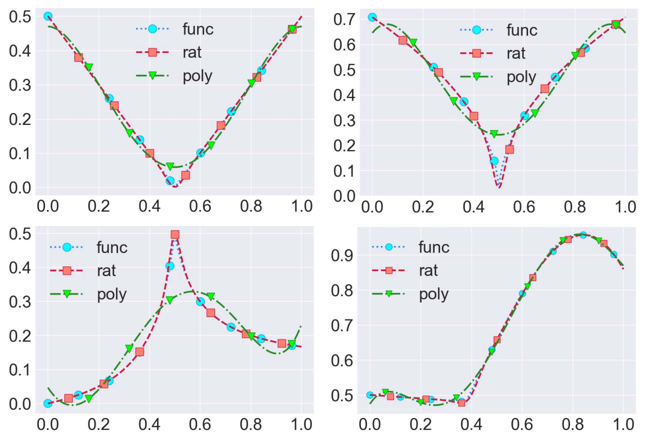

Fig. 1: Rational(rat) and polynomial(poly) approximation for several jump functions(func). Top left: $\sqrt{|x-0.5|}$; top right: $|x-0.5|$; bottom left: $\frac{x}{10|x-0.5|+1}$; bottom right: $max(0.5, sin(x+x^2)) - \frac{x}{20}$

Current studies on graph ConvNet heavily rely on polynomial approximation, which makes it difficult to estimate jump signals. Fig. 1 shows the behaviors of polynomial and rational function on jump discontinuity: rational approximation fits the functions considerably better than polynomials. It is widely recognized that **(1)** polynomial approximation suffers from Gibbs phenomenon, which means polynomial function oscillate and overshoot near discontinuities [21]; **(2)** Applying a higher order of polynomials could dramatically reduce the oscillation, but also incurs an expensive computational cost. **(3)** Polynomials require degree $\Omega(\text{poly}(1/\epsilon))$ to approximate functions near singularities and on an unbounded domain, while rational functions only need $\mathcal{O}(\text{poly}\ \log(1/\epsilon))$ to achieve ϵ-close [1], [2]. However, it is non-trivial to apply rational approximation. Polynomial-based method can easily transfer the function on eigenvalues to the same function on graph Laplacian so that matrix multiplication by eigenvector can be avoided. It is not easy for rational approximation to do so due to the additional denominator.

In this paper, the advantage of the rational function in approximation is transferred to spectral graph domain. Specifically, we propose a rational function based neural networks(RationalNet), which can avoid matrix multiplication by eigenvectors. To alleviate the local minimum problem of the neural network, a relaxed Remez algorithm is employed for parameter initialization. Our theoretical analysis shows that rational functions converge much faster than polynomials on jump signal. In a nutshell, the key innovations are:

- **Propose a neural network model based on rational function for recovering jump discontinuities**: To estimate the jump signal, our proposed method integrates rational approximation and spectral graph operation to avoid matrix multiplication by eigenvectors. For graph signal regression task, expensive matrix inversion can be circumvented by graph Fourier transform.
- **Develop an efficient algorithm for model parameters optimization**: Remez algorithm is theoretically optimal, but it is often not practical especially when approximating discrete signal. To alleviate this issue, the stopping rules of Remez algorithm are relaxed to initialize the neural networks parameters.
- **Provide theoretical analysis for the proposed method on jump signal**: For understanding the behaviors of polynomial and rational function on jump discontinuities, a uniform representation is proposed to analyze convergence rate regarding the order number theoretically.
- **Conducting extensive experiments for performance evaluations**[1]: The proposed method was evaluated on synthetic and real-world data. Experimental results demonstrate that the proposed approach runs efficiently and consistently outperforms the best of the existing methods.

The rest of the paper is organized as follows. Section II reviews existing work in this area. Necessary preliminary is presented in section III. Section IV elaborates a rational function based model for graph signal regression task. Section V presents algorithm description and theoretical analysis. In Section VI, experiments on synthetic and real-world data are analyzed. This paper concludes by summarizing the study's important findings in Section VII.

[1]https://github.com/aquastar/RationalGraphNet

II. Related Work

The proposed method draws inspiration from the field of approximation theory, spectral graph theory and recent works using neural networks. In what follows, we provide a brief overview of related work in these fields.

A. Approximation theory

In mathematics, approximation theory is concerned with how functions can best be approximated with simpler functions, and with quantitatively characterizing the errors introduced thereby. One problem of particular interest is that of approximating a function in a computer mathematical library, using operations that can be performed on the computer or calculator, such that the result is as close to the actual function as possible. This is typically done with polynomial or rational approximations. Polynomials are familiar and comfortable, but rational functions seem complex and specialized, and rational functions are more powerful than polynomials at approximating functions near singularities and on unbounded domains. Basic properties of rational function are described in books of complex analysis [21]–[31].

B. Spectral graph theory

Spectral graph theory is the study of the properties of a graph in relationship to the characteristic polynomial, eigenvalues, and eigenvectors of matrices associated with the graph [32]–[34]. Many graphs and geometric convolution methods have been proposed recently. The spectral convolution methods ([15], [18], [19], [35]) are the mainstream algorithm developed as the graph convolution methods. Because their theory is based on the graph Fourier analysis ([36], [37]). The polynomial approximation is firstly proposed by [17]. Inspired by this, graph convolutional neural networks (GCNNs) ([18]) is a successful attempt at generalizing the powerful convolutional neural networks (CNNs) in dealing with Euclidean data to modeling graph-structured data. Kipf and Welling proposed a simplified type of GCNNs [19], called graph convolutional networks (GCNs). The GCN model naturally integrates the connectivity patterns and feature attributes of graph-structured data and outperforms many state-of-the-art methods significantly. Li et al. [20] found that GCN is actual an application of Laplacian smoothing, which is inconsistent with GCN's motivation. In sum, this thread of work calculates the average of vertexes within Nth-order neighbors.

In this paper, we focus on the effectiveness of the approximation technique on graph signal. Under graph signal regression task, the superiority of rational function beyond polynomial function is analyzed, and a rational function based neural network is proposed.

III. Preliminaries

We focus processing graph signals defined on undirected graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where $\mathcal{V}$ is a set of n vertexes, $\mathcal{E}$ represents edges and $\mathcal{W} = [w_{ij}] \in \{0,1\}^{n\times n}$ is an unweighted adjacency matrix. A signal $x : \mathcal{V} \to \mathbb{R}$ defined on the nodes may be regarded as a vector $x \in \mathbb{R}^n$. Combinatorial graph Laplacian [32] is defined as $\mathbf{L} = D - \mathcal{W} \in \mathbb{R}^{n\times n}$ where D is degree matrix.

As $\mathbf{L}$ is a real symmetric positive semidefinite matrix, it has a complete set of orthonormal eigenvectors and their associated ordered real nonnegative eigenvalues identified as the frequencies of the graph. The Laplacian is diagonalized by the Fourier basis $\mathbf{U}^\intercal$: $\mathbf{L} = \mathbf{U}\,\Lambda\,\mathbf{U}^\intercal$ where Λ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues, i.e., $\Lambda_{ii} = \lambda_i$. The graph Fourier transform of a signal $x \in \mathbb{R}^n$ is defined as $\hat{x} = \mathbf{U}^\intercal x \in \mathbb{R}^n$ and its inverse as $x = \mathbf{U}\,\hat{x}$ [36]–[38]. To enable the formulation of fundamental operations such as filtering in the vertex domain, the convolution operator on graph is defined in the Fourier domain such that $f_1 * f_2 = \mathbf{U}\,[(\mathbf{U}^\intercal f_1) \odot (\mathbf{U}^\intercal f_2)]$, where $\odot$ is the element-wise product, and f_1/f_2 are two signals defined on vertex domain. It follows that a vertex signal $f_2 = x$ is filtered by spectral signal $\hat{f_1} = \mathbf{U}^\intercal f_1 = \mathbf{g}$ as:

$$\mathbf{g} * x = \mathbf{U}\,[\mathbf{g}(\mathbf{\Lambda}) \odot (\mathbf{U}^\intercal f_2)] = \mathbf{U}\,\mathbf{g}(\mathbf{\Lambda})\,\mathbf{U}^\intercal x.$$

