Hypergraph Learning vs. Graph Learning

Bushra Ibrahim
Illinois Institute of Technology
CS584: Machine Learning

December 6, 2023

Abstract

Existing GCN (Graph Convolutional Networks) and GNN (Graph Neural Networks) models have been popular in the machine learning field as of late. While they are efficient, the question becomes can they perform even better? Many researchers have proposed using hypergraphs as the base data structure in place of traditional graphs to achieve this, hypergraphs being the generalization of graphs. In this paper, we conduct vertex classification tasks with various datasets using a GCN model and an HGNN model, and we will analyze the results to determine if hypergraphs do indeed pose an advantage in machine learning as opposed to graphs.

1 Introduction

Traditional Graph Convolutional Networks (GCNs) and Graph Neural Networks (GNNs) have demonstrated significant success in machine learning tasks such as vertex classification and clustering by employing graph-structured data. However, these models are said to compromise some information due to the inherent limitations of graphs. Specifically, since graphs represent data in pairwise connections, they struggle to capture the full extent of complex and higher-order relationships, where not all connections are necessarily pairwise.

To address this, many researchers have proposed the use of hypergraphs, the generalization of graphs, as the base structure for neural networks, convolutional neural networks, and other machine learning methods in an attempt to represent richer and more intricate relationships between data points. Hypergraphs may be able to do this in a way that traditional graphs never could.

Observing the popularity of GCNs and GNNs in modern machine learning, the goal is to enhance these already effective methods by leveraging the hypergraph structure. Hypergraph Neural Networks (HGNNs) emerge as a promising solution, as they propose to operate with more speed, space, and efficiency while being computationally easier. Hypergraphs can fundamentally represent the same information a traditional graph can in less space.

In this paper, we conduct a comparative analysis between a GCN model and an HGNN model across various datasets. The objective is to ascertain whether the use of hypergraph structures is truly advantageous over traditional graph-based models or not.

1.1 Hypergraphs

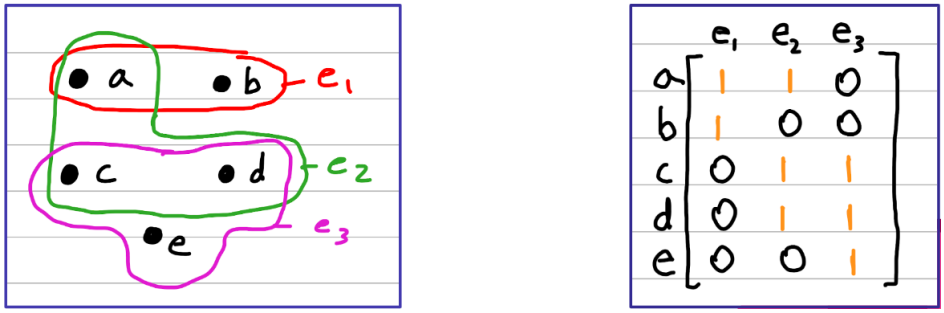
Let us review the concept of hypergraphs, a generalization of graphs, to increase understanding of this fundamental data structure that claims to have many benefits upon its use. This is crucial to understanding why these benefits exist and how HGNNs work.

A hypergraph is defined as $H = (V, E)$, consisting of Vertices, V; denoted as $(v_0, v_1, v_2, ..., v_n)$, representing individual data points or entities and Edges, E; denoted as $(e_0, e_1, e_2, ..., e_n)$, representing hyperedges which are subsets of V. Unlike edges in graphs that can connect only two nodes at a time, hyperedges can connect multiple vertices simultaneously. This is the key difference between graphs and hypergraphs.

The *order* of a hypergraph is the number of vertices, and the *size* of a hypergraph is the number of edges, just like traditional graphs. However, some complex connections that might require a lot of edges in a graph could be represented with just one edge in a hypergraph, and this is part of what contributes to hypergraphs' taking up less space and containing more information compared to graphs.

