SPECTRAL GRAPH NEURAL NETWORKS AND HERMITE CONVOLUTION FILTERS

A PROJECT REPORT

submitted by

Naveen S
(22MMA018)

to
National Institute of Technology Hamirpur
in part meeting the prerequisites needed to be awarded the degree
of
Master of Science
in
Mathematics and Computing

Department of Mathematics and Scientific Computing
National Institute of Technology Hamirpur
Hamirpur
177005
May 2024

CANDIDATE'S DECLARATION

I, the undersigned, hereby declare that the project titled **Spectral Graph Neural Networks and Hermite Convolution Filters** submitted for partial fulfillment of the requirements for the degree of Master of Science from the National Institute of Technology, Hamirpur, is my authentic work done under the supervision of Dr. Jeetendrasingh Maan, my guide. This submission reflects some of my initial ideas. When I adapted ideas or words from others, I have correctly cited and referenced the original sources. I confirm that I have followed the principles of academic honesty and integrity and have not misrepresented or falsified any data, idea, fact, or source in my submission. I recognize that any violation of these principles could potentially lead to disciplinary action by the institute and/or the university, as well as possible penalties from sources that were not properly cited or from which proper permission was not obtained. This report has not been used as the basis for any degree, graduating diploma, or similar title from any other university.

Place	:	Signature of student	: .....
Date	: May 20, 2024	Name of student	: Naveen S

ACKNOWLEDGMENT

I want to express my sincere gratitude to everyone who helped me prepare and present this project report titled **Spectral Graph Neural Networks and Hermite Convolution Filters** satisfactorily.

My deepest appreciation goes to my guide and supervisor, Dr. Jeetendrasingh Maan in the Department of Mathematics and Scientific Computing for their invaluable suggestions and critical feedback throughout the report preparation process. Their guidance significantly improved the quality of this work.

I would also like to thank Dr. Ramesh Kumar Vats, Head of the Department of Mathematics and Scientific Computing for their encouragement during this project.

Additionally, I am grateful to the Department of Mathematics and Scientific Computing at the National Institute of Technology, Hamirpur for providing the resources and support that were essential for completing this project.

Finally, a special thank you to my classmates for their unwavering support. Their willingness to listen to my project presentations and offer feedback has been invaluable.

Naveen S

(Reg. No. 22MMA018)

M Sc. (Mathematics and Computing)

Department of Mathematics and Scientific Computing

National Institute of Technology Hamirpur

ABSTRACT

Spectral Graph Theory, rooted in graph Laplacian eigenvalues and eigenvectors, provides insights into graph connectivity, clustering, and community detection. The spectral decomposition of graphs reveals hidden structures and patterns, making it a fundamental tool across scientific domains. Graph Neural Networks, inspired by spectral methods and graph convolutional operations, have achieved remarkable success in learning representations of nodes and graphs. These models extend traditional neural networks to handle non-grid data structures, enabling them to capture complex relationships and dependencies inherent in graphs.

This research introduces a new convolution operator in graph-based on Hermite polynomials, aiming to approximate the filter function used for message passing operations in graphs. We conduct experiments in Python to compare our proposed model, Hermiteconv, with already existing models to gain a deeper understanding. Hermiteconv demonstrates robustness and reliability in performance across different inputs compared to Chebynet, which utilizes Chebyshev polynomials as a convolution operator.

CONTENTS

Chapter 1

UNDERSTANDING GRAPH MATRICES

1.1 GRAPH THEORY

In this chapter we collect the basic definition which are needed for subsequent chapters.

Definition 1.1.1. A **graph** is a complex data structure consisting of a collection of nodes and the relationships between them. A graph G can be represented as an ordered pair $G = (V, E)$, where V denotes the vertices(fundamental unit of a graph representing an entity) and E denotes the edges(connection between two vertices).

Definition 1.1.2. The **degree** of a vertex v_i, where $i = 0, 1, 2, \cdots, n$, is the number of edges incident on it.

Definition 1.1.3. A **path** in a graph is a ordered collection of edges which connects nodes in a sequence.

Definition 1.1.4. The **neighbors** of a particular vertex v_i is the different vertices v_j that share common edges.

Definition 1.1.5. Graph convolution is the process of transforming features associated with a node by aggregating information from neighbors at different levels.

Definition 1.1.6. A **feature vector** associated with a node corresponds to the attributes held by the particular node v_i.

1.2 LINEAR ALGEBRA

Here is an introduction to some key concepts in linear algebra.

Definition 1.2.1. A **diagonal matrix** is a $n \times n$ matrix where all non diagonal entries represented by a_{ij}, $i \neq j$ are zero and diagonal entries non zero. It is represented by

$$\begin{pmatrix} \mu_1 & 0 & \cdots & 0 \\ 0 & \mu_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mu_n \end{pmatrix}$$

Definition 1.2.2. Two vectors u and v in $\mathbf{R}^n$ are said to be orthogonal to each other, if $u^{'}v$=0. The square matrix K of order n is said to be orthogonal if $KK^{'} = K^{'}K = I$.

