CHAPTER 1

Introduction

Temporal networks are networks whose nodes and edges change in time. In this thesis we look to model temporal networks. That is, describe the changing temporal network using mathematical language. The modelling of temporal networks is an important task in many real world applications including symptom interactions for mental health [27, 24], epidemiology [19], and protein interactions [34, 13].

Temporal networks can be seen as a system in which points, in our case nodes in a network, whose states, the edges connecting them, vary dependent in time, a dynamical system.

Discovering the underlying equations governing these dynamical systems proves challenging. That is because changes in network structure are typically observed in the form of discrete jumps from one state to another; for example an edge between two nodes not being observed at the first time step then being observed at the next.

In this thesis, we propose a hybrid statistical and deep learning framework. That is a framework in which a neural network is used to model probabilistic values of the temporal network. This framework allows us to model temporal networks as continuous-time dynamical systems, discover a fitting set of differential equations describing it, and, exploiting that discovery, predict the time evolution of a temporal network.

Differential equations are useful for modelling systems where the state of one variable can effect the trajectories of other variables. We observe this behavior in

temporal networks; nodes' connections within the network can influence the observation of edges between other nodes, for example the phenomenon observed in [7, 10], where a node is more likely to gain the connections the more connections it has (preferential attachment). With this in mind we might wish to draw on the rich mathematical literature of differential equation modelling.

In the common representation of networks as binary-valued adjacency matrices, matrices where the observation or lack thereof, of an edge is represented as a 1 or 0 respectively, the events recorded in a temporal sequence of networks correspond to the appearance or the disappearance of link.

Because of the discrete nature of events, directly modelling the temporal networks as dynamical systems would require us to handle discrete jumps. The discontinuous character of their temporal evolution, make it challenging to use differential equations techniques.

Here, we overcome the discreteness problem by interpreting temporal networks as a well established statistical model for complex networks, which embeds nodes in a continuous, low-dimensional metric space by using a truncated singular value decomposition (Random Dot Product Graphs[22]). In this way we translate the hard problem of modelling discrete events in the space of networks to the easier problem of modelling continuous change in the embedding space.

We then define and use Neural Network Differential Equations (NNDE)[28] to approximate the time evolution of the embedding space, and symbolic regression techniques to discover the functional form of the fitted NNDEs. These functional forms are interpretable (as they read as classic differential equations) and allow us to predict forward in time the evolution of the temporal networks.

In this manuscript, we show that the temporal network prediction problem can be successfully re-interpreted as a dynamical system modelling problem by taking the singular value decomposition of a sequence of adjacency matrices and training a

NNDE to model an approximation to the underlying differential equation. We then go on to create a functional equation of this approximation.

We apply our proposed framework to three small example temporal networks with the hope of exploring the limitations and strengths of the proposed framework.

The framework we are introducing is extremely flexible, and our research regarding the optimal structure of the Neural Networks used for the NNDEs is just started. We are confident that future research can identify more fitting Neural Network structures than the simple one adopted here. For this reason, we did not yet attempt to benchmark our model against other classic temporal network prediction methods.

As it is completely general, we believe that the framework we are introducing can be usefully applied to areas of medicine, especially protein interaction networks; population dynamics for network ecology; and social network modelling. In particular, we discuss how specific domain knowledge relative to the prediction scenario can be taken into account, moving from NNDEs to Universal Differential Equations (UDEs).

CHAPTER 2

Literature Review

1. Temporal Networks

1.1. Definitions. Temporal networks are used to describe a series of interactions across time. A temporal network G is defined as:

$$G = \{{}_tG = ({}_tN, {}_tE)\}_{t \in 1 \dots T} \tag{1}$$

${}_tx \in {}_tN, {}_te \in {}_tE$, where ${}_tx, {}_te$ are nodes and edges at time t respectively, and ${}_tN, {}_tE$ are the sets of nodes and edges at time t respectively. An example of this can be found in fig. 1.

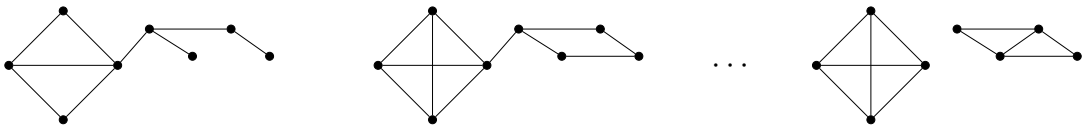


FIGURE 1. Example sequence of networks.