As previously mentioned, e denotes edges and v denotes vertices. An edge e is incident to a vertex v if $v \epsilon e$, that is, v is a vertex in hyperedge e which, again, is a subset of V, and therefore, a set of vertices. Similarly, vertex v is incident to an edge e if $e \epsilon v$, that is, e is a hyperedge that contains vertex v in its set.

These incidence relationships are what build a hypergraph's incidence matrix, which is one way of representing hypergraphs. Graphs also have incidence matrices which work in more or less the same way, save for the fact that each edge in a graph can only be incident to up to two vertices. Note the analogous features of graphs and hypergraphs – this is what makes hypergraphs a great replacement for graph structures because they already share many similarities in terms of representation and terminology.



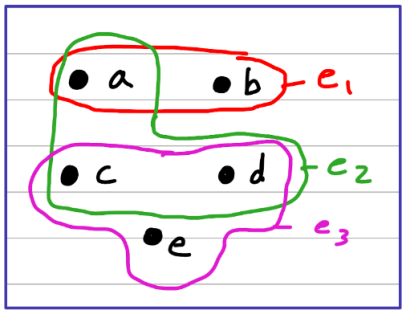
(a) A simple hypergraph, H

(b) The incidence matrix of H

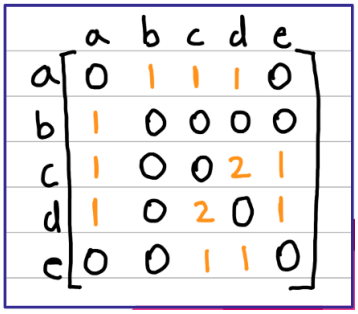
Figure 1: Here is an example of a simple hypergraph shown in (a) with vertex set $V = (a, b, c, d, e)$ and edge set $E = (e_1, e_2, e_3)$. For each hyperedge, $e_1 = (a, b)$, $e_2 = (a, c, d)$, and $e_3 = (c, d, e)$. The incidence matrix for this hypergraph is shown in (b).

The degree of a vertex v is the number of edges incident to that vertex. Similarly, the degree of an edge e is the number of vertices incident to that edge. Two vertices (v_1, v_2) are adjacent if there exists an edge e such that $(v_1, v_2) \epsilon e$, that is, there is a hyperedge that contains both vertices in question. Two edges (e_1, e_2) are adjacent if there exists a vertex v such that $v \epsilon e_1$ and $v \epsilon e_2$, that is, there is a vertex that exists in both hyperedges in question.

These adjacent relationships are what build a hypergraph's adjacency matrix, which is another way of representing hypergraphs. Graphs also have adjacency matrices which work in more or less the same way and are a popular form of representing graphs in code. Hypergraph adjacency matrices, like graph adjacency matrices, are also square and symmetric.



(a) A simple hypergraph, H



(b) The adjacency matrix of H

Figure 2: Here is an example of a simple hypergraph shown in (a) with vertex set $V = (a, b, c, d, e)$ and edge set $E = (e_1, e_2, e_3)$. For each hyperedge, $e_1 = (a, b)$, $e_2 = (a, c, d)$, and $e_3 = (c, d, e)$. The adjacency matrix for this hypergraph is shown in (b). Note that it is square and symmetric.

There are many more analogous features of hypergraphs to graphs, as well as more unique features such as *dual hypergraphs*. We refer you to the YouTube video *Introduction to Hypergraphs* [8] and the family of videos that follow for a more in-depth explanation of hypergraph theory. If reading is preferred, we refer you to the paper "Hypergraphs: an Introduction and Review" [6] for a literature review on hypergraphs and a thorough explanation.

1.2 Previous Works

There have been many previous works regarding this topic, especially explaining hypergraph theory as well as applications for hypergraphs in computer science as a whole. Of course, since hypergraphs are a generalization of graphs, trying to use them instead of graphs in existing graph applications is one of the more popular use cases and research topics we have found.

One of the most important works regarding hypergraph learning is "Learning with Hypergraphs: Clustering, Classification, and Embedding," [7] from 2006, by authors Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. It has proven to be a foundational article that builds the frameworks and mathematics for clustering, classification, and embedding with hypergraphs in place of graphs. It also uses the hypergraph Laplacian that, while varied definitions exist, this paper seems to have the most universally agreed upon definition and usage, primarily in clustering and spectral clustering methods. Numerous following works on this topic refer to this article due to its ground-up establishing of hypergraph learning methodologies and definitions that apply even to this day.