Note that a real symmetric matrix $\mathbf{L}$ can be decomposed as $\mathbf{L} = \mathbf{U}\,\mathbf{\Lambda}\,\mathbf{U}^{-1} = \mathbf{U}\,\mathbf{\Lambda}\,\mathbf{U}^\intercal$ since $\mathbf{U}^{-1} = \mathbf{U}^\intercal$. D. K. Hammond et al. and Defferrard et al. [17], [18] apply polynomial approximation on spectral filter $\mathbf{g}$ so that:

$$\begin{aligned}
&\mathbf{g} * x = \mathbf{U}\,\mathbf{g}(\mathbf{\Lambda})\,\mathbf{U}^\intercal x \\
&\approx \mathbf{U}\sum_k \theta_k T_k(\tilde{\mathbf{\Lambda}})\,\mathbf{U}^\intercal x \qquad (\tilde{\mathbf{\Lambda}} = \frac{2}{\lambda_{max}}\mathbf{\Lambda} - \mathbf{I_N}) \\
&= \sum_k \theta_k T_k(\tilde{\mathbf{L}})x \qquad (\mathbf{U}\,\mathbf{\Lambda}^k\,\mathbf{U}^\intercal = (\mathbf{U}\,\mathbf{\Lambda}\,\mathbf{U}^\intercal)^k)
\end{aligned}$$

Kipf and Welling [19] simplifies it by:

$$\begin{aligned}
&\mathbf{g} * x \\
&\approx \theta_0\,\mathbf{I_N}\,x + \theta_1\tilde{\mathbf{L}}x && (\text{expand to 1st order}) \\
&= \theta_0\,\mathbf{I_N}\,x + \theta_1(\frac{2}{\lambda_{max}}\,\mathbf{L} - \mathbf{I_N}))x && (\tilde{\mathbf{L}} = \tfrac{2}{\lambda_{max}}\,\mathbf{L} - \mathbf{I_N})) \\
&= \theta_0\,\mathbf{I_N}\,x + \theta_1(\mathbf{L} - \mathbf{I_N}))x && (\lambda_{max} = 2) \\
&= \theta_0\,\mathbf{I_N}\,x - \theta_1\,\mathbf{D}^{-\frac{1}{2}}\,\mathbf{A}\,\mathbf{D}^{-\frac{1}{2}}\,x && (\mathbf{L} = \mathbf{I_N} - \mathbf{D}^{-\frac{1}{2}}\,\mathbf{A}\,\mathbf{D}^{-\frac{1}{2}}) \\
&= \theta_0(\mathbf{I_N} + \mathbf{D}^{-\frac{1}{2}}\,\mathbf{A}\,\mathbf{D}^{-\frac{1}{2}})x && (\theta_0 = -\theta_1) \\
&= \theta_0(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}})x && (\text{renormalization:} \tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I_N}, \\
& && \tilde{\mathbf{D}}_{ii} = \textstyle\sum_j \mathbf{A}_{ij}),
\end{aligned}$$

rewrite the above GCN in matrix form: $\mathrm{g}_\theta * X \approx (\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}})X\Theta$, which leads to *symmetric normalized Laplacian* with raw feature. GCN has been analyzed GCN in [20] using smoothing Laplacian [39]: $y = (1-\gamma)x_i + \gamma\sum_j \frac{\tilde{a}_{ij}}{d_i}x_j = x_i - \gamma(x_i - \sum_j \frac{\tilde{a}_{ij}}{d_i}x_j)$,where γ is a weight parameter between the current vertex x_i and the features of its neighbors x_j, d_i is degree of x_i, and y is the smoothed Laplacian. This smoothing Laplacian has a matrix form

$$\begin{aligned}
Y &= x - \gamma\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{L}}x \\
&= (\mathbf{I_N} - \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{L}})x && (\gamma = 1) \\
&= (\mathbf{I_N} - \tilde{\mathbf{D}}^{-1}(\tilde{\mathbf{D}} - \tilde{\mathbf{A}}))x && (\tilde{\mathbf{L}} = \tilde{\mathbf{D}} - \tilde{\mathbf{A}}) \\
&= \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}x.
\end{aligned}$$

The above formula is *random walk normalized Laplacian* as a counterpart of *symmetric normalized Laplacian*. Therefore, GCN is nothing but a first-order Laplacian smoothing which averages neighbors of each vertex.

IV. Model description

This section formally defines the task of graph signal recovering and then describes our proposed RationalNet which aims to characterize the jump signal in the spectral domain.

A. Problem Setting

All the related works integrate graph convolution estimator and fully-connected neural layers for a classification task. This classification can be summarized as:

$$Y = f(\mathcal{G}, x)\Theta, \tag{1}$$

where Θ indicates the parameters of normal neural network layers connecting the output of f and the label Y, such as fully-connected layers and softmax layers for classification. Moreover, f is a neural network layer implemented by approximation techniques. However, whether the success is due to the neural networks(Θ) or the convolution approximation method(f) remains unknown. To focus on the analysis of approximation on f, a graph signal regression task is proposed to evaluate the performance of the convolution approximators f. Regression task directly compares the label and the output of f, removing the distraction of Θ.

Given a graph $\mathcal{G}$, raw feature x, and training signal on the part of vertexes, Y_{train}, our goal is to recover signal values, Y_{test}, on test nodes. Formally, we want to find a $f(\cdot)$ so that:

$$Y = f(\mathcal{G}, x).$$

If the raw features are good enough for the regression task, whether the effectiveness is due to f or x is difficult to verify. Therefore, one reasonable option for x is the uniform signal in spectral domain. Specifically, $x = \sum_i \mathbf{U}_i$ and $\hat{x} = \mathbf{U}^\intercal x = \mathbb{1} = \{1, 1, ..., 1\}$, which means that x represents eigenbasis of graph structure in spectral domain. Each entry of vector $\hat{x}_i$ indicates one eigenvector in the spectral domain. The physical meaning of the convolution operation is how to select eigenbasis in spectral domain to match the graph signal Y. Representing $\mathcal{G}$ with graph Laplacian, the regress task can be rewritten as:

$$Y = f(\mathbf{L}, \mathbf{U}) = \mathbf{U}\,\mathrm{g}_\theta(\mathbf{\Lambda})\,\mathbf{U}^\intercal \sum_i \mathbf{U}, \tag{2}$$

where g_θ is the spectral filter to approximate.

B. RationalNet

RationalNet is a neural network that has a rational function as the kernel for graph convolution operator. As shown in Fig. 2, graph connection information, i.e., graph Laplacian $\mathbf{L}$, is converted into spectral domain by inverse Fourier transform $\mathbf{U}^\intercal$. The same transform is operated on features X. Then the element-wise product of them in spectral domain is convolution or spectral filtering. After that, the convolution signal is approximated by a rational function whose parameters are learned during training. The new features of nodes, Z, are obtained by Fourier transform $\mathbf{U}$. Finally, the residues between the new features and ground truth are calculated for the backpropagation algorithm. Similar to polynomial

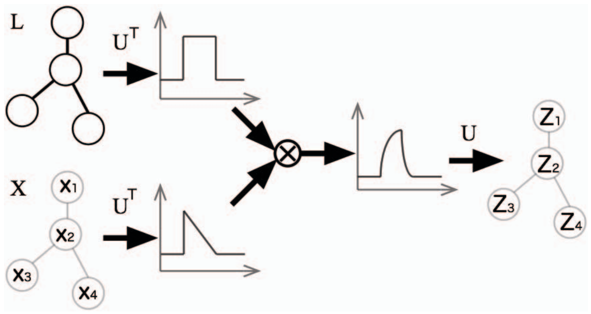

Fig. 2: Structure of RationalNet.

approximation on graph domain such as ChebNet [18] or GCN [19], RationalNet approximates the spectral filter by a widely used type of rational function, i.e., Padé approximator, which is defined as:

$$R(x) = \frac{\sum_{i=0}^{m} \psi_i x^i}{\sum_{j=0}^{n} \phi_j x^j}, \phi_0 = 1, \phi_j, \psi_i \in \mathbb{R}. \tag{3}$$

