Sheaf Neural Networks

Paul Jason Mello
Department of Computer Science and Engineering
University of Nevada, Reno
pmello@unr.edu

Abstract

A sheaf ties vector spaces together in a topological space with restriction maps that encode how information transforms and stays consistent across varying granularities of regions. More concretely, cellular sheaves extend Graph Neural Networks (GNNs) through generalization of the underlying mathematics. Formally, sheaves can be thought of as attaching local vector spaces to a graph's elements, with projections between these spaces that preserve the structural relationships of the data while maintaining the local independence of the sheaf. More eloquently, information flows across cellular sheaves like wind flowing over a field, which models the data manifolds through sheaf structures with independent vector spaces as stalks. Sheaf Neural Networks (SNNs) extend GNNs by restructuring the relationship between each node to handle dissimilarity and independence. My code for training multi-layer SNNs can be found at `https://github.com/pauljmello/SheafNN`.

"Nature uses only the longest threads to weave her patterns, so each small piece of her fabric reveals the organization of the entire tapestry."

- Richard Feynman [2]

1 Summary

In this work, we will cover significant ground as we endeavour to illuminate the mechanisms which guide SNNs which were initial developed by Hansen and Gebhart in "Sheaf Neural Networks" [4]. Sheaves have many fundamental properties which make them attractive extensions to graph networks, notably their convolutional operations and particularly their ability to express complex data manifolds through the stalk dimensionality which learns independently of the other sheaves in the cellular sheaf structure.

2 Introduction

GNNs, first introduced in 2005 by Gori et al. [3], have been a core training paradigm in artificial intelligence for the past two decades. Recently, they have started gaining significant attention for their generality and capabilities especially in graph data structures. Despite this, they suffer from two key problems, oversmoothing and the heterophily problem. Oversmoothing defines instances where stacking layers in graph based networks leads to node representations becoming very similar. The heterophily problem in GNNs isn't really a problem since GNNs have an homophily inductive bias, meaning it is more of a tradeoff of the architecture. GNNs have an inductive bias of homophily baked in where similar nodes will connect with other similar nodes. Heterophily defines the opposite tendency where dissimilar nodes tend to connect. In complex data environments, these heterophilic principles can prove critical for a model to learn.

SNNs, a concept from algebraic topology, offer solutions to these tradeoffs by extending GNN architectures with convolutional operations, information flow constraints, and independent local vector spaces for representation. To make this system work effectively, SNNs replace the core graph Laplacian with a sheaf Laplacian.

Like a field of sheaves, we will grow our understanding through the information we cover in this work spanning the sheaf data manifold. After patiently nurturing these grains, we will then be able to harvest our full understanding of SNNs.

3 Background and Motivation

As with most machine learning, SNNs were developed to turn cutting edge theories in physics and mathematics into a learnable kernel. However, they serve a broader purpose in improving graph architectures by solving oversmoothing and the heterophily problem. SNNs overcome these challenges by introducing local independence through their stalks and restriction maps, creating vector spaces that maintain feature distinctiveness while enabling complex directional relationships. This makes them particularly valuable for fields like drug discovery and materials science, where conventional graph networks often fail to capture the nuanced, multidimensional relationships between data points. This is a particularly vital difference between these architectures in that graph neural networks pass messages to similar nodes, while the SNN structure handles heterophilic settings better, leading to increased learned expressivity. SNNs offer a first principles approach to preserving both local structure and global patterns in the data.

4 Sheaf Neural Networks

SNNs improve on GNNs by introducing the cellular sheaves structure. This provides a highly expressive representation space for complex relational data. SNNs attach vector spaces or ”stalks” $F_v \in \mathbb{R}^d$ to each node $v \in V$ and $F_e \in \mathbb{R}^d$ to each edge $e \in E$. This geometric representation allows SNNs to model challenging datasets and solving the lack of heterophily in GNNs.

SNNs are built on the algebraic concept of sheaves with stalks being an essential component. This stalks are essentially localized representation spaces for each node and edge which encourages each stalk to learn diverse features. In a way, each stalk is simply a decoupled linear layer, and the SNN structure implements the inductive bias of heterophily. As we increase the dimensions of the stalks, we increase the feature manifold dimensions of that layer. From this perspective it is important to highlight that in standard feed forward MLPs information propagates globally all the time for each layer to learn together; However in SNNs, each sheaf only propagates its local information to its local neighbors based on the flow of data.

Then how does it pass its local information to its neighbors if each stalk is independent? Restriction maps offer a direction solution by utilizing linear transformations to map data from node stalks to edge stalks. Here we can imagine restriction maps as the bending of the stalk as information flows over it, almost like an activation function. These maps can be diagonal simple, orthogonal performant, or general flexible matrices.