One such work that refers to the above article is the 2016 study "Hypergraph Exploitation for Data Sciences," [9], where presenters Michael Wolf, Alicia Klinvex, and Daniel Dunlavy explain and perform experiments using hypergraph learning for spectral clustering. Their results showed promising advantages. When evaluating clusters generated by a graph base against clusters generated by a hypergraph base, the hypergraph-based clusters were shown to be much closer to the ground truth than the graph-based clusters, measured by the Jaccard Index. The runtimes were also significantly different, with the hypergraph models running almost thirty times faster in some cases.

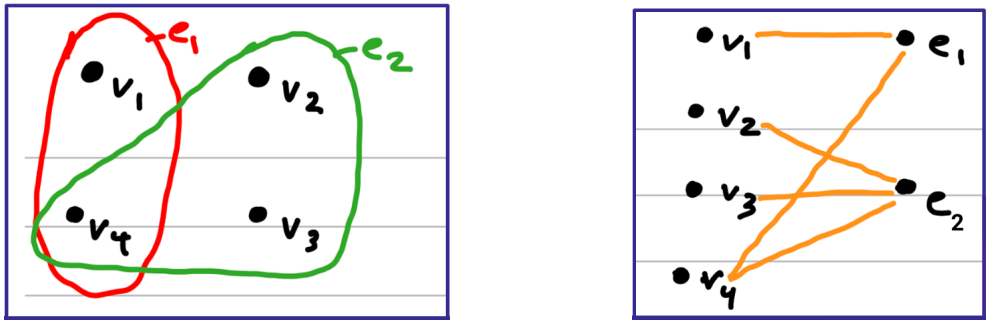
Two more works that refer to the first article [7] are used in this paper, specifically the HGNN models that they produced. The first work from 2019 is "Hypergraph Neural Networks," [3], where the authors propose a neural network framework that is hypergraph-based and discuss its potential. They define a fresh hypergraph convolution concept that is what makes their framework analogous to graph neural and convolutional networks but with the added benefits of hypergraphs.

The second work from 2022, "HGNN+: General Hypergraph Neural Networks," [4] has some of the same authors and builds off of the existing hypergraph neural networks (HGNNs) to develop a more robust HGNN model called HGNN+ or HGNNP. HGNN+ can to handle even more multifaceted and higher-order data, again, by employing the hyperedge functionality of hypergraphs.

These two works, "Hypergraph Neural Networks," [3] and "HGNN+: General Hypergraph Neural Networks," [4] and their source code are used in this paper. Namely, the HGNN model from the first work and the HGNN+ model from the second work are employed to see what kind of results in vertex classification a hypergraph-based model can produce.

2 Problem

Graph-based models such as GCN (Graph Convolutional Networks) and GNN (Graph Neural Networks) have exhibited great performance in various machine learning tasks such as vertex classification and clustering. However, the base structure being the traditional graph comes with inherent limitations since graphs represent pairwise information but struggle to do so with multi-modal and higher-order data that may not fit into a pairwise framework. Also, the same data represented as a hypergraph takes up less memory and storage than when represented as a graph. Due to these many areas for improvement, researchers see potential in hypergraph-based models to improve on current graph-based machine learning.



(a) A simple hypergraph, H

(b) The incidence graph of H

Figure 3: Here is an example of a simple hypergraph shown in (a) with vertex set $V = (v_1, v_2, v_3, v_4)$ and edge set $E = (e_1, e_2)$. For each hyperedge, $e_1 = (v_1, v_4)$ and $e_2 = (v_2, v_3, v_4)$. The incidence graph for this hypergraph is shown in (b). Note that it has six vertices and five edges, compared to four vertices and two edges in H. This is a small example to show that the same data represented as a hypergraph takes less storage than as a graph.