Definition 1.2.3. Euclidian Norm is widely used in GNN models to normalize node features.

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}.$$

Definition 1.2.4. Let A be $n \times n$ matrix. Then $\lambda \in \mathbf{R}$ is an eigenvalue if for $v \in \mathbf{R}$

$$Av = \lambda v,$$

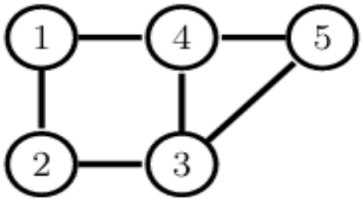
where v is called the eigenvector of A and $v \neq 0$.

In the following sections, we discuss about the important matrices of graphs which are foundation for graph convolutional networks.

1.3 ADJACENCY MATRIX

The adjacency matrix corresponds to a graph G is given by A. It denotes the connections between two nodes, defined as follows:

$$a_{ij} = \begin{cases} 1 & \text{if } v_i \text{ share edge with } v_j \text{ in G} \\ 0 & \text{otherwise.} \end{cases}$$



Then

$$A(G) = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Here it is evident that adjacency matrix of any graph G is symmetric matrix with diagonal entries zero. $i \neq j$, then the pricipal matrix formed will have deter-

minant -1. If vertex i is not adjacent to vertex j then then submatrix is given by $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. Otherwise it is zero matrix.

There are two main types of adjacency matrices, weighted and unweighted.

Unweighted Adjacency Matrix:

$$A_{ij} = \begin{cases} 1 & \text{if there is an connection between nodes } i \text{ and } j \\ 0 & \text{otherwise.} \end{cases}$$

Weighted Adjacency Matrix:

$$W_{ij} = \text{weight of the edge between nodes } i \text{ and } j.$$

1.4 LAPLACIAN MATRIX

The Laplacian matrix, often denoted as L, is a matrix representation of a graph's topology. It encapsulates information about the relationships between nodes in the graph. The Laplacian matrix plays a fundamental role in spectral graph theory and various graph-based algorithms.

The Laplacian matrix for an unweighted graph is

$$L = D - A, \tag{1.1}$$

where $L \in R^{n \times n,}$ and A is unweighted adjacency matrix and for weighted graph it is

$$L = D - W. \tag{1.2}$$

The Laplacian matrix in the normalized symmetric form can be represented as [4]

$$\hat{L} = I - D^{-1/2} A D^{-1/2}. \tag{1.3}$$

1.4.1 Spectral factorization

spectral factorization also known as spectral decomposition is the process of converting a matrix in the form of product of its eigenvalues and eigenvectors. Consider a matrix with n linearly independent eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ corresponding to eigenvalues $\mu_1, \mu_2, \ldots, \mu_n$, then spectral decomposition represents A as

$$A = Q \Lambda Q^{-1},$$

where Q is the matrix containing eigenvectors of A, i.e., $Q = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n]$, Λ is the diagonal matrix containing the eigenvalues of A, i.e., $\Lambda = \text{diag}(\mu_1, \mu_2, \ldots, \mu_n)$, Q^{-1} is the inverse of the matrix Q.

In the context of the Laplacian matrix L of a graph, spectral decomposition allows expressing L as a product of matrices involving its eigenvalues and eigenvectors, providing insights into the structural properties of the graph.

It is evident that the Laplacian matrix L associated with an undirected graph is positive semi-definite[1]. Let $f = \{f_1, f_2, ..., f_n\}$ be an arbitrary vector[4].

$$\begin{aligned} f^T L f = f^T D f - f^T W f &= \sum_{i=1}^{n} D_{i,i} f_i^2 - \sum_{i,j}^{n} f_j f_i W_{i,j} \\ &= \frac{1}{2} (\sum_{i=1}^{n} \sum_{j=1}^{n} W_{i,j} (f_i - f_j)^2). \end{aligned}$$

1.4.2 Incidence Matrix

The incidence matrix B of a graph G with n vertices and m edges is a $n \times m$ matrix, denoted as $B = [b_{ij}]$, where each entry b_{ij} is defined as follows:

$$b_{ij} = \begin{cases} 1 & \text{if vertex } i \text{ is incident to edge } j \\ -1 & \text{if vertex } i \text{ is incident to edge } j \text{ with opposite orientation} \\ 0 & \text{otherwise.} \end{cases}$$

In other words, if vertex i is an endpoint of edge j, b_{ij} is set to 1. If vertex i is an endpoint of edge j but with opposite orientation, b_{ij} is set to -1. Otherwise, b_{ij} is 0.

This matrix representation captures the relationships between vertices and edges in the graph, providing a way to encode the graph's connectivity and topology in a matrix format.