Applying to graph convolution operator, we have:

$$\begin{aligned}
&\mathrm{g}_\theta * x = \mathbf{U}\, \mathrm{g}_\theta\, \mathbf{U}^\intercal x && \text{(convolution theorem)}\\
&\approx \mathbf{U} \frac{\sum_{i=0}^{m} \psi_i \tilde{\mathbf{\Lambda}}^i}{1 + \sum_{j=1}^{n} \phi_j \tilde{\mathbf{\Lambda}}^j} \mathbf{U}^\intercal x && (\tilde{\mathbf{\Lambda}} = \frac{\mathbf{\Lambda}}{\lambda_{max}})\\
&= \mathbf{U} \frac{\mathbf{P}(\mathbf{\Lambda})}{\mathbf{Q}(\mathbf{\Lambda})} \mathbf{U}^\intercal x, && \text{(define P and Q)}
\end{aligned}$$

where $\mathbf{\Lambda}$ represents a diagonal matrix whose entries are eigenvalues, $\mathrm{g}_\theta = R$, and $\theta = \{\psi, \phi\}$. The division $\frac{P}{Q}$ is element-wise. The inverse of matrix Q(x) is equivalent to applying reciprocal operation on its diagonal entries, so the equation can be rewritten as:

$$\mathbf{U}\, \mathbf{P}(\mathbf{\Lambda})\, \mathbf{Q}(\mathbf{\Lambda})^{-1}\, \mathbf{U}^\intercal x.$$

Applying matrix rules, it's easy to have:

$$\begin{aligned}
&= \mathbf{U}\, \mathbf{P}(\mathbf{\Lambda})\, \mathbf{U}^\intercal\, \mathbf{U}\, \mathbf{Q}(\mathbf{\Lambda})^{-1}\, \mathbf{U}^\intercal x && (\mathbf{U}^\intercal = \mathbf{U}^{-1}, \mathbf{U}^\intercal \mathbf{U} = \mathbf{I_N})\\
&= \big[\mathbf{U}\, \mathbf{P}(\mathbf{\Lambda})\, \mathbf{U}^\intercal\big]\big[\mathbf{U}\, \mathbf{Q}(\mathbf{\Lambda})^{-1}\, \mathbf{U}^\intercal\big] x\\
&= \big[\mathbf{P}(\mathbf{U}\, \mathbf{\Lambda}\, \mathbf{U}^\intercal)\big]\big[\mathbf{U}\, \mathbf{Q}(\mathbf{\Lambda})^{-1}\, \mathbf{U}^\intercal\big] x && (\mathbf{U}\, \mathbf{\Lambda}^k\, \mathbf{U}^\intercal = (\mathbf{U}\, \mathbf{\Lambda}\, \mathbf{U}^\intercal)^k)\\
&= \big[\mathbf{P}(\mathbf{U}\, \mathbf{\Lambda}\, \mathbf{U}^\intercal)\big]\big[(\mathbf{Q}(\mathbf{\Lambda})\, \mathbf{U}^{-1})^{-1}\, \mathbf{U}^\intercal\big] x && (A^{-1}B^{-1} = (BA)^{-1})\\
&= \big[\mathbf{P}(\mathbf{U}\, \mathbf{\Lambda}\, \mathbf{U}^\intercal)\big]\big[(\mathbf{U}\, \mathbf{Q}(\mathbf{\Lambda})\, \mathbf{U}^{-1})^{-1}\big] x && (\mathbf{U}^\intercal = \mathbf{U}^{-1})\\
&= \big[\mathbf{P}(\mathbf{U}\, \mathbf{\Lambda}\, \mathbf{U}^\intercal)\big]\big[(\mathbf{U}\, \mathbf{Q}(\mathbf{\Lambda})\, \mathbf{U}^\intercal)^{-1}\big] x && (\mathbf{U}^\intercal = \mathbf{U}^{-1})\\
&= \big[\mathbf{P}(\mathbf{U}\, \mathbf{\Lambda}\, \mathbf{U}^\intercal)\big]\big[(\mathbf{Q}(\mathbf{U}\, \mathbf{\Lambda}\, \mathbf{U}^\intercal))^{-1}\big] x && (\mathbf{U}\, \mathbf{\Lambda}^k\, \mathbf{U}^\intercal = (\mathbf{U}\, \mathbf{\Lambda}\, \mathbf{U}^\intercal)^k)
\end{aligned}$$

Since $\mathbf{U}\, \mathbf{\Lambda}^k\, \mathbf{U}^\intercal = (\mathbf{U}\, \mathbf{\Lambda}\, \mathbf{U}^\intercal)^k$, we can rewrite the equation above as:

$$\mathrm{g}_\theta * x = \mathbf{P}(\mathbf{L})\, \mathbf{Q}(\mathbf{L})^{-1} x, \tag{4}$$

where $\mathbf{P}(x) = \sum_{i=0}^{m} \psi_i x^i$ and $\mathbf{Q}(x) = \sum_{j=0}^{n} \phi_j x^j$. Note that $\mathbf{P}(\mathbf{L})\, \mathbf{Q}(\mathbf{L})^{-1} x = \mathbf{Q}(\mathbf{L})^{-1}\, \mathbf{P}(\mathbf{L}) x$ in our case. Based on polynomial approximation, RationalNet only adds a inverse polynomial function $\mathbf{Q}(\mathbf{L})^{-1}$. Therefore, polynomial approximation (GCN/ChebNet) on graph is a special case of RationalNet when $\mathbf{Q}(\mathbf{L})^{-1} = \mathbf{I}$.

Computing inverse of a matrix is still of high complexity $\mathcal{O}(n^3)$ (GaussJordan elimination method) in Eq. 4, especially for large matrix. This can be avoided by transferring vertex graph signal and raw features into the spectral domain. Therefore, the Eq. 12 can be rewritten as:

$$\hat{Y} = \frac{\mathbf{P}(\mathbf{\Lambda})}{\mathbf{Q}(\mathbf{\Lambda})} \hat{x}, \tag{5}$$

where $\hat{Y} = \mathbf{U}^\intercal Y$ is the graph Fourier transform of graph signal, and $\hat{x} = \mathbf{U}^\intercal x$ is the graph Fourier transform of raw feature. By this step, we only need compute reciprocal of eigenvalues, rather than computing matrix multiplication and inversion at each update. Eq. 2 can be obtained via left multiplying both sides of Eq. 2 by transpose of eigenvectors. Note that Eq. 5 is applicable when there is no other layers between the output Y and the convolution operation. In contrast, Eq. 4 can be used not only in regression task like Eq. 2, but also in classification where there exist additional neural networks as described in Eq. 1. RationalNet has complexity $\mathcal{O}(|\mathcal{E}|)$ for Eq. 5 and $\mathcal{O}(|\mathcal{E}|^3)$ for Eq. 4.

C. Relaxed Remez Algorithm for initialization

RationalNet is powerful at approximating a function. However, it is often stuck in a local optimum due to the neural network optimization, which makes rational function not always better than the polynomial approximation. Remez exchange algorithm [31] is an iterative algorithm used to find simple approximations to functions. Nevertheless, the drawback of this minimax algorithm is that the order for optimum is unknown and the stopping condition is not often practical. To improve RationalNet's initialization, a relaxed Remez algorithm is proposed.

As Theorem **24.1** (equioscillation characterization of best approximants, [21]) states: Given the order of numerator(m) and denominator(n), and a real function f that is continuous in [p, q], there exists a unique best rational approximation R^* defined as Eq. 3. This means that the R^* optimizes the minimax error:

$$R^* = \arg\min_{R} \max_{x \in [p,q]} |f - R|. \tag{6}$$

A rational function R is equal to R^* iff $f - R$ equioscillates between at least m+n+2 extreme points, or say the error function attains the absolute maximum value with alternating sign:

$$f(x_d) - R(x_d) = (-1)^d E, d \in [0, m+n+1], \tag{7}$$

where d indicates the index of data point, and E represents the max of residues: $E = max_{x_d} |f(x_d) - R(x_d)|$. For rational function approximation, there is some nonlinearity because there will be a product of E with ϕ_j in the equations. Hence, these equations need to be solved iteratively. The iteration formula can be defined by linearizing the equations:

$$\sum_{i=0}^{m} \psi_i x_d^i - \left[f\left(x_d\right) - (-1)^d E_r \right] \sum_{j=1}^{k} \phi_j x_d^j = f\left(x_d\right) - (-1)^d E_{r+1}, \tag{8}$$

