Predicting pattern of connectivity between mobile sensors using Graphical Neural Networks

UNDER GRADUATE PROJECT

for the Degree of Bachelor of Technology

by

Vartak Sahil Dileep

(14789)

under the guidance of

Prof. Rajesh M. Hegde

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

KALYANPUR - 208016, UP

Acknowledgements

This Undergraduate project gave me the opportunity to explore wireless networks and deep learning under the mentor-ship of accomplished individuals. It would not have been possible without their help and kind support. I would like to extend my sincere thanks to all of them. The author would like to express his gratitude towards **Prof. Rajesh M. Hegde** for his kind co-operation and encouragement which helped me in the completion of this project. I am also very grateful to my mentor **Surender Redhu**.

Vartak Sahil Dileep
June 2020
Indian Institute of Technology Kanpur

Contents

Abstract

— Recent years have witnessed a dramatic increase of graph applications due to advancements in information and communication technologies. In a variety of applications, such as social networks, communication networks, internet of things (IOTs), and human disease networks, graph data contains rich information and exhibits diverse characteristics. Furthermore, in these applications, the graph data is evolving and expanding more and more dynamically. Driven by the outstanding performance of neural networks in the structured Euclidean domain, recent years have seen a surge of interest in developing neural networks for graphs and data supported on graphs. The graph is leveraged at each layer of the neural network as a parameterization to capture detail at the node level with a reduced number of parameters and computational complexity. Our goal in this project is to optimize the data transmission scheme's performance (In IoT networks) as it minimizes the data latency, improves link reliability and provides throughput improvement by learning the mobility patterns of IoT devices as it gives Apriori connectivity information. We will use the approach of link prediction which provides us with knowledge of how likely are the two nodes connected at a certain timestamp.

List of Figures

Chapter 1

Introduction

IoT sensor networks have an inherent graph structure that can be used to extract graphical features for improving performance in a variety of prediction tasks. We propose a framework that represents IoT sensor network data as a graph, extracts graphical features, and applies feature selection methods to identify the most useful features that are to be used by a classifier for prediction tasks. We show that a set of generic graph-based features can improve performance of sensor network predictions without the need for application-specific and task-specific feature engineering[2]. We apply this approach to a prediction task where four traces of bluetooth sightings by groups of users carrying small devices (iMotes) for a number of days. Our approach produced comparable results with most of the state-of-the-art methods, while maintaining the additional advantage of general applicability to IoT sensor networks without using sophisticated and application-specific feature generation techniques or background knowledge.
We seek to leverage the inherent graph structure in IoT sensor networks to find a generic graph representation of sensor network data. We can provide a tool to IoT application builders to boost prediction task performance if we can build a generic graph-based framework and graphical features to support different prediction tasks from different IoT sensor network data. Graph-based approaches have been successful in several IoT applications.For example, George and Shekar[3] used a graph representation for road navigation to efficiently find the shortest route that may change with time. Graph-based methods that were applied to activity recognition by Long and Holder[4] showed uncorrelated errors when compared to non-graph methods, which indicated the potential for improved performance through the use of a graph representation. Salomon and Tirnauca [5] learn users' behavioral patterns while using weighted finite automata by building a time-aggregated graph from the activity flow of a session (morning, afternoon, evening). They represent each activity as a node and time spent for a particular activity or the time elapsed between different activities as edge labels. George et al. [6] proposed Spatio-Temporal Sensor Graphs (STSG) to capture time dependence of sensor network parameters. We develop a Graphical Feature-based Framework (GFF) to represent and analyze IoT sensor network data.

1.1 Problem Formulation

Applying neural networks and other machine-learning techniques to graph data can be difficult. The first question to answer is: What kind of graph are we dealing with? We have a state machine with some mobile and stationary nodes (for experiment-1) in euclidean space. First we assign our nodes with identities preferably a number. Then you give all the rows the names of the states, and give all the columns the same names, so that the matrix contains an element for every state to intersect with every other state. Then we mark those elements with a 1 or 0 to indicate whether the two states were connected in the graph, or even use weighted nodes (a continuous number) to indicate the likelihood of a transition from one state to the next. Graphs have an arbitrary structure: they are collections of things without a location in space, or with an arbitrary location. They have no proper beginning and no end, and two nodes connected to each other are not necessarily "close". But in our case these problems can be bypassed by dealing in contact time between the nodes and thus we will be using the 1 or 0 method to assign our link connectivity as we are not currently interested in state transition probability.