In this paper, while we do not address the problem directly, we take a look at recently developed hypergraph-based models and pit them against a graph-based model to see if any of these claims hold. This study is by no means all-encompassing or conclusive. In fact, more research and experiments are encouraged with other graph and hypergraph models to get a more well-rounded view of the problem and its potential solutions.

2.1 Examples

Real-world data is often never truly pairwise, or at least, while they may be able to be represented in a pairwise fashion, that may not be the most efficient manner. Let us see a few examples and see

how graphs may limit the analysis of such data and how hypergraphs might perform better.

In a network of authors and publications, a graph representation would only be able to show each author and their work once at a time with one edge, one vertex being the author and the other being the work. For publications with multiple authors or authors that have worked on multiple publications, this would take a large amount of vertices and edges to represent. However, if the data is represented as a hypergraph, multiple authors who worked on a publication can all be encompassed under one hyperedge, and multiple works with the same author(s) can be encompassed in a hyperedge as well. By covering these relationships in a smaller number of edges than a graph representation would take, storage and memory are saved.

For another example, consider food chains and hierarchies in the environment. If a certain ecosystem, namely predators and prey, are represented simply species-wise in a graph representation, again, it would take a large number of vertices and edges to do so accurately. However, a hypergraph could represent an ecosystem by grouping predators and prey and their relationships using the unique quality of hyperedges, and accurately so with a smaller number of vertices and edges.

To see a more thorough explanation of these examples in practice, we refer you to the paper "Complex Networks as Hypergraphs," [2]. The authors conduct experiments with an author-work dataset and a species-relation dataset showing promising results when comparing a graph model to a hypergraph model.

3 Methodology

To test a graph-based model against a hypergraph-based model, the methodology used is fairly straightforward. Conduct a vertex classification task on some data with a graph model, do the same with a hypergraph model, and then compare and analyze the results. This is not the only way to go about it, there are many other ways this idea can be tested and many other experiments that can be conducted. However, with the limited time and scope of this paper, this is the extent to which testing was done. We recommend those interested to replicate these experiments themselves and build upon them to get even more conclusive results that may provide further insight into the question of whether hypergraph-based models are truly better than graph-based ones.

3.1 Models and Datasets Used

For our graph-based model, a GCN model sourced from the 2017 paper "Semi-Supervised Classification with Graph Convolutional Networks," [5] by Thomas Kipf and Max Welling is used. This model is designed to work directly with graphs, and it utilizes a novel method of semi-supervised classification by simplifying a typical spectral convolution on graphs and propagating that information throughout layers in the neural network. The authors claim that this model performs better than other existing GCN models and is able to do so while being computationally efficient. For these reasons, this model was chosen as a good graph-based model to compare to.

For our hypergraph-based model, two sources have been chosen. The first is "Hypergraph Neural Networks," [3] from 2019, which provides an HGNN framework and model to use for machine learning tasks. This model proposes to use hypergraphs which are more flexible and able to represent intricate connections within data more easily than graphs. The key computation used in this work is called a *hyperedgeconvolution*, which is tailored to manage such complex relationships. The authors tested the model on classification with citation network data as well as visual object recognition. In both cases, their results demonstrate the advantages of hypergraphs empirically.

The second source for our hypergraph-based model is from some of the same authors as the HGNN model, but from 2022, and as such, claims to be an updated version of and built from the prior

Table 1: Datasets Used

Description	Cora	PubMed	Citeseer
Nodes	2708	19717	3327
Edges	10858	88676	9464
Feature	1433	500	3703
Classes	7	3	6

HGNN model. "HGNN+: General Hypergraph Neural Networks," [4] proposes a HGNN+ or HGNNP model that extends the initial HGNN model and is designed to specifically handle higher-order correlations in multi-modal and multi-type data. Complex relationships are first captured with hyperedge groupings, and then they are merged using an adaptive fusion strategy. Building on alternate convolutions, the authors introduce a new spatial-domain hypergraph convolution scheme that is used in this model. HGNN+ was used and compared with several of the latest methods, and it was shown to outperform these methods.