This can be represented in many ways, for example as a series of interactions. The series of interactions is generally represented as a list of tuples containing the nodes which interacted and the time that the interaction took place.

$$G = \{(i, j, t) | i, j \in {}_tN, (i \leftrightarrow j) \in {}_tE\} \tag{2}$$

A temporal network can also be represented as a sequence of adjacency matrices. In the representation of a series of adjacency matrices, the columns and rows of the matrix each refer to each of the nodes. In this way, if an edge exists between nodes

i and j, the entry (i, j) of the adjacency matrix will be 1; if an edge does not exist the entry will be 0.

$$ {}_tG = {}_tA \tag{3}$$

$$ {}_tA_{ij} = \begin{cases} 1 & (i \leftrightarrow j) \in {}_tE \\ 0 & otherwise \end{cases} \tag{4}$$

The row ${}_tA_{i\cdot}$ describes the outgoing edges of the node i, while the column ${}_tA_{\cdot j}$ describes the incoming edges of the node j.

1.2. Applications. Temporal networks are present in many areas of research interest, ranging from symptom interactions in mental health[27, 24], to epidemiology[19], protein interactions[34, 13], and social networks[21, 15].

Generally attempts to model temporal networks either model the change in node state or the change in network structure.

The node state describes some information about that node:

$$ {}_tx = (m_1, \ldots, m_p) \tag{5}$$

Where m_i is some metadata about the node ${}_tx$.

An example of modelling the change in a node state could be the severity of a symptom at a given time, as in Contreras et al.[24]. Where the node ${}_tx$ represents a given symptom and m_1 would represent the severity of that symptom.

Contreras et al.[24] model the change in node state over time while the structure of the network remains constant. The authors look at the severity of symptoms over time, i.e. the state of the nodes, where the nodes exist in a statically structured temporal network.

Contreras et al. use multilevel vector autoregression, a method that finds correlations between the current state of a variable and the states of variables at previous time steps [9], to model how the current symptoms of people with paranoia might

predict their future symptoms. These models were then used to create three networks that linked symptoms.

Of particular interest to this thesis is the temporal network created by having a fully connected network of all symptoms including self-loops, where a symptom connects to itself; this self loop comes about from a symptom being correlated with itself from one time step to the next. The edge weights represent the extent to which the severity of each symptom at time (t) predicts the severity of itself and other symptoms at time $(t+1)$.

This framework keeps the overall structure of the network fixed throughout time and so is not particularly flexible, but is extremely human interpretable and can be used to extrapolate and predict into the future.

In Karimi and Holme[18], Karimi and Holme look to model the change in node state while the network structure changes.

Karimi and Holme represent their temporal network as a series of interactions (a list of tuples in the form (i, j, t), where i, j are the nodes interacting and t is the time the at which the interaction occurred).

The authors use a model to predict when a node will change from state 0 to state 1, in practice this could represent whether a user of a social network believes a rumour, or whether an ecological patch has been colonised.

Notably, in this model, the node never changes back; this is done to keep the model analytically tractable, but does limit its usefulness in some contexts. In this model an organism could not become locally extinct in an ecological patch for example.

To model the change in a nodes' state, the model uses a sliding time window and if the fraction of interactions with nodes of state 1 within that time window exceeds a given threshold, the node switches to state 1 as well.

In this work we aim to present a model for predicting the state of a node in the future. Instead, Karimi and Holme aim to present a tool for understanding the spread of ideas, viruses, etc. within a network.

Other efforts focus on modelling the changing structure of the network as in Sanna Passino et al.[37], which looks to model ${}_tG$ as a whole.

In this case Sanna Passino et al. represent the temporal network as a series of adjacency matrices, each matrix representing an observation of the changing network.

The authors look to model the structure of edges by using a spectral embedding (SVD). A spectral embedding uses the eigenvectors of a network's adjacency matrix to find vectors that represent the nodes of the network. The spectral embedding transforms their temporal sequence into a sequence of observations of a latent space.