where r indicate the iteration index. Eq. 8 is obtained by neglecting nonlinear terms of the form $(E_r - E_{r+1})\phi_j x_d^j$ in Eq. 7. This procedure can converge in a reasonable time ([40]). Expanding Eq. 8 for all sampled data points $x_0, x_1, ..., x_d$, it can be rewritten as:

$$\begin{bmatrix} x_0^0 & \cdots & x_0^m & (E_r - y_0)x_0^1 & \cdots & (E_r - y_0)x_0^n & (-1)^0 \\ x_1^0 & \cdots & x_1^m & (E_r - y_0)x_1^1 & \cdots & (E_r - y_0)x_1^n & (-1)^1 \\ x_2^0 & \cdots & x_2^m & (E_r - y_0)x_2^1 & \cdots & (E_r - y_0)x_2^n & (-1)^2 \\ \cdots \\ x_K^0 & \cdots & x_K^m & (E_r - y_0)x_K^1 & \cdots & (E_r - y_0)x_K^n & (-1)^K \end{bmatrix} \begin{bmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ .. \\ \psi_m \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ .. \\ \phi_n \\ E_{r+1} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \cdots \\ y_K \end{bmatrix} \tag{9}$$

where K=m+n+1. Starting from an assumed initial guess $E_{r=0}$ this set of linear equations can be solved for the unknown ψ_i, ϕ_j and E_{r+1}, when two successive values of E_r are in satisfactory agreement such as $|E_{r+1} - E_r|$ is less than 1e-6. Constructing a rational function with new coefficients, Remez computes the error residuals. If absolute value of the residuals δ are not great than $|E|$, the optimal coefficients are obtained. Otherwise, Remez calculates the roots of rational function and constructs a new set of control points by collecting the extremes in each interval of roots, and repeat the computation of Eq. 8 until residuals δ are not great than $|E|$. However, this stopping rule is not often satisfied, which makes the algorithm stuck in dead loop. Therefore, we add an iteration limit for avoid dead loop. The relaxed Remez algorithm could be summarized as follows:

1. **Prepare training data**
   - Specify the degree of interpolating rational function.
   - Pick $m + n + 2$ points from the data points $X = \{x_0, x_1, ..., x_{m+n+1}\}$ with equal interval. Under this discrete setting, the distances between any neighbors are considered equal if the data distribution are dense
2. **Optimization by equioscillation constraint**
   - Solve coefficients and E by Eq. 9
   - Form a new rational function R with new coefficients
   - Calculate residual errors
   - Repeat until E converges or $|E_{r+1} - E_r|$ is less than 1e-6
3. **Check stopping rule**
   - Calculate residual errors
   - Stops if the absolute value of any residual is not numerically greater than $|E|$.
   - Otherwise, find the n+m+1 roots of the rational function and find the points at which the error function attains its extreme value. Rerun the algorithm with this new set of training data from the second step.

We have considered an algorithm for obtaining minimax approximation when the function could be evaluated at any point inside the interval. In our case, the function is known only at a set of discrete points, since eigenvalues are not continuous. However, this problem is no essentially different form the continuous case if the set of points is rather dense in the target interval [p, q]. We simply assume that eigenvalue samples are dense enough, since we often normalized eigenvalues into the range [0,1], several hundreds of points are thereby sufficient. For example, our smallest size of the synthetic graph consists of 500 nodes, so there are 500 eigenvalues distributed in [0, 1], which should be enough for approximation.

If the degree of rational function is large, then the system of equations could be ill-conditioned. Sometime, the linear system of equations 9 is singular, which make the solution vector(ψ_i, ϕ_j, E_{r+1}) under-determined. We traverse all possible m/n pairs given the maximum of m and n. The relaxed algorithm discards any m/n if singular matrix error occurs.

We found that the residuals δ are not smaller than $|E|$ for some m/n pairs, and the algorithm continues to output the same values. In such case, the algorithm stops if the maximum and minimum residuals ($\delta_{min,max}$) converge or they satisfy $\delta_{0,1,...,m+n+1} < |E|$.

V. Algorithm and Theoretical analysis

This section elaborates algorithm details and analyzes its convergence rate on jump discontinuity.

A. Algorithm description and complexity analysis

The Algorithm 1 first calculate graph Laplacian(line 1) and spectral decomposition(line 3), and convert vertex signal into spectral domain by inverse Fourier transform(line 2). From given m, n, algorithm 1 traverse all possible m/n pairs (line 4). Picking up $m+n+2$ points with equal intervals, the optimal error is calculated (line 10). After convergence, optimal m/n and ψ_i/ϕ_i are determined (line 11). Then algorithm calculates the residuals(line 13). If the stopping rule is not satised, decrease the order of denominator or numerator in turns and repeat the same process, otherwise, output the parameters of rational function. With optimal parameters, graph convolution operation is calculated by rational approximation (line 19). Then we conduct typical neural networks optimization.

Solving a system of linear equations(line 6-15) has a complexity of at most $\mathcal{O}((n+m+2)^3)$ and at least $\mathcal{O}((n+m+2)^2)$. However, $m+n+2$ is often small in practice such as 12 in our experiments, since large $m+n+2$ will exponentially increase the complexity and violate its original intention. Therefore, the actual complexity is $\mathcal{O}(1)$. Since the initialization identifies a configuration near the optimum, training neural network(line 17-23) dose not take too much time.

B. Theoretical analysis

This section first represents jump discontinuity using a function(Eq. 10). Then convergence rate of rational function on jump discontinuity is analyzed(Theorem V.1). With the help of Lemma V.2, we prove Theorem V.1. Similarly, convergence rate of polynomial function(Theorem V.3) is also provided.

We found that $f_{\sigma=1} = a|x| + bx$ and $f_{\sigma=2} = a\frac{|x|}{x} + bx$ can characterize single jump discontinuity. For example, when a=b=1/2 and σ=0, $f_{1,2}$ is ReLU function. It is $sign(x)$ when a=1 and b=1, and σ=1. Thus, $f_{1,2}$ rotates or change the angle

```
Algorithm 1: RationalNet
   Input: a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$,
   rational function order: m,
   graph signal on nodes: Y(i), i $\in 1, 2, ..., |\mathcal{V}|$
   Output: a rational function with parameters: $\psi_i$ and $\phi_i$
1  compute graph Laplacian: $\mathbf{L} = A - D$
2  compute spectral signal by graph Fourier transform:
     $\hat{Y} = \mathbf{U}^\intercal Y$
3  perform eigen decomposition: $\mathbf{L} = \mathbf{U}\,\Lambda\,\mathbf{U}^\intercal$
4  n ← m
5  // initialize parameters by a relexed Remez
6  repeat
7      Pick m + n + 2 points $x_0, x_1, ..., x_{m+n+1}$ from full
        data $X$ with equal interval
8      r = 0, $E_r$ = 0
9      repeat
10         solve $\psi_{0\sim m}, \phi_{1\sim n}, E_{r+1}$                ▷ Eq. 8 or 9
11     until $E_{r+1} - E_r$ convergence;
12     form a Padé rational function $R_{\psi,\phi}$ with $\psi_{0\sim m}, \phi_{1\sim n}$
13     compute residues $\delta_d = |\hat{Y}(d) - R(x_d)|$
14     m ← m-1 or n ← n-1 in turns.
15 until $\delta$ convergence or $\delta_{min,max} < |E|$;
16 // initialize a rational function with the above coefficients
17 repeat
18     form a Padé rational function $R_{\psi,\phi}$ with $\psi_{0\sim m}, \phi_{1\sim n}$
        obtained in the above repeat loop
19     $R(\mathbf{L})x = \mathbf{P}(\mathbf{L})\,\mathbf{Q}(\mathbf{L})^{-1}x$                ▷ Eq. 4 or 5
20     $\theta = \{\psi_i, \phi_j\}$
21     compute the mean error function
        $\mathcal{L} = \mathbf{MSE}(R(x) - Y)$
22     compute derivatives to update parameters:
        $\theta \leftarrow \theta + \beta\nabla_\theta\mathcal{L}$, where $\beta$ is learning rate
23 until MSE converges;
```

between two lines at jump discontinuity based on $|x|$ and x. These two functions can be rewritten in an uniform formula:

$$f_{1,2} = a\frac{|x|}{x^{\sigma\in\{0,1\}}} + bx \tag{10}$$

where $a, b \in \mathbb{R}$.