In both works, the use of hypergraph structure and a novel convolution method was well explained and justified, not to mention their results were promising. As such, these are the two works from which we chose to source the hypergraph models from, namely HGNN and HGNN+. HGNN and HGNN+ will be compared to the GCN model mentioned earlier on vertex classification tasks on three different citation network datasets; Cora, PubMed, and Citeseer. These three datasets are well-documented and commonly used for testing models such as these, which is why they were chosen. They are also referred to in the HGNN, HGNN+, and GCN papers, making them a good dataset to use since these models are already familiar with them. See Table 1 for details about each of these three datasets.

The information about the datasets themselves, documentation on the models and source code, tutorials on how to use these models, and more were provided by the authors on GitHub at this link: `https://github.com/iMoonLab/DeepHypergraph`[1]. The creators combined their works and other referred works to create a new framework called *DHG*, which stands for Deep Hypergraph. It is a deep learning library that is open for public use, specifically designed for handling GNNs (Graph Neural Networks) and HGNNs (Hypergraph Neural Networks). With this library package installed, running and testing both types of models was made much easier. All instructions on installation, use, and more are available on their GitHub as mentioned before, as well as within their documentation. We encourage readers to explore both GitHub and the documentation to get a better understanding of HGNNs practically, as well as to replicate or build upon this paper's work if desired.

4 Experiment

The experiment explained in the methodology was also carried out in a straightforward manner with the limited time, scope, and resources at hand. First, the *DHG* package was installed and tested. *DHG* is capable of producing visualizations as well as deep learning with graphs and hypergraphs as it claims, so these were tested with simple lines of code to make sure the installation was successful.

Then, three source code files were created. One for the GCN model, one for the HGNN model, and one for the HGNN+ model. In each respective file, the correct model was imported from the *DHG* library, and the neural network was set up, following the training section. The source code was also modified a bit to capture the elapsed time that each model would take to produce results, since speed was another key quality we wanted to compare. Accuracy was already part of the model in the

Table 2: Results (Average Accuracies and Runtimes)

Dataset	Measure	GCN	HGNN	HGNN+
Cora	Accuracy	0.816	0.824	0.807
	Runtime(s)	0.022	0.037	0.020
PubMed	Accuracy	0.783	0.789	0.786
	Runtime(s)	0.083	0.562	0.263
Citeseer	Accuracy	0.714	0.714	0.709
	Runtime(s)	0.070	0.073	0.082

Evaluator function, which is a key function to show how accurate the model was based on the raw accuracy of the predictions, as well as the F1-score, which while interesting, is not discussed in this paper.

Then, each dataset was imported as needed and each model was run 5 times on that dataset. Each of the 5 times, the speed and accuracy were noted, and after 5 runs were completed, they were averaged.

4.1 Results

Once the code was prepared, the dataset was imported. First, the Cora dataset was used and tested. The model was run 5 times with 500 epochs and the accuracy and speed were averaged. For the Cora dataset, the GCN model had 81.6% average accuracy with a 0.022 second average runtime. The HGNN model had 82.4% average accuracy with a 0.037 second average runtime. The HGNN+ model had 80.7% average accuracy with a 0.020 second average runtime.

Next, the PubMed dataset was imported and tested. The model was run 5 times with 150 epochs (reduced due to the size of the dataset) and the accuracy and speed were averaged. For the PubMed dataset, the GCN model had 78.3% average accuracy with a 0.083 second average runtime. The HGNN model had 78.9% average accuracy with a 0.562 second average runtime. The HGNN+ model had 78.6% average accuracy with a 0.263 second average runtime.

Lastly, the Citeseer dataset was imported and tested. The model was run 5 times with 500 epochs and the accuracy and speed were averaged. For the Citeseer dataset, the GCN model had 71.4% average accuracy with a 0.070 second average runtime. The HGNN model had 71.4% average accuracy with a 0.073 second average runtime. The HGNN+ model had 70.9% average accuracy with a 0.082 second average runtime.

For a better look at these results, take a look at Table 2 where they are displayed more neatly.