A latent space is a continuous space in which similar nodes (nodes with many of the same neighbours) are positioned near each other. In this space nodes are represented as points in a continuous, low dimensional space. Another feature of this latent space, is that the dot product of any two points in the space gives us the probability that the nodes related to those points are connected, a Random Dot Product Graph (RDPG)[22].

With this series of embeddings, the authors employ a variety of time series techniques to predict the network structure at future time steps.

2. UDEs and NNDEs

In some temporal networks the state of a node might influence the evolution of its neighbours. For example, in social networks we often observe that the more neighbours a node has, the more likely they are to gain more neighbours, a power law distribution of edges[17, 12]. We also see that the state of symptoms (occurrence, severity, and distress), can influence the states of other symptoms in patients undergoing chemotherapy [23, 38], and for mental health disorders [24]. As well as

in ecological networks where the interactions of species are often represented as a network, and the populations of one species may influence the population growth of another [6, 1]; that is, the state (in this case the population) of a species (node), may influence the movement of another species.

With the observation that nodes in real world temporal networks may influence others, differential equations or difference equations seem appropriate for modelling the evolution of nodes; there is a rich body of literature exploring these methods use in modelling interacting variables.

One of the issues that the differential equation method encounters is the prevalence of discrete jumps of edges being observed and then not being observed. Because of this, we may consider exploring the possibility of using difference equations.

We note that the progression of the network is generally continuous even if the events happen as discrete jumps. We may wish to preserve this continuous temporal progression in our model.

Looking at the work of Sanna Passino et al.[37], the authors use RDPGs to approximate a temporal network as a matrix of probabilities of an edge existing between two nodes. In contrast to edges, these probabilities can evolve continuously, and so we can use differential equations to generate probabilistic networks at any temporal resolution we require. That is, we will be able to model the changes in the probabilities of edges continuously and from this, generate a network at any time step.

Using the method in Sanna Passino et al.[37], we can treat the sequence of embeddings as a dynamical system and then use differential equation modelling in a continuous space. Once we have found a latent space for each network, we know that each point in this space refers to a specific node in the network, and so, as the network changes, so too does the position of each of the points in the latent space. We then look to model this movement using differential equations.

Our aim for this framework is to be as general as possible and note that there may not yet be enough domain knowledge to perform traditional differential equation modelling to the desired accuracy. For example in network ecology models can be useful, but not perfect. Hence, we might wish to build on these models rather than trying to create something entirely new. This can be achieved by using this partial domain knowledge using it to create a UDE[28].

2.1. Theory. Developed by Rackauckas et al.[28], UDEs are a novel neural network architecture that aims to take the best of both traditional neural differential equation modelling and of flexible machine learning approaches to modelling.

If we consider a comparison to an Ordinary Differential Equation (ODE), the ODE is written as:

$$y(0) = y_0 \tag{6}$$

$$\frac{dy}{dt}(t) = f(t, y(t)) \tag{7}$$

The UDE will incorporate some component of $f(t, y(t)) \approx \hat{f}(t, y(t))$, and will learn the difference with a Neural Network (NN) such that:

$$y(0) = y_0 \tag{8}$$

$$\frac{dy}{dt}(t) = \hat{f}(t, y(t)) + NN(t, y(t), \theta) \tag{9}$$

Where θ is a vector of the parameters of the NN. UDEs combine domain knowledge in the form of a differential equation, with a NN. This hybrid machine learning model is then trained on the observed data.

Combining the two allows for most of the movement in the data to be captured by the differential equation, which contains the domain knowledge. In this way, the NN will only need to learn a theoretically simpler equation, given that a part of the system has already been captured.

This then allows the NN to be smaller, require fewer data, and be trained much faster than traditional NN models. Given the flexibility of NNs, the use of UDEs is natural when modelling a process that is not entirely understood[39].

Note that we can construct a special case of a UDE, an NNDE when $\hat{f}(t, y(t)) = 0$. This structure can be used when do not have any domain knowledge we wish to give the model.

There is an ecosystem of packages specifically designed for performing machine learning with UDEs. The ecosystem is called Scientific Machine Learning (SciML)[28]. Largely, the SciML ecosystem has been optimised for flexibility and efficiency with respect to models available and training performance. Given there does not seem to be any other ecosystem with this feature, it seemed like an obvious choice.