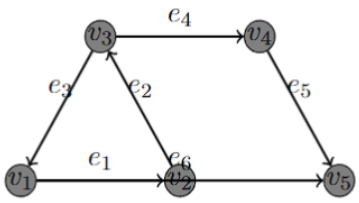


Figure 1: Graph G

$$B = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & 0 & -1 \end{bmatrix}$$

Some Properties

- Columns Sum is zero.
- rank(B) = n-1.
- If x is the left null space of B and if i is adjacent to j then

$$x^T B = 0.$$

implies

$$x_i = x_j.$$

- If there are k components in the graph then

$$\text{rank}(B) = n - k.$$

1.4.3 Laplacian Operator

The Laplacian operator is like a tool used in graph convolution. Graph convolution is a process where each node in a graph gathers information from its nearby nodes. Think of it as neighbors sharing information with each other. The Laplacian operator is what helps nodes perform this information gathering effectively, by considering the connections between them. So, you can think of the Laplacian matrix as the "recipe" that guides this process on the graph.

Let $l = \{l_1, l_2, ..., l_n\}^T$ be signal vector or node features associated with vertices $u = \{u_1, u_2, ..., u_n\}$. Let the incidence matrix is denoted by B.

$$B = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & 0 & -1 \end{bmatrix}$$

The matrix multiplication can be represented as follows: where B is the incidence matrix of directed graph and B^T is the transpose of incidence matrix.

$$B \begin{pmatrix} l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} l_1 \\ l_2 \\ l_3 \\ l_4 \\ l_5 \end{pmatrix} = \begin{pmatrix} l_1 - l_2 \\ l_2 - l_3 \\ l_3 - l_1 \\ l_3 - l_4 \\ l_4 - l_5 \\ l_2 - l_5 \end{pmatrix}$$

$$Bx_{i,j} = l_i - l_j,$$

if u_i is connected with u_j in direction i to j.

Consider $B^T(Bl)$ as

$$B^T(Bl) = \begin{pmatrix} 1 & 0 & -1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 1 \\ 0 & -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 \end{pmatrix} \begin{pmatrix} l_1 - l_2 \\ l_2 - l_3 \\ l_3 - l_1 \\ l_3 - l_4 \\ l_4 - l_5 \\ l_2 - l_5 \end{pmatrix} = \begin{pmatrix} 2l_1 - l_2 - l_3 \\ -l_1 + 3l_2 - l_{23} - l_5 \\ -l_1 - l_2 + 3l_3 - l_4 \\ -l_3 + 2l_4 - l_5 \\ -l_2 - l_4 + 2l_5 \end{pmatrix}$$

The Laplacian matrix L operating on given graph g given by Lx.

$$L = D - A,$$

where D is the degree matrix and A is the adjacency matrix

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$Lx = (D-A)x = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 3 & -1 & 0 & -1 \\ -1 & -1 & 3 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & -1 & 0 & -1 & 2 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 2x_1 - x_2 - x_3 \\ -x_1 + 3x_2 - x_{23} - x_5 \\ -x_1 - x_2 + 3x_3 - x_4 \\ -x_3 + 2x_4 - x_5 \\ -x_2 - x_4 + 2x_5 \end{pmatrix}$$

For given graph we can see

$$Lx = B^T(Bx),$$

and so

$$L = B^T B.$$

Consider an weighted graph with adjacency matrix W with entries $W_{i,j}$ always greater than or equal to zero. Non-zero if there is an edge connecting.

$$L = D - W$$

implies

$$(Lx)_i = (Dx - Wx)_i.$$

If we consider a graph with 2 vertices u_1 and u_2 and signal vector associated with vertices are f_1 and $_2$, respectively, then one has

$$Lf = (W - A)f = Wf - Af = \begin{bmatrix} k_{1,1} & k_{1,2} \\ k_{2,1} & k_{2,2} \end{bmatrix} \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} - \begin{pmatrix} W_{1,1} & W_{1,2} \\ W_{2,1} & W_{2,2} \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \end{pmatrix},$$

equivalently,

$$(Lf)_i = k_{i,i} f_i - \sum_{j=1}^{N} W_{i,j} f_j,$$

$$(Lf)_i = \sum_{j=1}^{N} W_{i,j}(f_i - f_j).$$

Further it can be rewritten in a quadratic form as

$$x^T Lx = \frac{\sum_{e_{i,j}} W_{i,j}(f_i - f_j)^2.}{2}$$

Laplacian operator in such way helps aggregating information from neighboring nodes.