The Sheaf Laplacian is the core component which defines how SNNs operate. They generalize the graph Laplacian of GNNs to the complex geometry encoded in the sheaf. We define this as $L_F = \delta^T\delta$, where $\delta\colon C^0(G,F) \to C^1(G,F)$ is the coboundary operator mapping from 0-cochains nodes to 1-cochains edges. For a given node v and edge e, we use the notation $v \triangleleft e$ to denote that node v is incident to edge e. Here, the Laplacian sums the differences in the representations between the connected nodes as:

$$L_F(x)_v := \sum_{e=(v,u)\in E} F_{v\triangleleft e}^T (F_{v\triangleleft e}x_v - F_{u\triangleleft e}x_u) \tag{1}$$

While the SNN structure is incredibly brilliant with many potential applications, Sheaf Diffusion [1] provides the mechanism for effective information propagation on the network through the Sheaf Laplacian. Utilizing differential equations, $\frac{\partial X(t)}{\partial t} = -\Delta_F X(t)$, where Δ_F is the normalized Sheaf Laplacian, Sheaf Diffusion determines how node features evolve over time. In a follow up paper titled Neural Sheaf Diffusion [1], the researchers identify Sheaf Diffusion to be particularly resistant to oversmoothing, as the harmonic space maintains dimensionality even at infinite depth. As this process evolves, the ”wind patterns” (information propagation) eventually converge to the harmonic space of the Laplacian - a state of perfect agreement with the structure of the sheaf. Additionally, they find that when the dimensions of the stalks are set to one, $d = 1$, we recover exactly the standard graph Laplacian: $\Delta_F|_{d=1} = \Delta_G$.

Overall, SNNs represent a significant step forward in graph modeling by abstracting graphs to cellular sheaves. Its complexity is grounded in elegant mathematics and the poetic structure of rolling waves of amber grain.

5 Methodology

After developing a brief theory around cellular sheaf theory, we can now define SNNs algorithmically as a sum of those components.

Algorithm 1 Sheaf Neural Networks

```
Require: Graph $G = (V, E)$, node features $X \in \mathbb{R}^{n \times f}$, stalk dimension $d$
Ensure: Node embeddings $Z$
 1: $H \leftarrow \phi(X)$                                        ▷ Project to $d$-dimensional stalk space
 2: for each edge $e = (u, v) \in E$ do
 3:     Compute $F_{u \triangleleft e}, F_{v \triangleleft e} \in \mathcal{O}(d)$          ▷ Orthogonal restriction maps
 4: end for
 5: Construct sheaf Laplacian $L_F$ where:
 6:     $L_F[v, v] \leftarrow \sum_{e \in E : v \in e} F_{v \triangleleft e}^T F_{v \triangleleft e}$
 7:     $L_F[v, u] \leftarrow -F_{v \triangleleft e}^T F_{u \triangleleft e}$   for edge $e$ connecting $v$ and $u$
 8: $D \leftarrow$ block diagonal of $L_F$
 9: $\Delta_F \leftarrow D^{-1/2} L_F D^{-1/2}$                          ▷ Normalized Sheaf Laplacian
10: $Z \leftarrow \sigma((I - \Delta_F)(I_n \otimes W_1) H W_2)$              ▷ Sheaf Diffusion
11: return $Z$
```

The SNN model employs several key components: $\phi(X)$ is an MLP that projects the original f-dimensional node features into a d-dimensional stalk space. σ is a non-linear activation function, while n is the number of nodes in the graph, I_n is the identity matrix used in the Kronecker product $(I_n \otimes W_1)$ to ensure node-independent feature transformations.

Our particular SNN uses orthogonal restriction maps $F_{u \triangleleft e}, F_{v \triangleleft e} \in O(d)$ that transport node features between stalks while preserving their geometric structure. These mappings comprise the Sheaf Laplacian L_F which is defined by the blocks for nodes connected by an edge e:

$$L_F[v, v] = \sum_{e \in E : v \in e} F_{v \triangleleft e}^T F_{v \triangleleft e} \quad \text{and} \quad L_F[v, u] = -F_{v \triangleleft e}^T F_{u \triangleleft e} \quad \text{for } e = (v, u) \in E \tag{2}$$

The normalized Sheaf Laplacian $\Delta_F \in \mathbb{R}^{nd \times nd}$ bounds our mdoel for stability during propagation.

The Sheaf Diffusion operator $(I - \alpha \Delta_F)$ represents a discretized time step of the continuous diffusion equation $\frac{\partial X(t)}{\partial t} = -\Delta_F X(t)$ which uses the implicit Euler method with an α step size, which propagates signals according to the topological structure encoded by the restriction maps. This operator is combined with learnable weight transformations $(I_n \otimes W_1) \in \mathbb{R}^{nd \times nd}$ and $W_2 \in \mathbb{R}^{d \times d}$ in the stalks and across each feature channel respectively. This yields the following propagation rule:

$$Z = \sigma\left((I - \alpha \Delta_F)(I_n \otimes W_1) H W_2\right) \tag{3}$$

This formulation generalizes conventional GNNs by replacing scalar edge weights with matrix-valued restriction maps that enable directional signal propagation and complex feature representations.