Theorem V.1 (convergence rate of rational approximation on jump discontinuity)**.** *Given n≥5 and b≥1, there exist a rational function $R_n(x)$ of degree n that satisfies*

$$\sup_{x\in[-c,c]} |f_{1,2} - R_n(x)| \leq Ce^{-\sqrt{n}}.$$

In our proof of Theorem V.1, for $n \in \mathbb{N}$, we follow [41] and define the Newman polynomial: $\mathbf{N_n}(x) := \prod_{i=1}^{n-1}(x + \alpha_n^i)$, where $\alpha_n := e^{-1/\sqrt{n}}$. To approximate jump discontinuity, define $A_n(x)$ as Newman approximation:$A_n(x) := x\frac{\mathbf{N_n}(x)-\mathbf{N_n}(-x)}{\mathbf{N_n}(x)+\mathbf{N_n}(-x)}$.

Lemma V.2. *Given $n \in [5, \infty) \cap \mathbb{Z}$, $c \in [1, +\infty)$, $\sigma \in \{0, 1\}$*

$$\sup_{x\in[-c,c]} \left|\frac{|x|}{x^\sigma} - \frac{cA_n(x/c)}{x^\sigma}\right| \leq 3ce^{-\sqrt{n}}. \tag{11}$$

proof for Lemma 11. **If** σ**=0**, left of Eq. 11 is equivalent to

$$\left||x| - cA_n(\frac{x}{c})\right| = \left|c(|\frac{x}{c}| - A_n(\frac{x}{c}))\right| = c\left||\frac{x}{c}| - A_n(\frac{x}{c})\right|.$$

Since $|\frac{x}{c}|$ and $A_n(\frac{x}{c})$ are both even, it suffices to consider the case when $0 \leq x \leq c$.

For $x \in [0, c\alpha_n^n = ce^{-\sqrt{n}}]$, since $\mathbf{N_n}(x) \geq \mathbf{N_n}(-x) \geq 0$ so that $\frac{a}{c}A_n(\frac{a}{c}) \geq 0$:

$$c\left||\frac{x}{c}| - A_n(\frac{a}{c})\right| \leq c\left||\frac{x}{c}|\right| = x \leq ce^{-\sqrt{n}} < 3ce^{-\sqrt{n}}.$$

For $x \in (c\alpha_n^n = ce^{-\sqrt{n}}, c]$:

$$\begin{aligned}
&c\left||\frac{x}{c}| - A_n(\frac{x}{c})\right| \\
=&c\left|\frac{x}{c} - A_n(\frac{x}{c})\right| && (\frac{x}{c} > 0) \\
=&c\left|\frac{x}{c} - \frac{x}{c}\frac{\mathbf{N_n}(\frac{x}{c}) - \mathbf{N_n}(-\frac{x}{c})}{\mathbf{N_n}(\frac{x}{c}) + \mathbf{N_n}(-\frac{x}{c})}\right| && (\text{definition of } A_n) \\
=&2x\left|\frac{\mathbf{N_n}(\frac{x}{c})}{\mathbf{N_n}(-\frac{x}{c})} + 1\right|^{-1} \\
\leq&2c\frac{x}{c}\left[\left|\frac{\mathbf{N_n}(\frac{x}{c})}{\mathbf{N_n}(-\frac{x}{c})}\right| - |-1|\right]^{-1} && (|a-b| \geq |a| - |b|) \\
\leq&2c\left[\left|\frac{\mathbf{N_n}(\frac{x}{c})}{\mathbf{N_n}(-\frac{x}{c})}\right| - 1\right]^{-1} && (\frac{x}{c} \leq 1) \\
\leq&\frac{2c}{e^{\sqrt{n}} - 1} && (\text{Lemma 3.2, Ch. 7, [42]}) \\
\leq&\frac{2c}{e^{\sqrt{n}} - \frac{1}{3}e^{\sqrt{n}}} && (\frac{e^{\sqrt{n}}}{3} \geq \frac{e^{\sqrt{5}}}{3} \approx \frac{3.19}{3} > 1) \\
=&3ce^{-\sqrt{n}}.
\end{aligned}$$

If σ**=1**, Following same procedure as σ=0, we have:

$$\left|\frac{|x|}{x} - \frac{cA_n(x/c)}{x}\right| \leq 3ce^{-\sqrt{n}}.$$

□

proof for Theorem V.1. Applying Lemma 11:

$$\begin{aligned}
|f_1 - R(x)| =& \left|(ax + b|x|) - (ax + bcA_n(\frac{x}{c}))\right| \\
=&b\left||x| - cA_n(\frac{x}{c})\right| \leq 3bce^{-\sqrt{n}}.
\end{aligned}$$

Similarly, we have:

$$\begin{aligned}
|f_2 - R(x)| =& \left|(ax + b\frac{|x|}{x}) - (ax + b\frac{cA_n(x/c)}{x}))\right| \\
=&b\left|\frac{|x|}{x} - \frac{cA_n(x/c)}{x}\right| \leq 3bce^{-\sqrt{n}}.
\end{aligned}$$

In sum,

$$\sup_{x\in[-c,c]} |f_{1,2} - R_n(x)| \leq Ce^{-\sqrt{n}},$$

where $C = 3bc$ in Theorem V.1 □

By Bernstein's theorem [43], polynomials can approximate a function with:

$$||x| - P_n(x)| \leq \frac{\beta}{n},$$

where $P_n(x)$ is a polynomial function of degree of n, and $\beta \approx 2.801$ [44]. Using the same settings for the rational function, we have a similar result for polynomials:

Theorem V.3 (convergence rate of polynomial approximation on jump discontinuity)**.** *Given $n \in [5, \infty) \cap \mathbb{Z}$, $c \in [1, +\infty)$, $\sigma \in \{0, 1\}$:*

$$\sup_{x\in[-c,c]} |f_{1,2} - P_n(x)| \leq \frac{C\beta}{n},$$

where $P_n(x)$ is a polynomial function of degree n, and C=3bc.

In a nutshell, when the order is large or equal to 5, polynomial converges linearly regarding the order number, while rational function converges exponentially.

VI. Evaluation

This section elaborates evaluation with a detailed analysis of the behaviors of the proposed method on synthetic and real-world graphs.

A. Training Setting and Baselines

The input include a graph Laplacian $\mathbf{L}$, a graph signal residing on each vertex Y, and raw feature x . In a nutshell, we aims at finding a function f that satisfies $Y = f(\mathbf{L}, x)$:

$$Y = \mathbf{U}\, \mathrm{g}_\theta(\mathbf{\Lambda})\, \mathbf{U}^\intercal\, x = \mathrm{g}_\theta(\mathbf{L})x, \tag{12}$$

where $\mathrm{g}_\theta(\mathbf{\Lambda})$ is set to be jump function such $|x|$ and $sign(x)$. In previous works, raw features and filtering signal are fed into the model to fit the graph signal. However, raw features have an impact on fitting graph signal, which distracts the analysis of filtering behaviors. As discussed in Section IV, x is set to be eigenvector $\mathbf{U}$ which is a uniform signal in spectral domain, so that we can focus on the behaviors of approximation methods. We compare Rational Net(RatNet or RNet) against several stat of art regression models:

- Linear Regression(LR)
- Polynomial Regression(PR) [45]
- Passive Aggressive Regression(PAR) [46]
- LASSO [47]
- Epsilon-Support Vector Regression(SVR) [48]. Three kernels were applied: linear(L), polynomial(P) and RBF(R).
- Ridge Regression(RR) [49]
- Bayesian Ridge Regression(BR) [50],
- Automatic Relevance Determination(ARD) [51]
- Elastic Net(EN) [52]
- Orthogonal Matching Pursuit(OMP) [53]
- SGD Regression
- Huber Regression [54]
- ChebNet [18]. PolyNet is proposed by replacing Chebyshev polynomial with normal polynomial.

B. experiments on synthetic data

To validate the effectiveness of RationalNet, we conduct a simulated test with synthetic data. The task is to recover signal on the vertexes, which is a regression problem. Specifically, we generated a graph comprised of several subgroups. The edge amount for each vertex in the same subgroup is randomly chosen between 0 and 8, while the links among different subgroups are sampled between 0 and 3. Experiments were conducted on a 500-node and a 1000-node graph. Two types of jump signals are fed into this network structure: $|x|$ and $sign(x)$. Since all eigenvalues are normalized into range [0, 1], jump points of $|x|$ and $sign(x)$ are moved into the same range. Specifically, we used $|x - 0.5|$ and $sign(x - 0.5)$. Detailed results are shown in Table II and I.

