Machine Learning Applications with Graph Signal Processing

John Shi, Jade Traiger, Carlos Taveras, Michael Kwan
MLSP 2021 Final Project Report (Draft), Group 14

Abstract

Traditional signal processing and machine learning (ML) have achieved state-of-the-art performance on grid-based data such as sounds and images. However, more and more data is defined on irregular grid structures / graphs. Traditional methods do not perform well on graph data. Graph signal processing (GSP) is an emerging field that extends DSP techniques for use on graph data. In this project, we propose using GSP techniques and traditional ML to solve graph problems.

1 Introduction

With the advent of “Big Data," more and more data is defined on an irregular non-grid-based structure. While traditional signal processing and ML methods perform well on grid-based data such as sounds and images, they do not perform well on data defined on graphs. One main reason for this is that unlike images and time series, graphs do not have a fixed ordering. This makes it difficult to represent graphs as vectors because vectors have a fixed ordering. Arbitrarily assigning graph nodes to vectors leads to comparison between graph nodes in each graph that may or may not correspond with each other, especially with graphs of different sizes. Graph classification is given a set of graphs with features on each node for each graph, classify each *graph*. For example, we might have a set of chemicals represented by graphs. Nodes represent atoms (e.g., hydrogen, carbon, oxygen) and edges represent chemical bonds between the atoms. Instead of predicting the class of each chemical atom, we would classify each graph as a chemical [1]. The goal of our project is to explore how to incorporate GSP theory and graphs into traditional ML techniques to achieve better accuracy on graph classification problems. We look for ways to solve the ordering issue with graphs, not present with images and time series. We examine the advantages of using a graph structure over not using one. We focus on ways to add graph data to existing ML algorithms and how to use GSP theory to develop new ML algorithms.

2 Literature Review

In DSP, consider a N node cycle graph with adjacency matrix A. The graph signal x is the $N \times 1$ vector formed by assigning a number to each node. A time shift of x is Ax. GSP extends DSP by assuming A is the adjacency matrix of an *arbitrary graph* [2]. The graph signal x is formed by assigning a number to each node. A shift in GSP is Ax. Convolution in the vertex domain is defined as $P(A)x$. The Graph Fourier Transform (GFT) is found by eigendecomposition, $A = GFT^{-1}\Lambda GFT$. The GFT forms a basis for the graph spectral domain, similar to the DSP frequency domain, but considering graph structure. The signal in the spectral domain is $\widehat{x} = GFTx$. Two other shifts have been found in GSP. One is the spectral (frequency) domain shift ($M = GFT\Lambda^* GFT^{-1}$). The other is the companion matrix of A ($C = V^{-1}\Lambda V$) where V is the Vandermonde matrix of the eigenvalues.

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

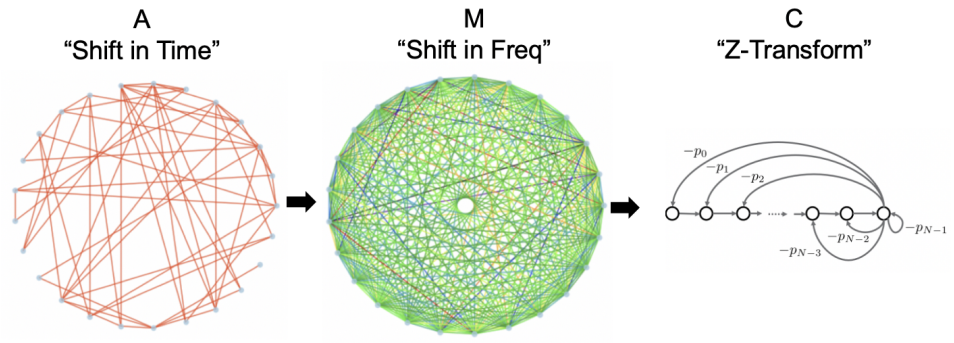

Figure 1: An example graph of A, M and C in GSP. The ordering affects A and M, but does not affect C.

C represents the GSP “z-transform." A property of C is that it is the same regardless of the ordering of nodes or the GFT matrix, depending only on the characteristic polynomial of A [3]. An example of A, M and C are shown in Fig. 1.

GSP has been applied to many graph problems including neuroscience, social networks, motion estimation [4] and COVID modeling [5]. A common method is to use the GFT to do a spectral analysis on the given graph data, similar to frequency analysis in DSP. This takes into account the underlying graph structure while traditional DSP does not.