Chapter 2

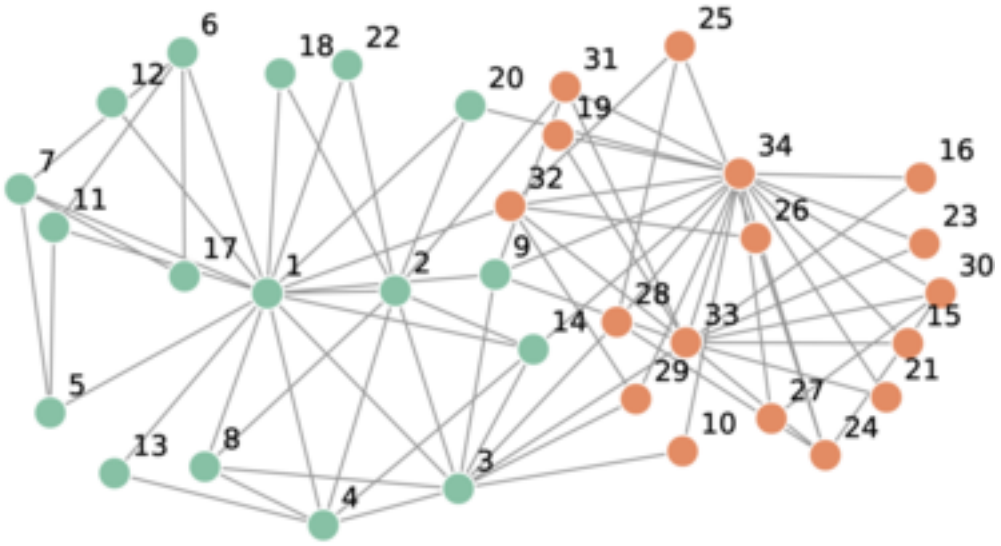
GNN'S AND SPECTRAL GRAPH CONVOLUTIONS

Spectral graph convolution is a procedure used in graph neural networks for processing data represented in graph. It uses concepts from discrete signal processing, Fourier analysis to perform convolution operation.

2.1 INTRODUCTION

A network or graph, represented as G, is a complex interconnected system consisting of nodes and edges, typically used to represent intricate relationships. In other words, a network helps describe how multiple individuals or instances are connected. For example, we can represent a group of individuals in a function with a network, where two individuals are connected by an edge if they shake hands. A proper way to represent these connections, besides a graph, is through an adjacency matrix.

let A be the adjacency matrix such that $[A]_{v\times v}$, where V is the collection of nodes.



2.1.1 Attribute Information

The feature information associated with graph usually means the attributes linked with nodes or edges, that can be represented with the help of a real matrix $X \in R_{|V| \times k}$.

Graphs are versatile data structures used to represent relationships and connections between entities. Here are examples of graph data in different domains:

Social networks model relationships between individuals or entities. Nodes represent users, while edges denote connections or friendships. Examples include Facebook, Twitter, and LinkedIn networks. Biological graphs represent relationships between proteins, genes, or species. Nodes signify biological entities, and edges represent interactions or evolutionary links. Examples include financial Networks or Supply Chain Networks. Recommendation systems use graphs where nodes represent users or items, and edges indicate user-item interactions or preferences. Examples include collaborative filtering systems.

2.2 GRAPH MACHINE LEARNING

Machine Learning on graph-structured data involves developing algorithms and models that can operate on graphs to make predictions, classifications, or uncover patterns within the data. Graph-based machine learning is adapted to a variety of domains where relationships and connections between entities are critical.

2.2.1 Node Level Predictions

A graph contains nodes, which are used to represent individuals or data points. Each node is associated with a number of features. The main goal of node-level prediction is to find the dependent variable y, which can be a continuous or categorical value, assigned to each node as in normal supervised machine learning tasks. Among node-level predictions, node classification has gained significant popularity. Examples include HIV inhibitor classification and congestion point prediction in traffic networks.

2.2.2 Edge Level Prediction

Node classification aids in inferring the class or continuous label associated with each node. If the task were solely about predicting node features, there would be no need to design such a complex structure for machine learning. Since graphs describe the relationships between nodes, it is also crucial to consider edge features. For instance, in the previously mentioned example of HIV inhibitor classification, it is essential to understand the interactions between HIV-causing cells and different inhibitor cells that can potentially inhibit HIV cells. In such cases, the training data includes known reactions, but the goal is to predict outcomes for new inhibitor cells. Therefore, edge-level inference is also important to predict the presence or absence of edges between nodes in a network.

Popular edge-level prediction applications in real life include gene-disease

association, call predictions, and traffic flow prediction.

2.2.3 Clustering

Node classification and relation prediction, which are essentially the graph equivalents of supervised learning, both call for inferring missing information about the graph data. Conversely, community detection is the graph an equivalent of uncontrolled grouping.

Consider a large online social platform where users are connected based on friendships and shared activities. Community detection algorithms can analyze the network structure to identify clusters of users who frequently interact with each other. These communities may represent distinct interest groups, such as sports enthusiasts, music lovers, or gamers. By understanding the community structure, the platform can enhance user experience by providing targeted content recommendations, facilitating more accurate advertising targeting, and fostering community engagement.

2.3 NETWORK SIGNAL PROCESSING

network signal processing is the adaptation of classical signal processing on graphs.