4.2 Analysis

Upon viewing the results, it seems that the graph/hypergraph comparison is not quite so easy to distinguish. First impressions show that while the HGNN model performed better than the GCN model for the Cora and PubMed datasets, the increase in accuracy is almost negligible, even more so when considering it took longer to run on average. For Citeseer, GCN and HGNN had identical accuracies, which begs the question of whether the HGNN model is truly better than the GCN model.

HGNN+ only performed better than the GCN model with the PubMed dataset, and that, only by 0.3%. With the other two datasets, HGNN+ performed worse than GCN and HGNN. We suspect this is due to HGNN+ being more equipped to handle more vast and higher-order data, and perhaps

the chosen datasets were not well-suited for HGNN+. Perhaps that is also why HGNN+ performed a bit better on the largest dataset used, PubMed, as opposed to the other ones which are relatively smaller.

Another possibility is that the chosen GCN model has been proposed to be better than its peers at the time it was created. Perhaps it being a better GCN model in general contributed to its higher accuracy and faster runtime, since the creators of this model claimed that it was more computationally efficient anyhow.

There is a lot to speculate here due to these inconclusive results that do not demonstrate explicitly that hypergraph-based models perform better than graph-based models. Based on various other papers, works, and studies, there have been many stronger results to suggest that hypergraph models are better, but with the ones chosen for this paper and the methodology used, this idea is not as well supported. We suspect that if a better methodology was used with data more tailored to each model, with more measures being observed in addition to accuracy and speed, and with models currently in use in the real world rather than recently developed ones by researchers, more explanatory results may be observed. More information and experimentation are needed for more conclusive results.

5 Conclusion

Motivated by the desire to enhance machine learning models, an investigation into Graph Convolutional Networks (GCNs) versus Hypergraph Neural Networks (HGNN and HGNN+) was undertaken. This exploration aimed to determine if using hypergraphs could offer a significant advantage over traditional graph-based models in vertex classification tasks. However, the outcomes were inconclusive. While the hypergraph-based models showcased superior performance in some cases, in others they either matched or underperformed compared to the GCN model in terms of accuracy and runtime. The variability in performance across the Cora, PubMed, and Citeseer datasets suggests that more deep analysis and testing are necessary to extract meaningful results. This will foster a better understanding of the real-world applications and potential superiority of hypergraph-based models, paving the way for more informed and effective usage of these neural network structures in future machine learning endeavors.

References

[1] *Deep Hypergraph*. 2022. URL: https://github.com/iMoonLab/DeepHypergraph.

[2] Ernesto Estrada and Juan A. Rodríguez-Velázquez. "Complex Networks as Hypergraphs". In: *Physica A: Statistical Mechanics and its Applications* 364 (May 2006), pp. 581–594. ISSN: 0378-4371. DOI: 10.1016/j.physa.2005.12.002. URL: http://dx.doi.org/10.1016/j.physa.2005.12.002.

[3] Yifan Feng et al. "Hypergraph neural networks". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 3558–3565. URL: https://github.com/iMoonLab/DeepHypergraph.

[4] Yue Gao et al. "HGNN: General Hypergraph Neural Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022). URL: https://github.com/iMoonLab/DeepHypergraph.

[5] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *CoRR* abs/1609.02907 (2016). arXiv: 1609.02907. URL: http://arxiv.org/abs/1609.02907.

[6] Xavier Ouvrard. "Hypergraphs: an introduction and review". In: (2020). arXiv: 2002.05014 [cs.DM].

[7] Bernhard Schölkopf, John Platt, and Thomas Hofmann. "Learning with Hypergraphs: Clustering, Classification, and Embedding". In: *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*. 2007, pp. 1601–1608.

[8] Vital Sine. *Introduction to Hypergraphs [Graph Theory]*. Apr. 2022. URL: https://www.youtube.com/watch?v=UwvZn9lm_98&list=PLZ2xtht8y2-IRjvXJJpka2GIJ3wn7u4g2.

[9] Michael Wolf, Daniel Dunlavy, and Alicia Marie Klinvex. "Hypergraph Exploitation for Data Sciences." In: (May 2016). URL: https://www.osti.gov/biblio/1367108.