UNSW
SYDNEY

TITLE?

Justin Clarke

Supervisor: Associate Professor Yanan Fan

School of Mathematics and Statistics
UNSW Sydney

October 2023

Submitted in partial fulfillment of the requirements of the degree of
Bachelor of Science with Honours

Plagiarism statement

I declare that this thesis is my own work, except where acknowledged, and has not been submitted for academic credit elsewhere.

I acknowledge that the assessor of this thesis may, for the purpose of assessing it:

- Reproduce it and provide a copy to another member of the University; and/or,
- Communicate a copy of it to a plagiarism checking service (which may then retain a copy of it on its database for the purpose of future plagiarism checking).

I certify that I have read and understood the University Rules in respect of Student Academic Misconduct, and am aware of any potential plagiarism penalties which may apply.

By signing this declaration I am agreeing to the statements and conditions above.

Signed: Justin Clarke Date: 21/07/2023

Acknowledgements

Abstract

Graph sparification techniques for graph neural networks have traditionally been used to accelerate training and inference on real-world graphs which have billions of paramaters. There are also many different climate models which use complex mathematical models to model the interactions between energy and matter over the world. Many of these models share components and the structure of these relationships is not easily found due to the complexity of these climate models. The space of all possible graphs grows super-exponentially with the number of nodes and as such any correlation or causality is difficult to find. In this paper, I attempt to quantify these relationships with graph sparsification techniques. (Talk more about climate?)

Contents

Chapter 1 Introduction 1
1.1 Motivation . . . 1
1.2 Approach . . . 1

Chapter 2 Background and Related Techniques 2
2.1 Climate Models . . . 2
2.2 Deep Learning . . . 2
2.3 Graph Neural Networks . . . 3
2.4 Sparsification Graph Neural Network . . . 3
2.5 Variational Autoencoder . . . 4

Chapter 3 Framework 5
3.1 Dataset . . . 5
3.2 Problem Formulation . . . 5
3.3 Computational features . . . 6
3.4 Implementation . . . 6

Chapter 4 Results 7
4.1 Model verification . . . 7
4.2 Model results . . . 7

Chapter 5 Discussion 8

Chapter 6 Conclusion 9

Chapter 7 Appendix 10

References 11

Chapter 1

Introduction

1.1 Motivation

The simplest climate modeles have existed since the 1950's with the very first computers modelling small two-dimensional climates. Modern models have become increasingly more complex in part due to the increasing computational power available today and the large amount of data available worldwide to train these models on. Many of these models have become unexplainable due to the sheer complexity and number of their parts yet many share components and frameworks. One of the most important questions that climate science is attempting to answer today is what impact have humans had on the future of the climate. The prediction of climate change is important as it can guide us on the potential harms we may be causing to environment and life around us. As such, many models and 'scenario runs' have been developed which predict various outcomes in temperature, precipitation, air pressure and solar radiation given a certain level of societal development. On the lower end, SSP126 assumes an increasingly sustainable world where consumption is oriented towards minimising material resource and energy usage while SSP585 assumes a worst case scenario where fossil fuel usage and an energy-intensive lifestyle intensifies. (Talk more about the math behind these models? Stochastic Differential models or talk about a few of the main models in use today?)

In recent years, Graph neural networks have become the premier method of processing data with non-cartesian structure. Much of this data exists in the world in applications such as chemical analysis, social networks and link prediction (Insert references for each from reading). The main feature of GNNs is the message passing framework, where information from features on each node is passed to neighbouring nodes then aggregated and embedded. This is then propagated through a neural network structure to perform a range of tasks on the entire graph, individual nodes and edges.

1.2 Approach

Overview on my goal and how I am testing this goal.

Chapter 2

Background and Related Techniques

In this chapter we will give a brief overview on how the climate works and review current standards in climate modelling along with the basics behind neural networks and the extensions towards graph neural networks.

This will basically be the section on literature review. Current methods and techniques being used etc. A lot of summaries of the papers saved in the papers folder need to be done to finish this section.

2.1 Climate Models

What climate models are used for etc. Use Yanan's climate papers.