The ordering issue with graphs when doing graph classification has been addressed in many different ways. Graph Convolutional Neural Networks (GCNNs) take the average of the nodal values before the fully connected layer, reducing the vector of values down to a single value [1]. This average does not depend on order. So, GCNNs avoid the ordering issue. Another approach is graph embedding where graphs are transformed to a Euclidian grid structure before traditional CNNs are used [6]. However, this method loses graph information that was present in the graph, but not the Euclidian grid. This can also be done using graph random walks or neural networks.

3 Datasets

We consider both synthetic and real world datasets. In all the datasets, the input is a set of graphs (expressed as adjacency matrices A) and data defined on graphs. The output is predicting the class of each graph. Preprocessing is needed to find the M and C matrices for each graph.

For the synthetic dataset, we generated graphs using Erdős-Rényi with different numbers of nodes: $N = 5, 25, 50, 75, 100$ using probability of edge formation, $p = 0.3$. The number of nodes varied slightly based on a normal distribution with mean N and standard deviation $0.1N$. Our goal was to predict whether a given graph contained a node of a certain degree or higher. To determine the degree threshold given N, we observed that each node has a binomial distribution binomial(N, p). Because we are normalizing N, we need to normalize how the threshold is chosen. We define the degree threshold, d_t as the inverse CDF of the binomial with parameters N,p at the point 0.65. Using MATLAB, we found the degree thresholds for $N = 5, 25, 50, 75, 100$ to be $d_t = 2, 8, 16, 24, 32$ respectively.

For each N and threshold, we generated 500 connected graphs with a node of at least degree d_t and 500 connected graphs with no nodes with degree greater than or equal to d_t. We classify each graph as “1" if it has a degree of d_t or higher. We classify each graph as “0" if it does not. Although this problem can be solved in polynomial time using BFS, DFS, we chose this synthetic dataset to see how the ordering of the graph could affect the classification and whether our method would do well on a simpler problem.

Dataset	Size	Classes	Avg. # of nodes	Avg. # of edges
MUTAG	188	2	17.9	19.8
NCL1	4110	2	29.87	32.30
AIDS	2000	2	15.69	16.20
IMDB-BINARY	1000	2	19.8	96.5
DBLP_v1	19456	2	10.48	19.65

Figure 2: The statistics of the real world datasets. MUTAG, NCl1 and AIDS are chemical datasets. DBLPv1 and IMDB-Binary are citation network and relationship graphs respectively.

For real world datasets, we examine graph classification datasets. MUTAG is a dataset where each chemical is represented by a graph with nodes being the atoms and edges being the bonds. The goal is to predict if given chemicals are mutagenic. AIDS is a dataset of molecular compounds where the goal is to predict which compounds have possible activity against HIV. NCl1 is another molecular compound dataset to predict whether a chemical compound reacts to cancer or not. DBLPv1 [7] is a Computer Science co-authorship network dataset. The goal is to predict whether a paper belongs to the DBDM (database and data mining) or CVPR (computer vision and pattern recognition) field. IMDB-Binary is a dataset of movies where each node in the graph is an actor or actress. Edges represent working together on a film [8]. Statistics for the real datasets can be found in Fig. 2.

4 Evaluation Metrics

For each graph problem, we compared our method to the state-of-the-art graph convolutional neural network (deep learning) accuracy from the literature [1]. The graph convolution neural networks take the average of the feature vector across all nodes to produce a single value in order to avoid the ordering problem.

5 Experiments and Results

5.1 BFS Ordering

We also considered the effect of reordering of the nodes on accuracy using A and M. We reordered the graph using a BFS traversal. We started from the node with the largest degree and visited the neighbors in order of descending degree. By this way, the nodes of larger degree are closer to first and the nodes of smaller degree are closer to last. Fig. 3 shows an example of the BFS ordering.

5.2 Experiment Procedure

We have preprocessed several datasets, producing vector representations of the adjacency matrix A, spectral graph shift M and the matrix C. We then ran these datasets using SVM, logistic regression and random forest to classify the graphs for both the BFS and non-BFS orderings. We varied the type of kernel in SVM, L2 penalty in logistic regression, the number of estimators for the random forest. All code was run in Python using the Network-X and Sci-Kit Learn packages.