2.2. Applications. Whilst there has been much interest in implementing UDEs and NNDEs, much of this work has been focussed on physics informed neural network and physics informed neural ordinary differential equations (PINNs and PINODEs) [31, 30, 33, 29], and on improving modelling of fluids [35, 40]. Alongside this, UDEs have gained popularity in the last few years, and there have been a number of studies exploring their usefulness in modeling the effect of restrictions due to COVID-19 on the virus' spread [26]. Although there has been a large amount of research into the usefulness of these types of models, to the best of this author's knowledge they have never been applied to the problem of predicting temporal networks. This seems to be a natural fit for UDEs and NNDEs, given we may know relatively little about the processes that govern the evolution of temporal networks, and any information we do have can be incorporated into the model.

3. Symbolic Regression

Symbolic regression is a type of analysis that looks to take input variables and find a relationship between them that describes the output variables. For symbolic regression, we construct a set of mathematical operations that may be used to find this relationship. The algorithm then generates combinations of these equations to find an expression that matches the output variables. Because we construct the set of potential operations, the resulting expression is interpretable by humans (a functional equation), as opposed to a NN for example (a numeric equation).

Having interpretable models that capture underlying relations is relevant to all fields of study, and so the applications of symbolic regression are incredibly varied.

Various methods of symbolic regression have been use in manufacturing systems, chemical systems, and tumor research[16, 32, 20]. As well research into improvements is still on going.

Finding a suitable functional expression can be achieved in many ways and remains active area of research. One of the most popular algorithms is genetic programming[14].

Genetic programming involves constructing a search tree using an evolutionary algorithm to yield candidate equations. An evolutionary algorithm is an iterative process. It begins by randomly generating a set of candidate equations with operations from the given set of operations:

$$Q_1 = \{f_1, \ldots, f_n\} \tag{10}$$

Each candidate function f_i is tested and given a fitness score based on how well it performed. The higher a candidates' fitness, the more likely it is to be selected for the next iteration.

Suppose f_1, f_2 were selected for the next iteration. The evolutionary algorithm would then randomly generate new functions based on f_1, f_2 again with the given set of operations.

$$Q_2 = \{f_{1,1}, \ldots, f_{1,n}, f_{2,1}, \ldots, f_{2,n}\} \tag{11}$$

This process is repeated until a suitable functional equation is found.

For the genetic programming algorithm, the fitness of a candidate function is determined by how closely it matches the data as well as numerically calculated partial derivatives of each of the input variables.

Comparing with these derivatives is useful as it helps to ensure that the produced function equations have some grounding in the real world process that generated the data. The search tree is then pruned by comparing the partial derivatives of the candidate equation, with partial derivatives calculated from data. This process is then iterated many times to generate a suitable equation.

The problem that symbolic regression aims to solve is to generate interpretable and "meaningful" equations from observed data.

For example, both Schmidt and Lipson[14], and Bongard and Lipson[11] test their methods' capacity for obtaining a functional equation with physical simulation data, various pendula specifically. A "meaningful" equation in this case may be an equation that conserves momentum.

Examples of genetic programming include discovering equations governing the movement of simple harmonic and chaotic double pendula from data. In Schmidt and Lipson[14], the authors present an evolutionary algorithm to generate multiple candidate functions that approximate the partial derivatives of each variable that are calculated directly from the data.

Another approach to symbolic regression can be found in Bongard and Lipson[11]. The authors propose a method to find a functional equation that approximates a learned numerical equation rather than directly from data.

The algorithm generates a set of candidate equations. The candidate equations are then tested against the numerical equation in such a way that the difference between the candidate and numerical predictions are maximised. The algorithm will generate new initial conditions to find behaviour exhibited by the numerical equation, for example a fixed point, that is not exhibited by the functional equation. When a difference is found, the functional equations are adjusted to also exhibit the behaviour of the numerical equation.

We will be using a UDE with relatively small data sets. The models will be trained on a small range of input values, and so we would not have confidence that any behaviour exhibited outside the range of these input values would reflect the underlying system. As such, this method may not yield us useful results.