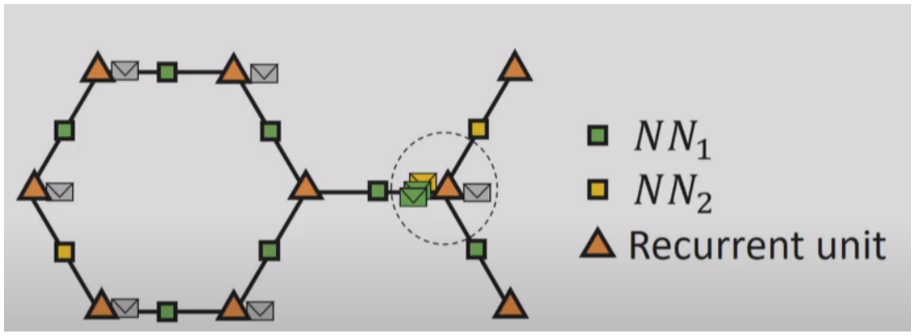

Fig. 1.1: Representation of our 9 node network for the First Experiment

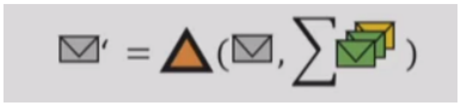

Fig. 1.2: The prediction equation of the particular node at next time stamp[1]

The only way to ingest data into our neural network is Data (graph, words) ->Real number vector ->Deep neural network. Now we have to decide what would our every node or state represent in a vector to be defined for our regression problem. As our data-set gives us information about the time lapsed between the previous connection, the number of times the link has occurred and which neighbouring node is connected to our reference node, this vector is part of the regression problem, we will define. Finally, we can compute derivative functions such as graph Laplacians from the vector/tensor, we defined that represents the graphs of our network, which we will perform an eigen analysis on. These functions will tell us things about the graph that may help us cluster it. aggregate features in vertex neighborhoods to learn vector representations of all vertices, using supervision from some labeled vertices during training. The predictor is then a function of the vector representation, and predictions are made independently on unlabeled nodes. This widely-adopted approach implicitly assumes that vertex labels are independent after conditioning on their neighborhoods but in practice it is seen that this far from the real-life scenario. So we assume that our internal nodes are the only dependent on themselves and not the external nodes given in dataset. With these assumptions we can develop a neural network where our dataset be run through and predict the probability(with error) of any two iMotes' connectivity on a given time stamp.

Chapter 2

Graph Convolutional Networks and comparative analysis

Most graph neural network models have a somewhat universal architecture in common. We will refer to these models as Graph Convolutional Networks (GCNs); convolutional, because filter parameters are typically shared over all locations in the graph.
For our models, the goal is then to learn a function of features on a graph G=(V,E) which takes as input:

- A feature description x_i for every node i; summarized in a $N \times D$ feature matrix X (N: number of nodes, D: number of input features)
- A representative description of the graph structure in matrix form; typically in the form of an adjacency matrix A

and produces a node-level output Z (an $N \times D$ feature matrix, where F is the number of output features per node).[7] Every neural network layer can then be written as a non-linear function H(l+1)=f(H(l),A), with H(0)=X and H(L)=Z , L being the number of layers. The specific models then differ only in how the function f() is chosen and parameterized. The generation of the dataset for GCN training is discussed next, followed by algorithm development for training our neural network.[8]

2.1 Training the GCN

In this section, development of the dataset for training the GCN is discussed. An algorithm for the proposed method for regression using GCNs .

2.2 Dataset for Training

The input training data of the deep neural network consists of row matrices with information of which nodes are connected, how many times have been connected and how much time has passed since they have been connected . The output training data consists of a probability matrix for the input data at each timestamp. The dataset used to derive this data is collection of three experiments with different number of iMotes of which contact time is provided between internal and external devices. We have considered only internal connections for input and output vector for our neural network.[9]

2.3 Algorithm development for regression using GCN

Training a deep neural network for prediction is a non-trivial problem, but with our input and output being firmly defined and can be easily extracted from dataset makes algorithm exponentially easier. We tried three different models and the larger model works the fastest with similar errors to dense or wider models.

Algorithm 1 Algorithm development for regression using GCN

```
 1: procedure OPTIMIZING INPUT AND OUTPUT:
 2:     Generate dataset
 3:     for $i = 0 : maxtime - 1$ do
 4:         Convert our provided dataset to the input and output vectors defined by us before.
 5:         The first four values of input are directly derived from the given dataset.
 6:         The final output value is 0 or 1 depending on the whichver nodes are connected at that given timestamp.
 7:     end for
 8:     Return: Input and Output Vectors.
 9: end procedure
10: procedure TRAINING:
11:     Input: Vector developed above
12:     Import Keras for our model(Keras is a deep learning library)
13:     Baseline model has 3 layers of which most important is the last one where sigmoid function gives us our predicted output.
14:     Run this prediction model with test samples to calculate error
15:     Return: Predicted Output
16: end procedure
```

Chapter 3

Evaluation and Conclusion

Node	Node	Instance of connection	Time elapsed	Outcome	Predicted Outcome
1	2	5	18027	1	1
1	2	5	18028	1	1
1	2	5	18029	1	1
2	3	0	18030	0	0
2	3	0	18027	0	0
2	3	0	18028	0	0
2	3	0	18029	0	0
2	4	2	18030	1	1

Table 3.1: Output of our model on first experimenet on timestamp 18027 to 18030

In this work we have seen other regression models like logistic or linear regression give substantial errors in comparison to our model. Most of GCNs which are being researched on have their inputs as graphs and then they detect nodes and edges of those graphs. But as our dataset had given us our input and output vectors. The error of incorrectly picking nodes and edges is being avoided. Because of this, our model consistently predicts correct outcome. The work described herein can be extended to provide social network models. The biggest advantage of this method is the euclidean space is ignored due to contact time reference. This approach eases our problem in graphical neural networks as constant change in edges is an issue researchers are dealing with. But for specifically for this dataset, a different network can be proposed.

