ORIE 4741 Final Project - Optimizing Graph Based Semi-Supervised Learning

Introduction

In the real world, data often is noisy and may contain any number of missing values and labels. In our investigation, we specifically targeted the problem of how to leverage datasets with limited labels. One approach that was very intriguing to use was to represent the data as graphs and use properties of the generated graph to determine labels for the unlabeled points. The leading question in our investigation was how graphs could help in machine learning with datasets with limited labeling and what approaches to graph based semi-supervised learning are most effective.

More formally, our focus was on reducing/optimizing classification error rate for classification problems in sparsely distributed graphs using graph based semi supervised learning. Due to the presence of limited labelled data in the real world, semi-supervised learning approaches are an efficient way to compound unsupervised learning methods with available data and use them to the best potential. The various models we included were based on datasets corresponding to studies about breast cancer, happiness index in a given town and national flags, all cited below.

Background Research

Like all tasks in machine learning, there are certain assumptions required in order to support the feasibility and logic of the approaches/algorithms developed. In the case of graph based semi-supervised learning, some assumptions were made regarding how a graph can depict relationships between points. First, we assumed that the graphs constructed would all use weighted edges and that points connected by edges of larger weights would be more similar. This would consequently make them likelier to have the same label.

We began our research with a general introduction to graph-based semi-supervised learning, which would familiarize us with the process and various different approaches. Most of this introductory material came from Graph-based Semi-supervised Learning: A Comprehensive Review[7].

There are two stages to consider in the use of graph based semi-supervised learning. First is graph construction, in which we must find a suitable way to represent the relationships between the original data points. Second is label inference, in which label information is propagated from the labeled nodes to the unlabeled nodes in the graph.

Graph semi-supervised problems are characterized into two types: transductive and inductive. In transductive settings, we are only concerned with inferring the labels of the unlabeled nodes. In inductive settings, the model is meant to predict labels for any unseen node. For the purposes of our research we focus only on the transductive case due to our interest in how the generated graph properties can be used for inference. In addition, in order to apply and test a broader range of algorithms, we focus only on undirected graphs, which means edges have no start and end nodes. We also use weighted graphs as weighted edges can encode information about the similarity between nodes.

Common graph construction methods require either a distance or similarity function that determines the relationships between each node in the dataset. In unsupervised methods, nodes that are determined to be more similar according to these functions are more likely to be connected by an edge. Examples of these unsupervised approaches are KNN, ε-neighborhood based graph construction, and b-matching.

There are also supervised methods for graph construction which use the given labels as a form of prior knowledge. However, we did not research these methods in depth, as we wanted to focus more on approaches to label inference. Label propagation is a technique used to iterate through the network of points until it reaches convergence. This is when each node has the majority label of its neighbors. As labels propagate, densely connected groups of nodes quickly reach a consensus on a unique label[4].

In most label inference methods, node labels propagate to unlabeled nodes based on the similarity of node pairs. One class of inference methods are the graph regularization methods. These methods all have the following general framework.

$$\mathcal{L}(f) = \sum_{(x_i, y_i) \in \mathcal{D}_l} \underbrace{\mathcal{L}_s(f(x_i), y_i)}_{\text{supervised loss}} + \mu \sum_{x_i \in \mathcal{D}_l + \mathcal{D}_u} \underbrace{\mathcal{L}_r(f(x_i))}_{\text{regularization loss}} ,$$

This can be seen as searching for a function f on the graph and the loss can be broken into a supervised loss component and a regularization loss component. In these algorithms, various different constraints can be enforced with different choices of regularizers. Some of the algorithms we researched that fit in this framework are Gaussian Random Fields, Local and Global Consistency, p-Laplacian, Directed Regularization, Manifold Regularization, Label Prediction via Deformed Graph Laplacian, and Poisson Learning.

A second class of label inference algorithms are graph embedding methods. The aim of these methods is to represent the nodes in lower dimensional vector spaces through an encoding and train the classifiers with this embedding result. With the encoded vector, a decoder framework is used to reconstruct information about a node's neighborhood in the original graph and a similarity matrix.

In particular, shallow graph embedding attempts to use matrix factorization to find a low dimensional approximation of a graph's similarity matrix. Some examples include Locally Linear Embedding, Laplacian Eigenmaps, Graph Factorization, GraRep, HOPE, and M-NMF. In addition there are random-walk based models including DeepWalk, Planetoid, Node2Vec, LINE, PTE, and HARP. More advanced embedding includes deep graph embedding which have more complex encoders based on deep neural networks, to incorporate graph structure and attribute information.