2.3.1 Graph Filters

Graph filters play a crucial role in Graph Neural Networks (GNNs) by aggregating information from neighboring nodes and representing features in a more graphical manner. A graph consists of nodes and the connections between them, making it essential to consider these relationships when transforming features so that every node inherits the structural data of the network. Graph filters help capture both local and global patterns in network data and are specifically designed to

operate on non-Euclidean data. In classical signal processing, a filter accepts an input x and transforms it into another form $\tilde{x} = H(x)$, which can be represented as follows

$$\tilde{x}_i = \sum_{j=1}^{n} W_{i,j} x_i, \tag{2.1}$$

which is equivalent to

$$\tilde{x} = Wx, \tag{2.2}$$

where W weighted adjacency matrix

2.3.2 Spectral Graph Transform

In traditional signal processing, the Fourier transform decomposes a signal into its constituent frequency components [10], revealing the signal's frequency-domain representation. In a similar vein, the Laplacian matrix's eigenvectors indicate frequency components that function as the graph's Fourier basis, while the matrix's eigenvalues could reflect graph frequencies and create the graph's spectrum.

Let the eigen basis which is also called harmonic basis of graph be $U = (u_1, u_2, \cdots, u_n)$ where $u_k \in R^n$, $k = 1, 2, \cdots, n$ corresponding to eigenvalues of matrix $\{\mu_1, \mu_2, \cdots, \mu_n\}$ and node's signal vectors be $x = (x_1, x_2, \cdots, x_n)^T$. After graph fourier transform we get $x = (x_1, x_2, \cdots, x_n)^T$ become

$$\tilde{x} = (\tilde{x}(\mu_1), \mu_x(\mu_2), \cdots, \tilde{x}(\mu_n))^T.$$

It can be represented as general as

$$\tilde{x} = U^T x,$$

and

$$x = U\tilde{x},$$

which is the inverse graph Fourier transform.

Hence, by using the eigenvectors of the Laplacian as the basis functions, any signal distributed across the graph can be represented.

$$x = \tilde{x}(\mu_1)v_1 + \tilde{x}(\mu_2)v_2 + ... + \tilde{x}(\mu_n)v_n$$

Analogous to the Spectral transform, the GFT decomposes a graph vertex features or signals into its constituent frequency components, showing the signal's spectral properties with respect to the graph structure.

2.4 CONVOLUTION OPERATIONS

In convolution neural networks, convolution operations uses the local structure of data in grid structure like pixels in images. Extending this idea to the non grid structure, graph convolution operations are utilized. Spectral graph convolution operates in the spectral domain by leveraging the graph Fourier transform. The Graph signal's spectral representation with the help of filters helps to exploit local and global features of graph. Graph Laplacian filters are commonly used to capture smoothness properties of graph signals.

In the Fourier domain convolution operation is defined by

$$f_\theta(\cdot_G)x = f_\theta(L)x = f_\theta(Q\Lambda Q^T)x = Qf_\theta(\Lambda)Q^T x, \tag{2.3}$$

where $f_\theta(\Lambda) = diag(Q^T x)$ and $f \in \mathbf{R}^n$ is used to filter signal x. Different choices of $(\cdot_G)$ leads to various spectral GNN models.

Chapter 3

HERMITE POLYNOMIALS AS CONVOLUTION FILTER

3.1 INTRODUCTION

An innovative graph neural network model (GNN) capable of capturing higher-order dependencies in complex graph data has been developed. Existing models such as ChebNet often exhibit poor stability when encountering complex data due to the presence of higher-order neighbors, resulting in decreased interpretability. Hermite polynomials, which are a set of orthogonal polynomials, have the ability to accurately approximate any function within the real line to any desired degree. Compared to traditional GNN models like GCN, the GNN model utilizing Hermite polynomials has demonstrated an enhanced ability to capture higher-order dependencies within a graph, thereby enabling the capture of more complex patterns.

In traditional GNNs like Chebynet, which utilize Chebyshev polynomials, small changes in the input graph, such as adding or removing nodes or edges, can lead to disproportionately large changes in the model's predictions. However, the use of Hermite polynomials facilitates adaptation to address this issue. Therefore, addressing stability issues plays a significant role in ensuring the robustness of the model.

3.2 HERMITECONV

In this section, we proposed a innovative graph neural network model named Hermiteconv, which utilizes K-order Hermite polynomials to approximate filter functions used for graph signal transform. Hermite polynomials are a family of orthogonal polynomials defined on the entire real line and are orthogonal to e^{-x^2}. The explicit expression for Hermite polynomials of order N is given by

$$h(z) = N! \sum_{l=0}^{\lfloor N/2 \rfloor} \frac{(-1)^l (2)^{N-2l}}{l!(N-2l)!} (z)^{N-2l} \gamma_l. \tag{3.1}$$

Given $k(z)$ as the order N Hermite polynomial. γ_l is treated as a model parameter of length l where $l = 0, 1, 2, \cdots, N$. That is, γ_l will be initialized as a trainable parameter to store the coefficients of the basis of Hermite polynomials of a particular order.