In very recent work, Kidger[39] demonstrates that, with UDEs, two of the assumptions necessary for symbolic regression can be overcome. Those are the requirement for paired observations and derivatives, as well as the assumption that the function can be expressed as a shallow tree of symbolic operations. We are modelling a differential equation, and so will have access to the derivatives of our numerical model at any point, as neural networks are easily differentiable.

In Rackauckas et al.[28], the authors demonstrate that by using symbolic regression to find a functional equation of our model, as opposed to just a UDE, we can improve the accuracy of extrapolating.

CHAPTER 3

Methodology

We are given a sequence of graphs, and our goal is to predict the next graph in the sequence.

To achieve this, we reinterpret our temporal network as a sequence of points in a latent space; that is, a space in which similar vertices (ones with many of the same neighbours) are mapped to a similar point in space. We then train a Neural Network Differential Equation (NNDE) to approximate the rate of change of the points in this latent space. With this numerical equation we can then use symbolic regression techniques to find a functional equation that matches the numerical equation. Using this process we discover an interpretable function that governs the evolution of a temporal network.

1. Singular Value Decomposition and Random Dot Product Graphs

In this section we discuss the Singular Value Decomposition (SVD) of a matrix, and its usefulness in creating Random Dot Product Graphs (RDPG).

We define the matrix A to be a real $m \times n$ with $n \leq m$. A can be expressed as [2]:

$$A = l\sqrt{\Sigma}\sqrt{\Sigma}r' \tag{12}$$

$$A = LR' \quad \text{where } L = l\sqrt{\Sigma},\ R = \sqrt{\Sigma}r' \tag{13}$$

Where l, r are real valued, orthonormal matrices. The columns of l consist of the n largest eigenvectors of AA' and the columns r consist of the eigenvectors of $A'A$. Σ

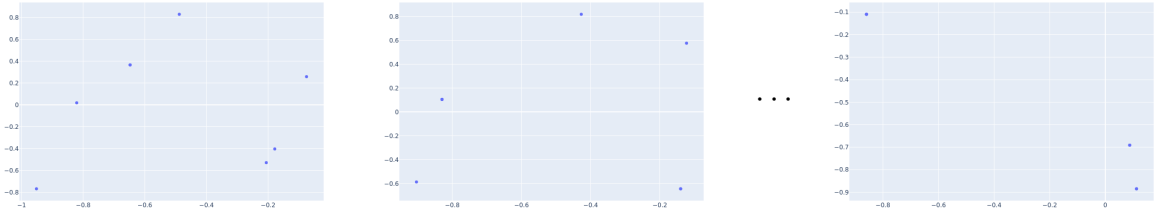
$$\mathbf{H}$$

$$\begin{bmatrix} \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot & \cdot \\ 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & 1 & \cdot & 1 & 1 & \cdot & \cdot & \cdot \\ 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot & 1 & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \end{bmatrix} \begin{bmatrix} \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot & \cdot \\ 1 & \cdot & 1 & 1 & \cdot & \cdot & \cdot & \cdot \\ 1 & 1 & \cdot & 1 & 1 & \cdot & \cdot & \cdot \\ 1 & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot & 1 & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot \end{bmatrix} \quad \cdots \quad \begin{bmatrix} \cdot & 1 & 1 & 1 & \cdot & \cdot & \cdot & \cdot \\ 1 & \cdot & 1 & 1 & \cdot & \cdot & \cdot & \cdot \\ 1 & 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\ 1 & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 \\ \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot \end{bmatrix}$$

(A) Sequence of sparse adjacency matrices associated with the networks in fig. 1.

$$\begin{bmatrix} -0.82 & 0.01 \\ -0.64 & 0.36 \\ -0.95 & -0.76 \\ -0.64 & 0.36 \\ -0.48 & 0.82 \\ -0.17 & -0.40 \\ -0.20 & -0.52 \\ -0.07 & 0.25 \end{bmatrix} \begin{bmatrix} -0.82 & 0.10 \\ -0.82 & 0.10 \\ -0.90 & -0.58 \\ -0.82 & 0.10 \\ -0.42 & 0.81 \\ -0.13 & -0.64 \\ -0.13 & -0.64 \\ -0.12 & 0.57 \end{bmatrix} \quad \cdots \quad \begin{bmatrix} -0.85 & -0.10 \\ -0.85 & -0.10 \\ -0.85 & -0.10 \\ -0.85 & -0.10 \\ 0.08 & -0.69 \\ 0.11 & -0.88 \\ 0.11 & -0.88 \\ 0.08 & -0.69 \end{bmatrix}$$

(B) Sequence of truncated SVDs associated with the networks in fig. 1.