We started our research on semi-supervised learning to answer our overarching question of how to maximize the power of limited labeled data. Semi-supervised Learning with Deep Generative Models[5] offered great insight and detail on what to expect when working with an underdeveloped dataset. Our main focus after research became reducing classification error rate for sparsely distributed graphs using graph based semi supervised learning. This is mainly going to be done by model parameter tweaking and graph structure assumptions. Each model/approach will be used in several experiments on different graph data, with varying properties and assumptions.

The DeepWalk algorithm[6] further explained random walk and provided similar code to what we would use on our datasets.

Methodology

This flow of steps was performed on three distinctive datasets which varied in feature set sizes and data point sizes.

Dataset Preprocessing

The first step was to acquire the dataset and preprocess it to represent multivariate classes as numerical data, for standardisation of our graph construction method. For example, the happiness index dataset had binary classification of '+' and '-' for all its features, which were transformed into 0s and 1s. The class predictions for the happiness was also preprocessed into discernable values, which would ease graph construction. This was the general methodology across the several datasets. In addition, given that we chose fully labeled datasets, we randomly sampled small proportions of the dataset to retain its label, while also removing the label from the remaining data points. The true labels for the entire dataset were saved in order to analyze the accuracy results after predictions were generated.

Graph Construction

For graph construction, we implemented the following three methods and compared the results.
All three methods were unsupervised, as a baseline for comparison.

- KNN: nodes associated with nearest neighbors based on a distance metric
    - Link nearest neighbors greedily to generate irregular graphs with some node degrees higher than k
    - Use a proximity function: sim(x_i, x_j)
        - Measures resemblance/disparity between training points
    - Weights
        - W_{ij} = sim(x_i, x_j) if i in N(j)
        - W_{ij} = 0 otherwise
- ε-neighborhood based graph construction
    - If distance between a node pair is smaller than a predefined constant epsilon, an edge is formed between them
        - However, a misleading choice of epsilon could lead to disconnected graphs

        - KNN-based graphs outperform ε-neighborhood graphs with better scalability
- b-matching: guarantees every node in the resulting graph has exactly b neighbors
    - I.e. graph is regular
    - Steps
        - Graph sparsification
        - Edge re-weighting

In our analysis we found some pros and cons of each method.

KNN

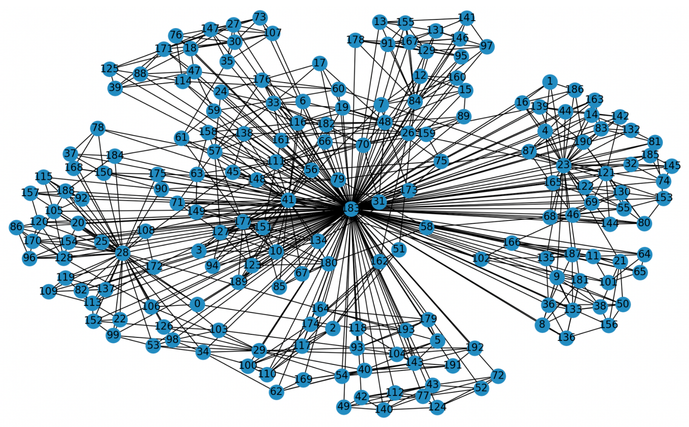
- Pros:
    - not very computationally expensive/relatively fast.
    - All nodes are connected in the graphs.
- Cons:
    - Tendency for “hubs” to form
        - Some nodes are much more connected in the graph than others, and as a result might have a disproportionately large influence in label prediction.

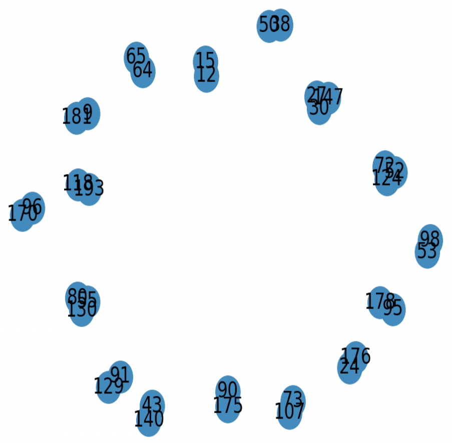
E-neighborhood

- Pros:
    - Not very computationally expensive/relatively fast.
    - No tendency of “hubs” forming as in KNN.
- Cons:
    - Bad choice of epsilon leads to poorly connected graphs.
    - There is no guarantee that all nodes will be connected by edges.

B-matching

- Pros:
    - Graph is balanced as each node has an equal number of neighbors, which also means no “hubs”
- Cons:
    - Very computationally expensive, involves solving optimization problems with thousands of variables, even for relatively small dataset, leading to scalability concerns





On the left, we have a fully connected graph generated with the use of the KNN construction method and on the right, we have a poorly connected graph which was generated by a poor choice of epsilon in ε-neighborhood based graph construction.