The convolution operation can be expressed in the following way[9],

$$N! \sum_{l=0}^{\lfloor N/2 \rfloor} \frac{(-1)^l (2)^{N-2l}}{l!(N-2l)!} (\tilde{L})^{N-2l} \gamma_l X,$$

where X node signals and $\tilde{L}$ is the Symmetric normalized Laplacian matrix.

Orthogonality

Hermite polynomials are a family of orthogonal polynomials defined on the entire real line and they are orthogonal to e^{-x^2}.The Hermite polynomials are orthogonal, given by

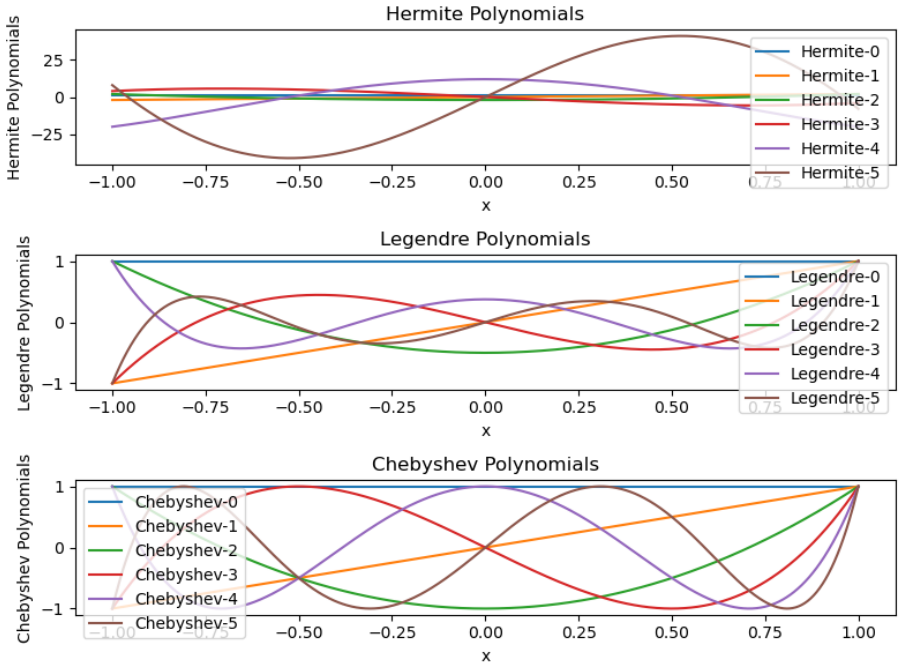
$$\int_{-\infty}^{\infty} H_m(z) H_n(z) e^{-x^2} dz = \sqrt{\pi} 2^n n! \delta_{mn}, \tag{3.2}$$

δ_{mn} is the Kronecker delta. Although Hermite polynomials exhibit more rapid oscillations as the degree of the polynomial increases, especially outside

the interval $[-1,1]$, within this interval, Hermite polynomials, similar to Legendre polynomials[3], provide a very good approximation and smooth behavior for functions. Chebyshev polynomials defined as $T_n(z)$ with orthogonality property

$$\int_{-1}^{1} T_m(z)T_n(z)(\sqrt{1-z^2})^{-1}dz = \pi\delta_{mn}.$$

They tends to show more oscillations on the interval $[-1,1]$ since the main objective is to reduce error.



Comparing Chebyshev and Legendre polynomials, Hermite polynomials have the upper hand in showing smoothness over the interval. This suggests that during the filtering process, they can deal with anomalies in the graph and variations in node degrees more skillfully. By contrast, Chebyshev polynomials, which minimise errors within the interval $[-1,1]$, are frequently used in classic GNN algorithms. Hermite polynomials are also subjected to irregularities at higher orders but less compared to Chebyshev.

3.2.1 Spectral Representation using Hermite Polynomials

Using filtering function $g_\theta(\cdot_G)]$, we can filter a signal or node feature X using the filter $g_\theta(\cdot_G)$ as

$$f_\theta(\cdot_G)X = f_\theta(L)X = f_\theta(Q\Lambda Q^T)X = U f_\theta(\Lambda)Q^T X. \tag{3.3}$$

For $l = 1,2,...,N$, $h_l^N(z) = \frac{(-1)^l(2)^{N-2l}}{l!(N-2l)!}$,

be the l-th order Chebyshev-Hermite basis, γ_l is the coefficient of $h_l^N(z)$.