2.2 Deep Learning

Traditional machine learning techiques generally require meaningful data cleaning and feature creation which was costly to develop and often had many errors. The advent of deep learning provided algorithms that could automatically extract higher-level features from raw data. [2] The MLP The multi-layer perceptron [8] is formulated using linked layers of nodes which transforms a set of inputs into an output. For a regression task, the MLP attempts to approximate some ground-truth function which may also be non-linear. This can be represented as

$$f(x) = \sigma(\Theta^T X) \text{ where } \Theta = \begin{bmatrix} b \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \text{ and } X = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \tag{2.2.1}$$

Where $\sigma(.)$ is some activation function such as ReLu, hyperbolic tangent or logistic function, X is the data, Θ is the learned parameters and b represents a bias term. These parameters are set to some initial values and are iteratively updated in a back-propagation training process,

$$\theta^{t+1} = \theta^t - \eta \frac{\partial E(X, \theta^t)}{\partial \theta} \tag{2.2.2}$$

Where $E(.)$ is some loss function and η is the learning rate. The next advancement in the deep learning space came with the Convolutional Neural Network (CNN) which was a regularised MLP that could handle data with data with structure and multiple dimensions far better than the traditional MLP due to its use of weight sharing, sampling and local receptive fields. [4]
Suppose we have an image or some other kind of data in matrix form. Let $\mathbf{X} \in \mathbb{R}^{H \times W}$ be the input image and $\mathbf{W} \in \mathbb{R}^{h \times w}$ be the kernel or filter. By performing a

convolution, we are effectively 'sliding' our weight matrix kernel over our input and the resulting feature map $\mathbf{Z} = \mathbf{X} * \mathbf{W}$,

$$Z_{i,j} = \sum_{u=0}^{h-1} \sum_{v=0}^{w-1} x_{i+u,j+v} w_{u,v} \tag{2.2.3}$$

The novelty of the convolutional layer compared to a linear layer is that the kernel is shared across all locations of the input and therefore if a pattern in the input moves, the corresponding output will also follow this movement. This provides shift equivariance which is something that early MLP's failed to achieve. [6]

2.3 Graph Neural Networks

Summarise the extension of GNN's from NN's and how they are useful in certain applications. When it comes to data in a graph-like structure, standard CNN's cannot be applied due to the non-euclidean nature of a graph. In an image or a matrix, our kernel is generally a $n \times n$ matrix which can be applied to the entirety of the data. In graphs this is not always possible due to the fact that any number of nodes can be connected by any number of edges. [9] The Graph Neural Network (GNN) was developed for this purpose and they can be broadly categorised into gating and attention based methods [10] and spectral or spatial methods within Graph Convolutional Network (GCN) research. [5] This propagation rule is as follows:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}) \tag{2.3.1}$$

Here, $\tilde{A} = A + I_N$ is the adjacency matrix of the undirected graph $\mathcal{G}$ with self-connections from the identity matrix I_N. $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ is the row sum of the adjacency matrix and $W^{(l)}$ is a layer trainable weight matrix. $\sigma(.)$ is an activation function such as $\text{ReLU}(.) = \max(0, .)$. $H^{(l)} \in \mathbb{R}^{N \times D}$ is the matrix of activations in the lth layer with $H^0 = X$.

2.4 Sparsification Graph Neural Network

New section for graph sparsification? Towards sparisification of GNN's [7] and unified lottery ticket hypothesis [1] for gnn's should be reviewed here.

In recent years, the size of deep learning models has grown exponentially with some models using billions of parameters. Although neural networks do not tend to be affected greatly by overparameterisation, these models have become increasingly costly both in terms of inference and prediction. The Lottery Ticket Hypothesis (LTH) [3] explored the possibility of simplifying redundant models by trainable sparse subnetworks whilst still training to full accuracy. When training a traditional neural network, this was done by instability analysis etc. etc.? (should i talk about this?)

Chen et. al. [1] extended the LTH to Graph Neural Networks by co-simplifying both the adjacentcy matrix of the graph and the weights in the network of the model. For a semi-supervised classification task, the objective function is:

$$\mathcal{L}(\mathcal{G}, \Theta) = -\frac{1}{|\mathcal{V}_{\text{label}}|} \sum_{v_i \in \mathcal{V}_{\text{label}}} y_i \log(z_i), \tag{2.4.1}$$

where $\mathcal{L}$ is the cross-entropy error of all samples and y_i is the label vector of node v_i. The Unified GNN Sparsification (UGS) framework then introduced two masks m_g and m_θ with the same shape as the adjacency matrix $\mathbf{A}$ and the weights matrix Θ, which gives the following objective function:

$$\mathcal{L}_{\text{UGS}} = \mathcal{L}(\{m_g \odot A, \mathbf{X}\}, m_\theta \odot \Theta) + \gamma_1 ||m_g||_1 + \gamma_2 ||m_\theta||_1, \tag{2.4.2}$$

where $\odot$ is the element-wise product, γ_1 and γ_2 are hyperparameters to control the shrinkage of m_g and m_θ. After training, the lowest magnitude elements in m_g and m_θ are set to zero with respect to some set values of p_g and p_θ. These sparse masks are then applied which prune $\mathbf{A}$ and Θ.