Method	S-ERR($\|x\|$)	V-ERR($\|x\|$)	S-ERR($sign(x)$)	V-ERR($sign(x)$)
SVR-R	.0044±.0000	.0043±.0000	.3840±.0000	.2573±.0000
SVR-L	.0165±.0000	.0111±.0000	.3218±.0000	.2799±.0000
SVR-P	.0179±.0000	.0131±.0000	.3587±.0000	.2573±.0000
LR	.0161±.0000	.0110±.0000	.3211±.0000	.2788±.0000
RR	.0160±.0000	.0110±.0000	.3199±.0000	.2786±.0000
LASSO	.0157±.0000	.0137±.0000	.5581±.0000	.5087±.0000
EN	.0157±.0000	.0137±.0000	.5969±.0000	.5438±.0000
OMP	.0161±.0000	.0110±.0000	.3211±.0000	.2788±.0000
BR	.0161±.0000	.0110±.0000	.3210±.0000	.2788±.0000
ARD	.0161±.0000	.0110±.0000	.3210±.0000	.2788±.0000
SGD	.0152±.0000	.0116±.0000	.3191±.0001	.2795±.0003
PAR	.2871±.0997	.2740±.1033	1.0370±.8892	.9745±.8418
Huber	.0202±.0000	.0123±.0000	.3219±.0000	.2794±.0000
PolyFit	.0016±.0000	.0010±.0000	.2057±.0000	.1703±.0000
ChebNet	.0021±.0000	1.1904±.0052	.2058± .0067	.2084± .0043
PolyNet	.0016±.0000	.0038±.0000	.2011±.0095	.2001±.0056
RNet	**5.2971e-6±1.2501e-8**	**.0001±.00000**	**.0103±.0001**	**.0153±.0006**

TABLE I: 1000-node graph test: s-err indicates error in spectral domain, while v-err represents error in vertex domain.

Method	S-ERR(|x|)	V-ERR(|x|)	S-ERR($sign(x)$)	V-ERR($sign(x)$)
SVR-R	.0043±.0000	.0044±.0000	.2691±.0000	.2867±.0000
SVR-L	.0148±.0000	.0131±.0000	.2612±.0000	.2748±.0000
SVR-P	.0137±.0000	.0138±.0000	.2784±.0000	.2875±.0000
LR	.0140±.0000	.0130±.0000	.2582±.0000	.2734±.0000
RR	.0140±.0000	.0130±.0000	.2579±.0000	.2741±.0000
LASSO	.0135±.0000	.0137±.0000	.4723±.0000	.4865±.0000
EN	.0135±.0000	.0137±.0000	.5260±.0000	.5374±.0000
OMP	.0140±.0000	.0130±.0000	.2582±.0000	.2734±.0000
BR	.0140±.0000	.0130±.0000	.2581±.0000	.2734±.0000
ARD	.0140±.0000	.0130±.0000	.2581±.0000	.2734±.0000
SGD	.0135±.0000	.0138±.0000	.2597±.0007	.2764±.0008
PAR	.4026±.3980	.3982±.3954	.7412±.5682	.7456±.5029
Huber	.0158±.0000	.0135±.0000	.2581±.0000	.2734±.0000
PolyFit	.0010±.0000	.0011±.0000	.1488±.0000	.1699±.0000
ChebNet	.0044±.0000	.0044±.0000	.2025±.0000	.2115±.0004
PolyNet	.0016±.0000	.0016±.0000	.2059±.0000	.2083±.0004
RNet	**.0001±.0000**	**.0001±.0000**	**.0108±.0001**	**.1479±.0001**

TABLE II: 500-node graph test: s-err indicates error in spectral domain, while v-err represents error in vertex domain.

In 1000-node graph test on $|x|$(first two columns in Table I), PolyFit achieved the second lowest MSE(0.0016 for S-ERR). PolyNet's MSE(0.0016) is the same as that of PolyFit, which shows the power of polynomial regression. Chebyshev polynomial(ChebNet) dose not improve PolyNet, which implies that neural network might approximate the best coefficients of polynomials no matter what type of polynomial is used. LR, RR, LASSO, EN, OMP, BR, ARD, SGD SVR(L/P) performed at the same level(0.0015-0.0018). Our method(5e-6) significantly outperformed all the baselines by a large margin. Both the errors in spectral domain and vertex domain show the advantage of RationalNet. The Similarly, PolyFit and PolyNet and SVR(R) performed better than all the baselines except RationalNet. Our method still achieves the lowest MSE(0.004619 for S-ERR). The 500-node graph experiment(Table I)) also demonstrates the effectiveness of RationalNet.

Regression behaviors on synthetic data is shown in Fig. 3. Methods (SVR(L), Ridge, OMP, LASSO, Linear regression, ENet, ARD, Huber, etc.) fitted the $|x|$(upper of Fig. 3) using a straight Line, while better baselines(SVR(R)), PolyFit, PolyNet, ChebNet) approximate the function with curves. RationalNet almost overlapped with the target function which makes its MSE very small(5e-6). Similarly, in lower part of Fig.3, the methods using straight lines(SVR(L), Ridge, OMP, SGD, LASSO, BR, Huber LR, ENet, ARD) performed relatively bad. Fitting with curves, PolyFit, PolyNet, ChebNet

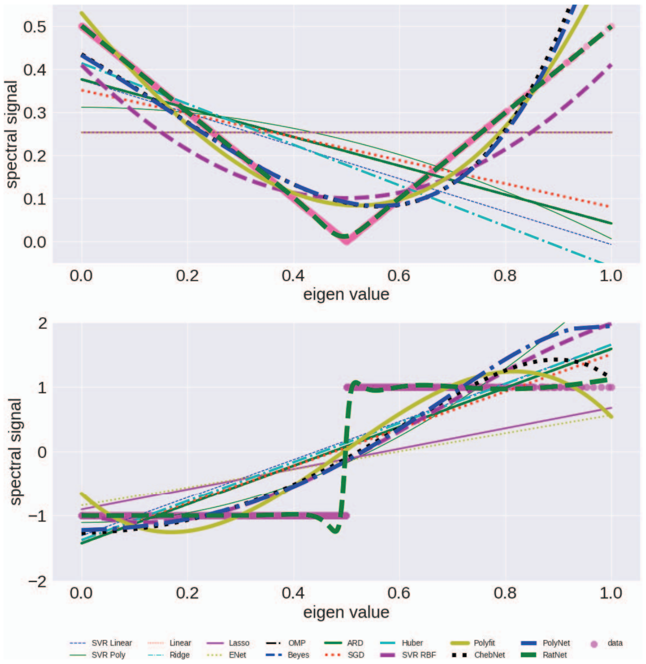

Fig. 3: Regression comparison on $|x|$ and $sign(x)$.

improved the performance by a large margin. Similarly, RationalNet overlapped the signal and achieved the lowest error.

	$\|x\|$	$sign(x)$
Remez (spectral)	4.531041e-6	0.057233
Remez (vertex)	0.000105	0.109641
RNet w/o Remez (spectral)	0.000145	1.364790
RNet w/o Remez (vertex)	0.000252	0.887602
RNet w/ Remez (spectral)	1.981569e-6	0.010645
RNet w/ Remez (vertex)	9.569891e-5	0.093268
Improved (spectral)	56.26%	81.39%
Improved (vertex)	9.37%	14.93%

TABLE III: Remez and RationalNet on 1000-node graph: MSE improvement in spectral and vertex domain

Since RationalNet initializes parameters by a relaxed Remez algorithm, we analyze the performance of neural networks and Remez respectively. As shown in Table III, the first two rows show the MSE of Remez only. Compared with Remez algorithm, RationalNet without Remez initialization(3rd and 4th lines) performed badly. On the contrary, RationalNet with Remez initialization(5th and 6th lines) improved the Remez by 56.26% and 81.39%(7th line) for $|x|$ and $sign(x)$ separately in spectral domain, which also reduces their MSE in vertex domain by 9.37% and 14.93%(8th line). This result illustrates that Remez and RationalNet cannot find the optimum independently. Therefore, it is reasonable to integrate these two methods for optimizing the coefficients.