We represent Hermite polynomials as follows

$$H_N(z) = \sum_{l=0}^{N} h_l^N(z).\gamma_l. \tag{3.4}$$

In Hermite-net, the given filter function h is a set of Hermite polynomials that is used to approximate graph signals. We define the spectral filter as $Qf_\theta(\Lambda)Q^T$ where $f_\theta(\Lambda) = diag[H_K(\mu_1), H_K(\mu_2), ..., H_K(\mu_n)]$. For the given input features X the convolution operation is given by

$$\begin{aligned} z &= Qdiag[H_K(\mu_1), H_K(\mu_2), ..., H_K(\mu_n)]Q^T X, \\ &= N! \sum_{l=0}^{\lfloor N/2 \rfloor} \frac{(-1)^l(2)^{N-2l}}{l!(N-2l)!}(\tilde{L})^{N-2l}\gamma_l X. \end{aligned}$$

3.2.2 Related work

Graphs are one of the versatile data structures that is well known for their expressive power. Machine Learning models are implemented in this domain that extract information from underlying graph structure such as link prediction, classification , modeling complex elements and their relationships.

Graph Convolutional Neural Networks (CNNs) are a novel type of neural network which is the generalization of classical Convolutional Neural Networks

designed specifically for complex graph structured data[8][6]. Researchers are continuously exploring new methods and ideas in the domain.The ability to learn sequence of filters to extract complex information makes CNN's so effective.

As the size of complex graph data continuous to increase in the current time various architectures that handle graph data comes into existence which is totally based on convolution theorem. SCNN leverages the spectral decomposition of the graph Laplacian matrix to define convolutional filters for graph signals. These filters are applied in the spectral domain and allow for capturing localized information in the graph structure[2]. GCN [5] uses neighborhood aggregation to aggregate and update node features. Chebynet that uses Chebyshev polynomials for graph convolution[6].Graphheat uses a heat kernel function to perform heat diffusion on graph signals and constructs a graph convolution operator based on the diffusion coefficients obtained during the heat diffusion process [12]. Graph Sample and Aggregated Embeddings (GraphSAGE)[7] uses neighborhood sampling and aggregation techniques to learn informative representations of nodes in large-scale graphs, enabling effective learning on graph-structured data. Graph Attention Networks (GATs) [11] utilizes attention mechanism gives more weight to the relevant and less weight to the less relevant parts. As a result, by concentrating on the most crucial data, the model is able to produce forecasts that are increasingly accurate.

3.3 TEST PROCEDURE

In this Section we will compare our Proposed model Hermiteconv performance to cutting-edge graph neural networks using publicly accessible graph datasets.

Dataset explanation

In this section we will provide explanation for the datasets used which in-

clude Pytorch Geometric library datasets used by researchers around the globe. They are classic datasets like the citation networkdatasets like Cora, CiteSeer, and PubMed where nodes represent papers and edges represent citation links between papers. The University of Texas at Austin's computer science department provided web pages for the Texas and Cornell portions of the WebKB dataset. Nodes represent web pages, and edges represent hyperlinks between these pages. AMZ Computers and Photo are co-purchase graphs that were taken from Amazon, with features being bag-of-words vectors that were taken from the 74 product reviews, nodes representing products, and edges representing the co-purchased relations of products. We will try to find the accuracy of our proposed model on these different classical datasets and try to find advantages over already existing models.

	Nodes	**Edges**	**Features**	**Classes**
Cora	2,808	5,429	1,433	7
CiteSeer	3,312	4,732	3,703	6
PubMed	19,617	44,338	500	5
Computers	13,814	24,4571	767	10
Photo	7,652	11,9099	749	8
Texas	183	279	1,703	5
Cornell	183	277	1,703	5

Table 3.1: Description of datasets used for GNN research.

Amoung the 7 datasets given CORA and CiteSeer exhibit lower order dependency relationship between nodes. In the CORA dataset some nodes are connected to large number of other nodes which are usually first order relationships demonstrating lower order dependency. Similarly, the Cornell and Texas datasets exhibits typical higher order dependency. We also take into account that low-pass datasets correspond to low node dependency while high-pass datasets correspond to high node dependency[3].

MACHINE LEARNING TASK

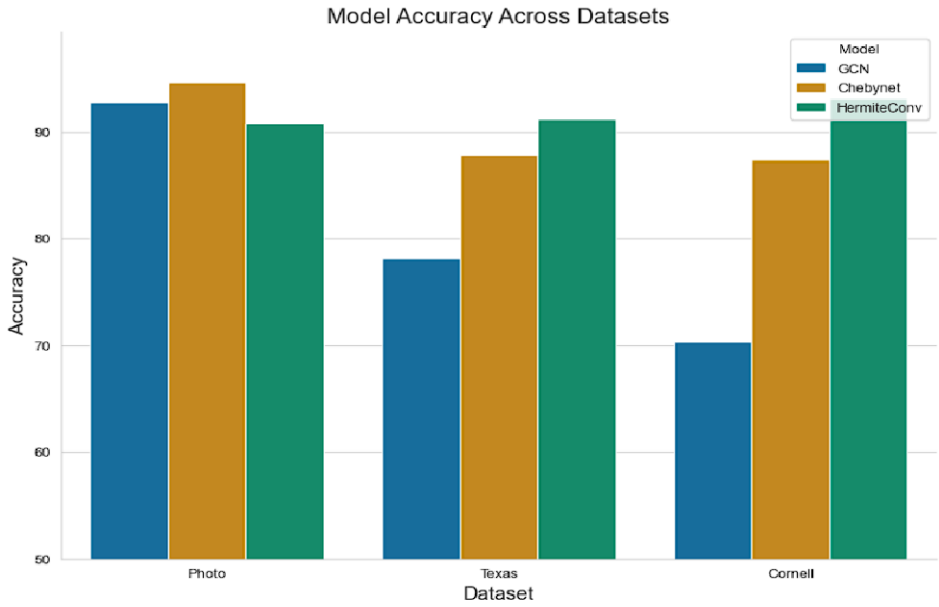
We are interested in Supervised node classification tasks. The approach for this study is inspired by [3]. For each datasets given node features are divided into training, testing and validation sets. The codes for existing models are chosen based on the latest advancements. All these models are trained on the same splits generated by random seed. Later compute average performance metrics over all different splits.