2.5 Variational Autoencoder

This section should be done with more time if the original sparsification section is completed. Show how this is an alternative in graph discovery.

Chapter 3

Framework

3.1 Dataset

The datasets used are from the CIMP6 scenario runs made available on the KNMI Climate Explorer website. The KNMI is part of the World Meteoriological Organization (WMO) []. The scenario runs include monthly predictions for temperature, min temperature, max temperature precipitation, radiation and pressure all over the globe in a 192x144 grid between 1850-2100 for 40 different models. For simplicity and brievity, this was filtered to just temperature during the 1960-1970 period in just Australia.

Talk a bit more about how these models in the dataset are all related by certain parts.

3.2 Problem Formulation

Maybe add this to the background section and put more of the regression, diagrams of the process and shrinkage prior stuff here that it more specific to this thesis

We define a graph as $\mathcal{G} = (\mathcal{V}, \mathbf{A})$, where $\mathcal{V}$ represents a set of verticies which contains a list of nodes $\{v_1, \ldots, v_n\}$ and $\mathbf{A} \in \mathbb{R}^{n \times n}$ the adjacentcy matrix which contains information on the graph topology. If an edge exists between two node v_i and v_j, then $\mathbf{A}_{ij} = 1$ else, $\mathbf{A}_{ij} = 0$. We also define the degree matrix as $\mathbf{D} = \sum_j A_{ij}$ where each entry on the diagonal is equal to the row sum of the adjacency matrix $\mathbf{A}$. Each node has a p-dimensional feature vector $x_i \in \mathbb{R}^p$ which describes some information about the node in the graph. By combining all n feature vectors from all nodes, we have a feature matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$. The graph also has a regression target $Y \in \mathbb{R}$ which refers to the historical temperature that each model from the graph is attempting to predict. As mentioned earlier, other variables such as precipitation, pressure and radiation are available but for simplicity, just temperature is currently being used. The two-layer GNN from [5] can be expressed as

$$f(\mathbf{A}, \mathbf{X}) = \sigma_2(\hat{\mathbf{A}}_2 \sigma_1(\hat{\mathbf{A}}_1 \mathbf{X} W^{(0)}) W^{(1)}) \tag{3.2.1}$$

where $\sigma_1(.)$ and $\sigma_2(.)$ are an activation function such as ReLU, and $\hat{A} = \tilde{D}^{-1/2}(A + I)\tilde{D}^{-1/2}$ is the symmetrically normalised adjacency matrix. The final regression problem can be formulated as

$$f : L \times X \to Y \tag{3.2.2}$$

where f denotes the learning function, L the graph, X denotes the time series input and Y the regression target.

Need to describe the math behind graph sparsification. More about shrinkage see Xiongwens.

3.3 Computational features

computation of neural network models. See georges paper

3.4 Implementation

Need to finish code to finish this section.

Chapter 4

Results

4.1 Model verification

If the VGAE section is completed, we can compared the sparsified graph with the VGAE produced graph to determine how good graph sparsification is when used for graph discovery and thereby correlation in a graph structure.

4.2 Model results

Is there some way we can test the models results depending on how sparse we make the graph etc. Research required to find some quantitative measure for this.

Some figures of the NN structure would also be helpful for this. Need to use nx or some other graph representation tool in python for this.

Chapter 5

Discussion

Chapter 6

Conclusion

Chapter 7

Appendix

References

[1] Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. A unified lottery ticket hypothesis for graph neural networks, 2021.

[2] Li Deng and Dong Yu. *Deep Learning: Methods and Applications*. Now Foundations and Trends, 2014.

[3] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis, 2020.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[5] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[6] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

[7] Hongwu Peng, Deniz Gurevin, Shaoyi Huang, Tong Geng, Weiwen Jiang, Omer Khan, and Caiwen Ding. Towards sparsification of graph neural networks, 2023.

[8] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[9] Benjamin Sanchez-Lengeling, Emily Reif, Adam Pearce, and Alexander B. Wiltschko. A gentle introduction to graph neural networks. *Distill*, 2021. https://distill.pub/2021/gnn-intro.

[10] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.