In comparison to other regression like linear regression we have considerable advantage. But the regression is much easier to follow and implement.
Why is there a need of another system when GNNs has already been proposed? GNNs are largely used currently for node classification and extremely limited in regression. Our contact duration model doesn't take full advantage of what GNNs have to offer. Although quite a lot of research has been done on GNNs currently. Almost all machine learning frameworks have a very easy way to implement GNNs. On the other hand, our novel GNNs requires proficiency to code and is a little heavy because it is a network of neural networks.

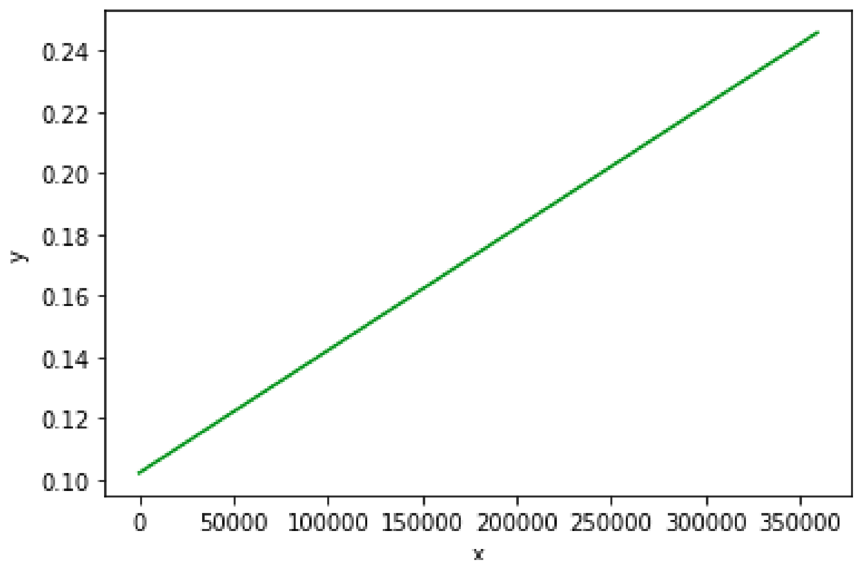

Fig. 3.1: Probablity of connection vs time of Node 2 in Experiment no.1 according to linear regression

References

[1] Alexandra Gaunt, “Graph neural networks: Variations and applications.,” *Microsoft research, University of Cambridge.*

[2] Syeda Akter and Lawrence Holder, “ Improving IoT Predictions through the Identification of Graphical Features.,” *Sensors (Basel). 2019 Aug; 19(15): 3250. Published online 2019 Jul 24. doi: 10.3390/s19153250.*

[3] Shekhar S. George B., “Time-Aggregated Graphs for Modeling Spatio-temporal Networks; pp. 191–212..,” *Journal on Data Semantics XI. Springer-Verlag; Berlin, Germany: 2008.*, pp. 191–212.

[4] Holder L.B Long S.S., “ Using Graphs to Improve Activity Prediction in Smart Environments Based on Motion Sensor Data.,” *Proceedings of the 9th International Conference on Smart Homes and Health Telematics; Montreal, QC, Canada. 20–22 June 2011.*, pp. 57–64.

[5] Tîrnăucă C Salomón S., “ Human Activity Recognition through Weighted Finite Automata.,” *Proc. Int. Conf. Ubiquitous Comput. Ambient. Intell. 2018;2:1263. doi: 10.3390/proceedings2191263.*

[6] Shekhar S. George B., Kang J.M., “ Using Graphs to Improve Activity Prediction in Smart Environments Based on Motion Sensor Data.,” *A Data Model for the Discovery of Spatio-temporal Patterns. Intell. Data Anal. 2009;13:457–475. doi: 10.3233/IDA-2009-0376.*, pp. 457–475.

[7] Thomas N Kipf and Max Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2016.

[8] Chris V. Nicholson, “ Math behind Neural Network.,” *Skywind.ai Wiki:part 3. 2019 Aug;. Published online 2019 May 24.*, pp. 1–13.

[9] James Scott and Richard Gass and Jon Crowcroft and Pan Hui and Christophe Diot and Augustin Chaintreau, “CRAWDAD dataset cambridge/haggle (v. 2009-05-29),” Downloaded from url-https://crawdad.org/cambridge/haggle/20090529, May 2009.