Before applying these construction methods, we developed a similarity function that could be used to assess the relationships between the data points. Each dataset had specific similarity functions designed that would assign each node a specific weight based on its proximity to its nearest neighbors. Based on that the graph would be constructed with the in-feature distance as the edges of the graph structure and using the GraphVisualization library, the graphs were visible as a network of nodes and edges. Based on our results, we chose the use only the KNN approach for the inference analysis due to the low computation cost and its ability to produce well connected graphs.

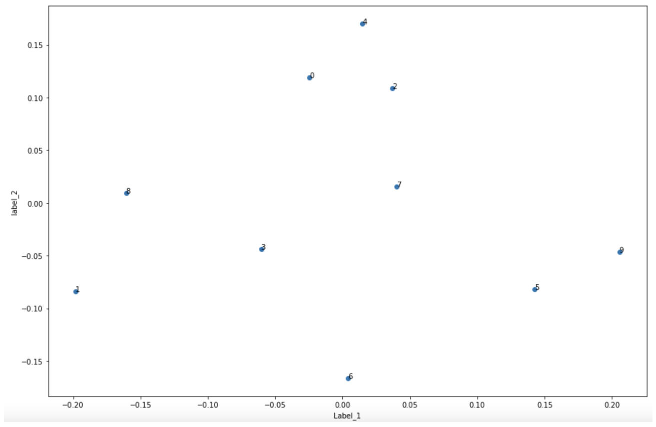
Graph Prediction

Using 4 different graph prediction methods, each dataset was processed and fitted to model an average prediction over several iterations. The methods used were:
Gaussian Random Field (GRF) - Early GSSL algorithm that retains the label for previously labeled points and takes an average of the neighboring points labels for unlabeled points.
Local and Global consistency (LGC) - develops a smoothing activation function to predict labels from unlabeled features.
Label prediction via deformed graph laplacian (LPDGL) - uses local smoothing regularization to predict labels via semi supervised learning techniques
Poisson Learning - solves the issues of degeneracy by structuring the training points into the poisson distribution variables and solves for those in its regularization.

Using these methods on all the datasets, we were able to obtain model predictions and their accuracy scores.

Graph Embedding

Node2Vec and DeepWalk, highly circulated embedding techniques, were used to find low dimensional structure in the data and use that structure to improve the resulting predictions. The datasets with constructed graphs were embedded with the above mentioned algorithms and visualised using Principal Component Analysis (PCA) to recognize data patterns in low dimensions. It was more of an exploratory approach to improve the predictions and can be a crucial tangent in the future scope of this topic of research.



PCA Plot for Happiness Data -

Analysis

Custom Method: Although our general process remained the same in all experiments, we experimented with 2 notable tweaks and their effect on the prediction accuracy. First, we had no basis or standard practice for determining a similarity function that would be used to construct the graphs. Readings suggested that a distance function can be used for the similarity function, so we attempted the process with different distance functions between the nodes. We also compared this to another method we developed, which was to determine the most impactful features by running linear regression on the labeled portion of the dataset. By running linear regression, the coefficients with highest weights were determined to be the most impactful. From these results, we created our own similarity function which weighted the most important feature more.

For example, two points with the same values for more important features would have higher similarity values. We determined from experiments that this approach led to higher accuracy than any of the distance measures. However, it also increased the computation time. To account for this, we experimented with a second tweak. The algorithms LGC, LPDGL, and Poisson Learning all required solving an optimization problem with no closed form solution, so we had to use an optimization solver in our programs. Finding an exact solution required a very long computation time due to the large number of variables and constraints involved in these problems. We tweaked the stopping criteria parameters and loosened constraints of the models slightly in an attempt to lower the computation time of the algorithm. This did result in all of the algorithms running in less time and also only marginally impacted the accuracy. Our method of using the customized similarity function and finding an inexact optimization solution roughly took the same amount of time as the original process and improved the accuracy for all three datasets. Consequently, we recommend this procedure for the application of GSSL in classification problems.