RESULTS

Dataset	Models				
	GCN	GAT	ChebyNet	APPNP	Hermiteconv
Cora	88.33 ± 0.89	89.18 ± 0.76	87.47 ± 0.72	89.52 ± 0.79	76.24 ± 1.48
CiteSeer	79.97 ± 0.67	88.12 ± 0.49	79.04 ± 0.91	80.70 ± 0.72	75.76 ± 0.99
PubMed	88.42 ± 0.34	88.12 ± 0.35	89.48 ± 0.23	87.56 ± 0.46	85.11 ± 0.43
Computers	84.93 ± 0.54	87.22 ± 0.77	88.97 ± 1.01	83.23 ± 0.27	85.02 ± 1.72
Photo	92.74 ± 0.39	91.43 ± 0.98	94.64 ± 0.87	91.96 ± 0.83	90.84 ± 0.37
Texas	78.20 ± 2.62	74.59 ± 3.93	87.87 ± 0.75	73.93 ± 3.44	91.15 ± 1.48
Cornell	70.33 ± 3.28	74.59 ± 4.43	87.40 ± 1.48	79.67 ± 2.13	93.11 ± 1.48

Checking the obtained accuracy above, it is valid to notice that Hermiteconv appears to perform better on high-pass graph datasets[1]. Since the convolution filter uses Hermite polynomials as a convolution operator, it involves intense calculations, which leads to more training on large graph data compared to benchmark models. Hyperparameter tuning has a great impact on accuracy of model which was not done under the research.

[1]Graph data where there is a higher-order dependency of nodes



Our proposed model, HermiteConv, demonstrates strong performance in datasets with complex connections. From the graph above, it is evident that in the Texas and Cornell datasets, HermiteConv outperformed all other base models. However, in the Photo dataset, HermiteConv performed slightly worse compared to other models, likely due to its intensive computational requirements during training. Overall, HermiteConv, utilizing Hermite polynomials to approximate the filter function, excels in capturing high-order dependencies in graph data.

3.4 SCOPE FOR FUTURE WORK

HermiteConv's success on high-pass graph datasets suggests promise for future exploration. Optimizing hyperparameters and improving computational efficiency are vital for wider applicability. Furthermore, investigating its performance on various graph types and integrating it with deep learning frameworks hold potential. Additionally, future work can address the observed accuracy loss in low-dependency datasets compared to other models. Here, exploring alternative filter approximations or leveraging hardware acceleration techniques could be effective ways to reduce time complexity on large graphs, particularly when dealing with

less intricate data. By delving into these areas, future research can solidify HermiteConv's contributions to the field of graph neural networks.

BIBLIOGRAPHY

[1] Mustapha Aouchiche and Pierre Hansen. Some properties of the distance laplacian eigenvalues of a graph. *Czechoslovak Mathematical Journal*, 64:751–761, 09 2014.

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs, 2014.

[3] Jiali Chen and Liwen Xu. Improved modeling and generalization capabilities of graph neural networks with legendre polynomials. *IEEE Access*, 11:63442–63450, 2023.

[4] Xinye Chen. Understanding spectral graph neural network. 12 2020.

[5] Michael Cohen and Can Tan. A polynomial approximation for arbitrary functions. *Applied Mathematics Letters*, 25, 08 2011.

[6] MichaÃ≪l Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. 2017.

[7] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018.

[8] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[9] P. Ramadevi and Tapobrata Sarkar. On link invariants and topological string amplitudes. *Nuclear Physics B*, 600:487–511, 04 2001.

[10] S. Sardellitti and S. Barbarossa. On the graph fourier transform for directed graphs. *IEEE Journal of Selected Topics in Signal Processing*, PP, 01 2016.

[11] Petar VeliÄkoviÄ, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro LiÃ^2, *andYoshuaBengio*. *Graphattentionnetworks*, 2018.

[12] Bingbing Xu, Huawei Shen, Qi Cao, Keting Cen, and Xueqi Cheng. Graph convolutional networks using heat kernel for semi-supervised learning, 2020.