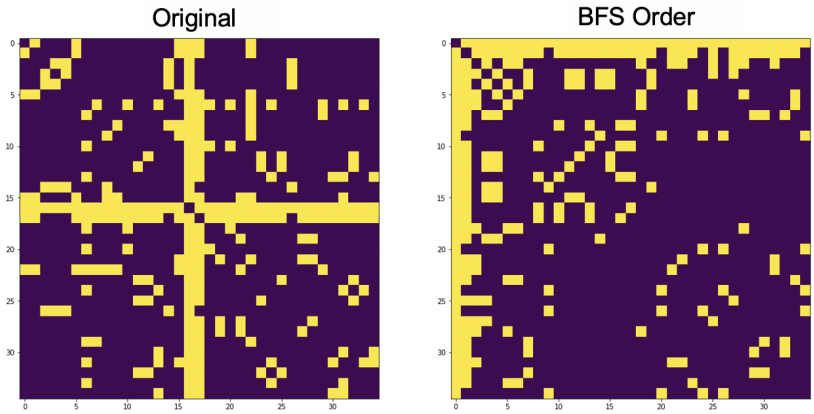

Figure 3: An example of the BFS ordering for an adjacency matrix. The nodes are in descending order by degree.

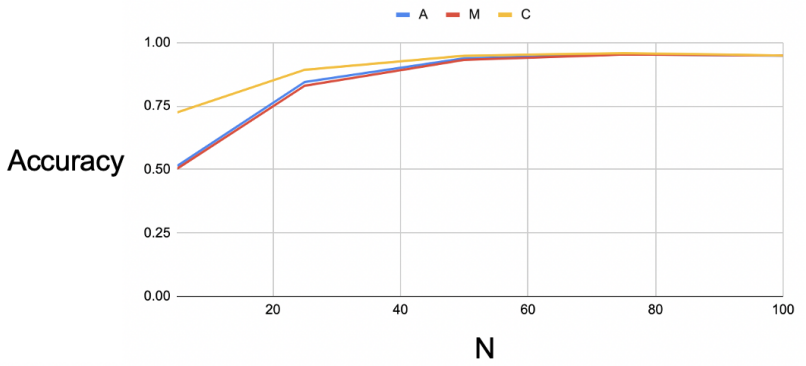

Figure 4: The accuracies for A, M, C for the synthetic dataset for $N = 5, 25, 50, 75, 100$.

5.3 Results

5.3.1 Synthetic Dataset Results

The accuracies for A, M and C for the synthetic dataset are given in Fig. 4. For the synthetic dataset, A and M perform similarly, while C performs considerably better for smaller N and similarly for larger N. Smaller graphs are more impacted by permutations because are fewer values in each vector to use for classification. Because of this, C, which is ordering invariant, performs better on smaller datasets. For larger N, one has more values in each vector to consider (when checking if a node of a certain degree exists), so the ordering matters less. For C, the ordering does not affect it, so we see it have higher overall accuracy than A and M. For larger N, the accuracies for A, C and M are similar because ordering issues matter less for these problems. This also explains why the accuracy increases in general when N increases.

5.3.2 Real World Dataset Results

The accuracies for A, M and C and baseline graph convolutional neural network accuracy from the literature (if any) for the real datasets are given in Fig. 5. We found that the RBF kernel for SVM worked best, with other built-in kernels performing significantly worse. For the logistic regression, the L2 penalty was varied from 0.01,0.1,1,10,100. For the random forest, 1,10,25,50,100 estimators were used. The highest values from these runs are shown in this table.

For SVM, Logistic Regression and Random Forest, C performed better than or similar to A and M on chemical datasets (NCI1, MUTAG, AIDS). However, it performed worse for DBLPv1 and IMDB-

		SVM			Logistic Regression			Random Tree		Baseline
Dataset	A	M	C	A	M	C	A	M	C	GCNN Acc
DBLP_v1 (No BFS)	**0.7503**	0.7423	0.617	**0.76**	0.7589	0.7567	**0.7611**	0.73	0.657	-
DBLP_v1 (BFS)	**0.772**	0.733	0.617	**0.7722**	0.7555	0.7567	**0.7811**	0.723	0.657	-
IMDB-Binary (No BFS)	**0.6559**	0.5962	0.5888	**0.654**	0.609	0.6298	**0.6967**	0.671	0.669	0.71
IMDB-Binary (BFS)	**0.6705**	0.5661	0.5888	**0.6967**	0.6713	0.6298	**0.669**	0.645	0.669	0.71
AIDS (No BFS)	0.993	**0.995**	0.9891	0.9958	**0.9976**	0.9934	0.9928	**0.996**	0.993	0.99
AIDS (BFS)	0.994	**0.994**	0.9891	0.9928	**0.9958**	0.9934	0.9922	**0.995**	0.993	0.99
NCI1 (No BFS)	0.6121	0.5956	**0.6237**	0.5888	0.5999	**0.6301**	0.5592	0.59	**0.599**	0.728
NCI1 (BFS)	0.6014	0.5891	**0.6237**	0.6486	0.6194	**0.6301**	**0.6097**	0.578	0.599	0.728
MUTAG (No BFS)	0.8561	0.807	**0.8631**	0.8246	0.8596	**0.8772**	0.8187	0.813	**0.877**	0.86
MUTAG (BFS)	0.6596	0.6631	**0.8631**	0.8246	0.8596	**0.8772**	0.8012	0.836	**0.86**	0.86