Method	S-ERR(FF)	V-ERR(FF)	S-ERR(MI)	V-ERR(MI)
SVR-R	.0364±.0000	.0406±.0000	.0393±.0000	.0358±.0000
SVR-L	.0652±.0000	.0599±.0000	.0670±.0000	.0627±.0000
SVR-P	.1226±.0000	.1014±.0000	.0518±.0000	.0499±.0000
LR	.0640±.0000	.0595±.0000	.0662±.0000	.0621±.0000
RR	.0639±.0000	.0595±.0000	.0662±.0000	.0621±.0000
LASSO	.2026±.0000	.2030±.0000	.2141±.0000	.2138±.0000
EN	.1595±.0000	.1594±.0000	.1609±.0000	.1592±.0000
OMP	.0640±.0000	.0595±.0000	.0662±.0000	.0621±.0000
BR	.0640±.0000	.0595±.0000	.0662±.0000	.0621±.0000
ARD	.0640±.0000	.0595±.0000	.0662±.0000	.0621±.0000
SGD	.0639±.0001	.0598±.0000	.0664±.0000	.0622±.0000
PAR	.4960±.3273	.4948±.3200	.4255±.4575	.4222±.4588
Huber	.0646±.0000	.0597±.0000	.0666±.0000	.0624±.0000
PolyFit	.0346±.0000	.0382±.0000	.0384±.0000	.0346±.0000
ChebNet	.0468±.0006	.0468±.0006	.2336±.0094	.2336±.0094
PolyNet	.0468±.0006	.0468±.0006	.0490±.0049	.0490±.0009
RNet	**.0064±.0007**	**.0064±.0007**	**.0046±.0012**	**.0046±.0006**

TABLE IV: Regression comparison on Fairfax(FF) and Minnesota(MI) road networks. s-err indicates error in spectral domain, while v-err represents error in vertex domain.

C. Case study on real-world scenario

In this section, we study a traffic congestion signal on Minnesota state-level road network [2] and Fairfax county-level road network VA[3] [55]. Specifically, the signal is a high-pass filtering which can be written as $\zeta = \frac{sign(x-0.5)+1}{2}$ in Fourier domain. ζ is a threshold function sets the output to 0 when normalized eigenvalues $\in [0, 0.5)$, and 1 for $\in (0.5, 1]$. Therefore, this function filter out signal of low frequency. The physical meaning of the convolutional operation is a weight function that chooses the eigenbasis(φ_i) to fit the traffic signal Y. The top line of Fig. 4 shows several examples in eigen space of Minnesota road networks. First two sub figures are the 2nd and 3rd eigenvector φ_1, φ_2 on vertex domain: φ_1 emphasizes the south of Minnesota(red area), while φ_2 highlights the capital St. Paul and its biggest city Minneapolis. Note that the 1st eigenvector φ_0 is a constant vector for any connected graph. φ_1, φ_2 correspond to λ_1, λ_1, which represent the first two lowest frequencies. As these figures show, low frequencies represent smooth signals, which means that the neighbors of each node are likely to have similar signal value. By contrast, high-frequency basis captures non-smooth component as the 3rd and 4th sub figures show: signal values vary frequently in some areas. Combining top 50% high-frequency eigenbasis, the last sub figure shows the ζ signal on the graph. In addition, the degree of non-smoothness of signal regard graph structure can be evaluated quantitatively by Dirichlet energy([36]): $\mathcal{E}_{\varphi_i} = \varphi_i^\intercal \mathbf{L} \varphi_i$. Dirichlet energy of examples is shown in the caption of Fig. 4. Eigenvectors of low frequency($\varphi_{1,2}$) are smooth, so their Dirichlet energies are low. While high-frequency eigenvectors are less smooth since their Dirichlet energy is higher(around 4.04). Summing up the top 50% high frequencies, the Dirichlet energy of ζ in the last sub figure is very large(15384.10). The bottom line of Fig. 4 shows similar examples from Fairfax road networks. φ_1 highlights Fair City Mall(red area) and the road to this mall, while φ_2 underlines Fairfax Circle Shopping Center and a residential neighborhoods nearby. Similarly, φ_{701} and φ_{702}

[2]https://www.cise.ufl.edu/research/sparse/matrices/Gleich/minnesota.html
[3]https://github.com/gboeing/osmnx

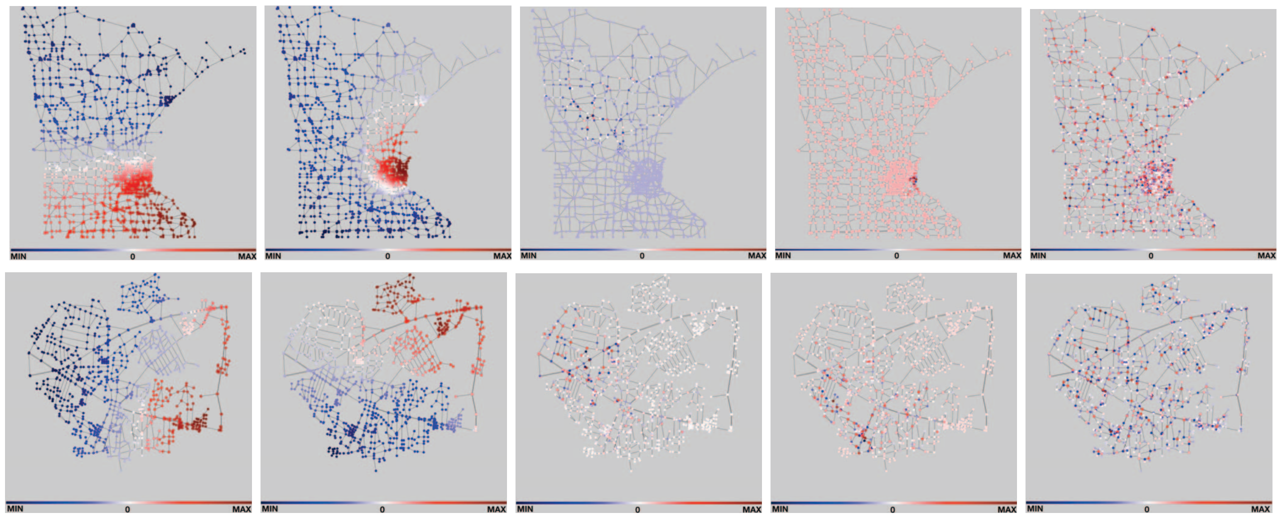

Fig. 4: Top line: Minnesota road network. From left to right are φ_1,φ_2,φ_{1001},φ_{1002}, and ζ. $\mathcal{E}_{\varphi_1}$=0.00084 $\mathcal{E}_{\varphi_2}$=0.00207 $\mathcal{E}_{\varphi_{2001}}$=4.03932 $\mathcal{E}_{\varphi_{2002}}$=4.04661, $\mathcal{E}_{\zeta}$=15384.10112. **Bottom line**: Fairfax road network. From left to right are φ_1,φ_2,φ_{701},φ_{702}, and ζ. $\mathcal{E}_{\varphi_1}$=0.00250 $\mathcal{E}_{\varphi_2}$=0.00296 $\mathcal{E}_{\varphi_{701}}$=3.82741 $\mathcal{E}_{\varphi_{702}}$=3.81540, $\mathcal{E}_{\zeta}$=6376.54224

show two non-smooth graph signals. Summing up top 50% high frequencies, the 5th sub figure exhibits an extremely non-smooth signal. Characterizing non-smooth graph signal or high frequencies is not a trivial task. Therefore, approximating this high pass filtering is significantly challenging. Table IV shows

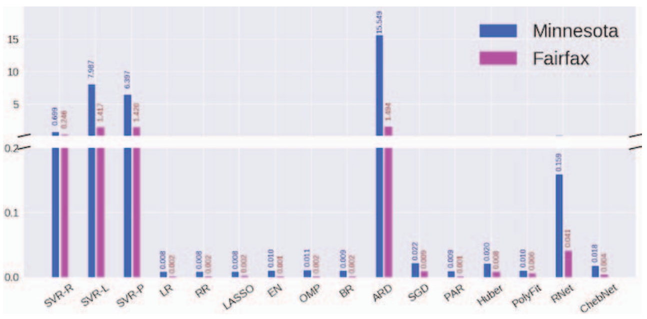

Fig. 5: Comparison of average running time in seconds.

similar results as in synthetic data: The proposed method still performed much better(3e-5) than the baselines. PolyFit achieved the second best level(0.0008), ChebNet, PolyNet and SVR(RBF) are generally good(0.0039,0.0039 and 0.0055), this is probably because they fitted the target with curves. The methods using straight lines have highest level of MSE(around 0.01). The results on another dataset, Fairfax road, also show that RationalNet has huge advantage beyond the baselines.