6 Experiments and Results

To test the effectiveness of SNNs, a synthetic network dataset was constructed consisting of 160 training nodes and 40 test nodes, with each node being randomly assigned to one of five class labels. These class labels are injected into a data sample that was sampled from a normal distribution. Each node is then randomly assigned to 2-3 other nodes in the network utilizing bidirectional edges. This dataset configuration creates a mixed homophily pattern containing roughly 500 edges.

The experiments hyperparameters were chosen from intuitions built over time. The SNN stalk dimensions are set to $d = 3$, and the input features dimensions are set to 4. The architecture utilizes a 2 layer deep SNN with ReLU activation functions for tried and true nonlinearity. Training for 250 epochs using AdamW optimization with learning rate $\eta = 0.01$ and weight decay coefficient $\lambda = 0.01$, provided strong results. For computational efficiency, the implementation employed batch processing with sizes of 64 for degree matrix computation and 256 for Sheaf Diffusion operations.

Evaluation of the SNN is done against a comparable Graph Convolutional Network (GCN) that performs neighborhood aggregation to match similar nodes based on feature information and edge-wise message passing. These additions to the GCN architecture are provided to give a better chance to GCNs on the homophily dataset.

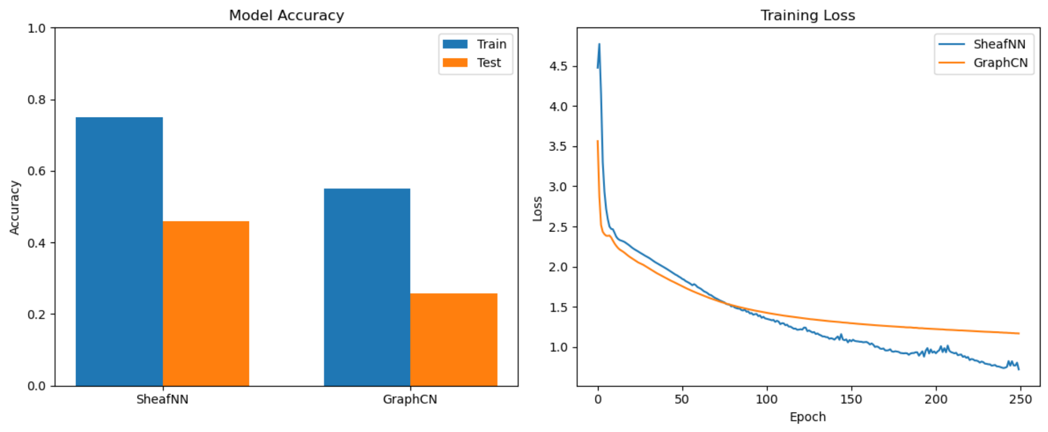

Figure 1: Performance comparison between SNNs and GCNs on a synthetic dataset with varying homophily levels. While the SNN is more computationally intensive to train, it outperforms GCN on training, test, and loss performance. Note, the GCN was augmented with improved functionality for the synthetic heterophilic dataset to give a comparable evaluation.

The results demonstrate that SNN outperform GCNs in our environment and demonstrate that SNNs can be an effective technique to model complex data.

7 Discussion

The beauty of SNNs lie in its generalization of the diffusion operation done in the graph convolutions networks. While GNNs operate with fixed sheaves consisting of identity restriction maps, SNNs, extend this to an n-dimensional vector space with non-linear mappings. This fundamental feature enables the SNN to function while handling complex data patterns. Specifically, the sheaves natural structure directly influence the networks ability to separate the nodes by class effectively.

By handling these cellular sheaf bundles through structured decision maps, SNN's enable a form of multi-channel transport along the graph. This fundamental change over GNNs extend the underlying framework to enhance the relationships and representations between nodes in the graph.

8 Conclusion

In this work, SNNs were expanded to the multi-layer setting with the applied Sheaf Diffusion strategy across the cellular sheaves and demonstrated that SNNs outperform GCN in a variety of ways.

These strategies were utilized, but were developed in prior works. These prior works include the Sheaf Laplacian, which generalizes the graph Laplacian through restriction mappings, and the Sheaf Diffusion, which guides information flow through the Sheaf Laplacian. These methods help to significantly boost the generalization capabilities through abstractions.

SNNs provide a very unique approach to modeling complex data. By extending and improving these approaches SNNs will excel in complex domains like drug discovery, materials science, social network analysis, financial systems, and knowledge graphs where heterophily and expressivity are key.

References

[1] Cristian Bodnar, Francesco Di Giovanni, Benjamin Paul Chamberlain, Pietro Liò, and Michael M. Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns, 2023.

[2] Richard P. Feynman. *The Character of Physical Law*. MIT Press, Cambridge, MA, 1965.

[3] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734 vol. 2, 2005.

[4] Jakob Hansen and Thomas Gebhart. Sheaf neural networks, 2020.