Figure 5: The accuracies for A, M, C for real datasets and the graph convolutional neural network benchmark. For the logistic regression, the L2 penalty was varied from 0.01,0.1,1,10,100. For the random forest, 1,10,25,50,100 estimators were used. The highest values from these runs are shown in this table.

Binary. One reason for this is that chemical datasets rely mainly on the chemical / graph structure and less on the graph signal (an enumeration of the type of atom at each node) to distinguish between chemicals. Using C depends on only the characteristic polynomial, so it is easier to distinguish between different graph structures than it would be with the A, due to the ordering issue. On the other hand, DBLPv1 and IMDB-Binary are datasets where both the graph and graph signal (data at each node) matter. The graph signal contains a lot of information such as an actors' or a paper's features. The graph structure on its own contains less information and may look similar between the two classes. Thus, it would be difficult to distinguish between classes using C, which uses the characteristic polynomial. A and M retain eigenvector information as well, which can help distinguish between graphs, despite the ordering issues. This may cause higher accuracy for the A and the M over C. In addition, we do not perform as well using our method when compared to graph convolutional neural networks. This is probably because GCNNs incorporate both the graph structure and the graph data, while we only use the graph structure in our approach. For MUTAG, a chemical dataset where the graph data matters less, we obtain similar accuracies to GCNNs.

There is not a significant difference between the BFS accuracies and the non-BFS accuracies across the datasets. BFS ordering assumes that the highest degree nodes of a graph should be compared with the highest degree nodes of another graph and the lowest degree nodes should be compared with the lowest. Since there is not a significant difference in accuracy, this suggests that our assumption of comparing similar degree nodes to produce an ordering is not significantly better than using the given, original arbitrary ordering of the datasets.

6 Conclusion

In this paper, we used GSP to generate M and C graphs from A. The C graph had the advantage that it was always the same regardless of ordering. However, C only depends on the characteristic polynomial / eigenvalues of A and not the eigenvectors of M. This makes it easier to distinguish graphs in chemical datasets where the graph structure matters more than the graph signal (nodal data) and ordering correctly matters more. On the other hand, in datasets like DBLPv1 and IMDB-Binary, the graph signal distinguishes between classes more than the graph structure. Thus, C performs worse than A and M because C does not consider the eigenvectors of A. This means graphs that are similar in structure and eigenvalues are harder to distinguish with C. In the future, we will look into ways to incorporate the graph signal into the ML algorithms when using M and C to improve the accuracy. We will also look for algorithms to order these graph signals. By using both the graph signal and the M and the C, we believe the accuracy can be improved further.

References

[1] M. Cheung, J. Shi, O. Wright, L. Jiang, X. Liu, and J. M. F. Moura, "Graph signal processing and deep learning," *IEEE Signal Processing Magazine*, vol. 37, Nov 2020.

[2] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Proc.*, vol. 61, pp. 1644–1656, Apr 2013.

[3] J. Shi and J. M. F. Moura, "Graph signal processing: A signal representation approach to convolution and sampling," Mar 2021.

[4] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, pp. 808–828, May 2018.

[5] A. Kapoor, X. Ben, L. Liu, B. Perozzi, M. Barnes, M. Blais, and S. O'Banion, "Examining covid-19 forecasting using spatio-temporal graph neural networks," 2020.

[6] M. Xu, "Understanding graph embedding methods and their applications," *Siam Review*, vol. 63, no. 4, p. 825–853, 2021.

[7] C. Z. P. Y. S. Pan, X. Zhu, "Graph stream classification using labeled and unlabeled graphs," *International Conference on Data Engineering (ICDE)*, pp. 398–409, 2013.

[8] K. Kersting, N. M. Kriege, C. Morris, P. Mutzel, and M. Neumann, "Benchmark data sets for graph kernels," 2016.