Fig. 5 shows the comparison of running time on two real-world networks. Minnesota dataset contains 2642 vertexes, while Fairfax network consists of 993 nodes. Most baseline methods are efficient such as LR, RR, LASSO, EN, OMP, BR, PAR. They finish computing within around 0.01 second on Minnesota graph and 0.002 second on Fairfax graph. SGD, PolyFit, and Huber only require 0.02 and 0.01 for Minnesota and Fairfax network respectively. SVR group performed slower, but they complete the calculation within 10 seconds for Minnesota graph and 2 seconds for Fairfax. ARD needs around 15 seconds and 1.4 seconds separately, which is the slowest baseline. Note that the number for RationalNet and ChebNet in Fig. 5 is the time for each iteration. RationalNet took 0.159 seconds for one update on Minnesota network, and 0.041 seconds on Fairfax network. In practice, RationalNet often converges within 300 iterations, which takes less than one minute for both datasets. Due to the complexity of computation, it is natural that RationalNet is slower than its counterpart ChebNet and several baselines. However, it shows that our algorithm can run reasonably fast in real-world datasets. Our case study on real-world graph justifies that RationalNet can accurately estimate the high pass filter within a reasonable time.

VII. Conclusion

In this paper, we have introduced a neural network model for graph signal recovering. To estimate jump discontinuity, a rational function is employed due to its powerful ability of approximation. The proposed method can avoid multiplication with the eigenvector matrix. With the help of a relaxed Remez algorithm, RationalNet can identify the optimal configuration. In theory, RationalNet obtains exponential convergence rate on jump signal, significantly fast than the polynomial-based approximation. Experiments on synthetic datasets suggest that the proposed RationalNet model is capable of model typical jump function accurately.

Acknowledgments

Feng's research is supported in part by NSF IIS-1441479, NSF IIS-1815696, and an NSF CAREER award IIS-1750911. Rongjie Lai's research is supported in part by NSF DMS-1522645 and an NSF Career award DMS-1752934. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V GPU used for this research.

References

[1] S. Liang and R. Srikant, "Why deep neural networks for function approximation?" in *International Conference on Learning Representations (ICLR)*, 2017.

[2] M. Telgarsky, "Neural networks and rational functions," in *International Conference on Machine Learning*, 2017, pp. 3387–3393.

[3] D. Lazer, A. S. Pentland, L. Adamic, S. Aral, A. L. Barabasi, D. Brewer, N. Christakis, N. Contractor, J. Fowler, M. Gutmann *et al.*, "Life in the network: the coming age of computational social science," *Science (New York, NY)*, vol. 323, no. 5915, p. 721, 2009.

[4] M. G. Bell, Y. Iida *et al.*, "Transportation network analysis," 1997.

[5] M. E. Newman, "Spread of epidemic disease on networks," *Physical review E*, vol. 66, no. 1, p. 016128, 2002.

[6] V. Marx, "High-throughput anatomy: charting the brain's networks," *Nature*, vol. 490, no. 7419, p. 293, 2012.

[7] E. H. Davidson, J. P. Rast, P. Oliveri, A. Ransick, C. Calestani, C.-H. Yuh, T. Minokawa, G. Amore, V. Hinman, C. Arenas-Mena *et al.*, "A genomic regulatory network for development," *science*, vol. 295, no. 5560, pp. 1669–1678, 2002.

[8] J. H. Drew and H. Liu, "Diagnosing fault patterns in telecommunication networks," Sep. 23 2008, uS Patent 7,428,300.

[9] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion." in *AAAI*, vol. 15, 2015, pp. 2181–2187.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[11] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 160–167.

[12] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[14] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[15] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *International Conference on Learning Representations (ICLR)*, 2013.

[16] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

[17] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[18] 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.

[19] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[20] Q. Li, Z. Han, and X. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *AAAI*, 2018.

[21] L. N. Trefethen, *Approximation theory and approximation practice*. Siam, 2013, vol. 128.

[22] L. V. Ahlfors, "Complex analysis: an introduction to the theory of analytic functions of one complex variable," *New York, London*, p. 177, 1953.

[23] R. Pachon, "Algorithms for polynomial and rational approximation," Ph.D. dissertation, University of Oxford, 2010.

[24] M. J. D. Powell, *Approximation theory and methods*. Cambridge university press, 1981.

[25] H. Cohen, *Numerical approximation methods*. Springer, 2011.

[26] P. P. Petrushev and V. A. Popov, *Rational approximation of real functions*. Cambridge University Press, 2011, vol. 28.

[27] N. I. Achieser, *Theory of approximation*. Courier Corporation, 2013.

[28] E. Ziegel, "Numerical recipes: the art of scientific computing," 1987.

[29] J. P. Boyd, *Chebyshev and Fourier spectral methods*. Courier Corporation, 2001.

[30] J. C. Mason and D. C. Handscomb, *Chebyshev polynomials*. CRC Press, 2002.

[31] E. Y. Remez, "Sur la détermination des polynômes dapproximation de degré donnée," *Comm. Soc. Math. Kharkov*, vol. 10, pp. 41–63, 1934.

[32] F. R. Chung, *Spectral graph theory*. American Mathematical Soc., 1997, no. 92.

[33] R. Grone, R. Merris, and V. S. Sunder, "The laplacian spectrum of a graph," *SIAM Journal on Matrix Analysis and Applications*, vol. 11, no. 2, pp. 218–238, 1990.

[34] K. C. Das, "The laplacian spectrum of a graph," *Computers & Mathematics with Applications*, vol. 48, no. 5-6, pp. 715–724, 2004.

[35] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[36] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[37] D. I. Shuman, B. Ricaud, and P. Vandergheynst, "Vertex-frequency analysis on graphs," *Applied and Computational Harmonic Analysis*, vol. 40, no. 2, pp. 260–291, 2016.

[38] X. Zhu and M. Rabbat, "Approximating signals supported on graphs," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3921–3924.

[39] G. Taubin, "A signal processing approach to fair surface design," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM, 1995, pp. 351–358.

[40] P. Macgregor, "Numerical methods for scientists and engineers by h. m. antia," vol. 88, pp. 404–405, 01 2004.

[41] D. J. Newman *et al.*, "Rational approximation to $|x|$." *The Michigan Mathematical Journal*, vol. 11, no. 1, pp. 11–14, 1964.

[42] G. G. Lorentz, M. von Golitschek, and Y. Makovoz, *Constructive approximation: advanced problems*. Springer Berlin, 1996, vol. 304.

[43] N. Achiezer, "Theory of approximation (transl. by cj hyman), ungar, new york, 1956," *Google Scholar*.

[44] R. S. Varga and A. J. Carpenter, "On the bernstein conjecture in approximation theory," *Constructive Approximation*, vol. 1, no. 1, pp. 333–348, Dec 1985. [Online]. Available: https://doi.org/10.1007/BF01890040

[45] S. M. Stigler, "Gergonne's 1815 paper on the design and analysis of polynomial regression experiments," *Historia Mathematica*, vol. 1, no. 4, pp. 431–439, 1974.

[46] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *Journal of Machine Learning Research*, vol. 7, no. Mar, pp. 551–585, 2006.

[47] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[48] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[49] A. Y. Ng, "Feature selection, l 1 vs. l 2 regularization, and rotational invariance," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 78.

[50] D. J. MacKay, "Bayesian interpolation," *Neural computation*, vol. 4, no. 3, pp. 415–447, 1992.

[51] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of machine learning research*, vol. 1, no. Jun, pp. 211–244, 2001.

[52] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[53] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on signal processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[54] P. J. Huber, "Robust statistics," in *International Encyclopedia of Statistical Science*. Springer, 2011, pp. 1248–1251.

[55] G. Boeing, "Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks," *Computers, Environment and Urban Systems*, vol. 65, pp. 126–139, 2017.