Happiness Dataset Accuracy Results

	LGC	LPDGL	GRF	Poisson Learning
Original Process	28.572	38.234	26.901	38.095
Custom Process	30.080	41.544	26.983	40.424

Table - 1

BreastCancer Dataset Accuracy Results

	LGC	LPDGL	GRF	Poisson Learning
Original Process	30.286	41.670	22.142	43.572
Custom Process	34.722	45.865	23.010	46.120

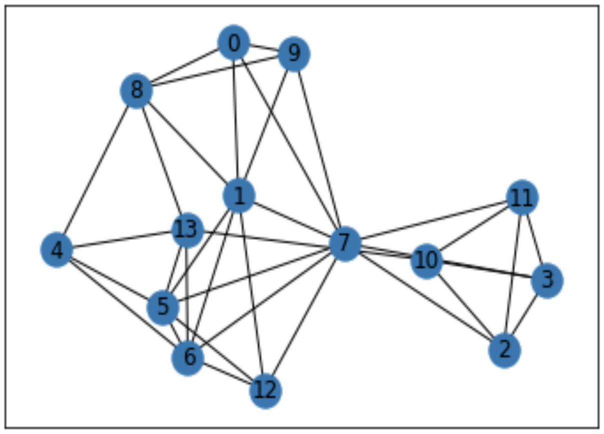
Table - 2

Flag Dataset Accuracy Results

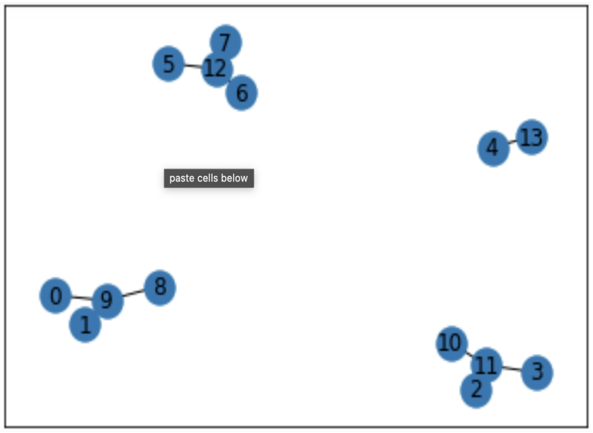
	LGC	LPDGL	GRF	Poisson Learning
Original Process	34.406	29.875	21.532	41.074
Custom Process	41.350	35.012	22.102	50.702

Table - 3

The above are aggregated accuracy results on the several datasets used in the classification problems being addressed in this experimentation. In general Poisson Learning seems to have the highest accuracy irrespective of dataset size or feature set size (that results in increased model complexity in this case). This is because of the inherent dataset structural assumption that poisson learning makes as compared to the other models. The other models have a degeneracy limitation that disables them from assuming structure about the dataset. LPDGL and LGC generally perform worse than Poisson because of the degeneracy limitation in its approach. GRF usually performs the worst out of all because of the approach to averaging its nearest neighbours labels, and since the nearest neighbours are not clustered well enough, there is a high degree of convolutional error.



GRF with KNN when n=4



GRF with KNN when n=1

Future Scope

In the future, it would be most beneficial to continue experimenting on various datasets as we found larger datasets with more classes allow for more data to train on. With such large variability and access to such data, you can train the dataset by removing labels manually and testing it as a semi-supervised model. We also found great value in using weighted edges to connect similar points in semi-supervised learning as distance learning alone would not represent the dataset accurately.

A tweak that could be made in the future is the use of an adaptive algorithm that assumes different things about the data structurally, combining what poisson learning offered while taking some benefits of the other methods we used that did not return high accuracy. A combination of these methods could potentially increase accuracy further than what was found with poisson alone.

Bibliography

[1] “Breast Cancer Survival.” UCI Machine Learning Repository: Haberman's survival data set. Accessed December 5, 2021. https://archive.ics.uci.edu/ml/datasets/Haberman%27s+Survival.

[2] “Flags Dataset.” UCI Machine Learning Repository: Flags data set. Accessed December 5, 2021. https://archive.ics.uci.edu/ml/datasets/Flags.

[3] “Happiness Index in Somerville .” UCI Machine Learning Repository: Somerville happiness survey data set. Accessed December 5, 2021. https://archive.ics.uci.edu/ml/datasets/Somerville+Happiness+Survey.

[4] “Graph Algorithms: Label Propagation.” Big Data Zone. Accessed December 5, 2021.

Graph Algorithms in Neo4j: Label Propagation - DZone Big Data

[5] Kingma, Diederik. “Semi-Supervised Learning with Deep Generative Models - NeurIPS.” Accessed December 5, 2021. https://proceedings.neurips.cc/paper/2014/file/d523773c6b194f37b938d340d5d02232-Paper.pdf .

[6] “DeepWalk Algorithm.” GeeksforGeeks, July 18, 2021. https://www.geeksforgeeks.org/deepwalk-algorithm/.

[7] Song, Zixing. “Graph-Based Semi-Supervised Learning: A ... - Arxiv.org.” Accessed December 6, 2021. https://arxiv.org/pdf/2102.13303.pdf.

[8] Needham, Mark. “Graph Algorithms in neo4j: Label Propagation - DZone Big Data.” dzone.com. DZone, March 8, 2019. https://dzone.com/articles/graph-algorithms-in-neo4j-label-propagation.