(C) Sequence of embedded points associated with the networks in fig. 1.

FIGURE 1. Visual illustration of the proposed framework.

is a diagonal matrix whose entries are the square root of the positive eigenvalues of $A'A$ in decreasing order.

In order to reduce the problem size, we can truncate the L, R to the first d columns to yield, $\hat{L}, \hat{R}$ such that:

$$A \approx \hat{L}\hat{R}' \tag{14}$$

$$\hat{A} = \hat{L}\hat{R}' \tag{15}$$

The approximation of A is defined as $\hat{A}$ and is given by eq. (15) can be interpreted as a graph where the entry (i, j) of the matrix is the probability that an edge $(i \leftrightarrow j)$ exists. This approximation $\hat{A}$ is an RDPG[22].

Using an RDPG lets us model the probabilities of edges existing as continuous variables, as opposed to the discontinuous observations themselves. A benefit of using an SVD to create the RDPG, is that, to model the probabilities of $\hat{A}$, we can model the change of $\hat{L}$ and $\hat{R}$ separately. This is useful because the rows of the matrices $\hat{L}$ and $\hat{R}$ can be viewed as vector points in a d-dimensional space, where d is the number of singular values we take for our SVD truncation. Viewing $\hat{L}$ and $\hat{R}$ as a collection of vector points allows us to change the problem of modelling changing matrices to a problem of modelling a dynamical system. This modelling can be accomplished by finding an appropriate DE.

The SVD can be considered to give a latent space representation of the matrix[8]. That is, in this space similar nodes in the network are mapped to similar points in space. A visual representation of this process can be seen in fig. 1.

It is important to note that the SVD is not unique. The distances between points will be the same for every embedding, but the orientation may be different, and so, if the SVD of each time step is not aligned, learning will be effectively impossible as we will essentially be trying to model random noise.

Here, we can use Procrustes alignment[41] to align the matrices after they have been generated.

When obtaining the RDPGs for our temporal network, we take each network observation fig. 1 and represent them in the form of an adjacency matrix fig. 1a. We then use a truncated singular value decomposition to reduce the dimensionality of our adjacency matrices[3].

This truncation introduces some loss, but can dramatically reduce the size of our problem. Runghen, Stouffer, and Dalla Riva[36] effectively employ this method to reduce a problem involving over 130,000 visitors to a 6 dimensional latent space, whilst preserving 70% of the information.

To execute the SVD we use the ARPACK Julia package [5]. This package uses an iterative approach to approximate the eigenvectors and singular values of each of our adjacency matrices[4].

2. Neural Network Differential Equation

Once we have the SVD embedding of points, we look to model the trajectories of a target point. That is, with some input data, point location, distance to neighbours, etc, we want to model the differential equation that governs the movement of a point. Notably this method can be used to model multiple points by simply changing the node being predicted and trained on.

From eq. (8) our problem can be represented as the UDE:

$$\frac{d\hat{A}}{dt} = \hat{f}(t, \hat{A}(t)) + NN(t, \hat{A}, \theta) \tag{16}$$

The focus of this thesis is to understand the behaviour of the neural network when modelling temporal neural networks. Hence, we set $\hat{f}(t, \hat{A}(t)) = 0 \forall t$. With this, the

problem becomes:

$$\frac{d\hat{A}}{dt} = NN(t, \hat{A}, \theta) \tag{17}$$

A special case of a UDE called a Neural Network Differential Equation(NNDE).

In this thesis, as a proof of concept, we focus on a modelling the change in structure of a single node, referred to as the target node u, where $u(t) = {}_t\hat{L}_1.$. To model u we use the direction vectors from u to its k (15) nearest neighbours in $\hat{L}$. We define the function $K(\hat{L}(t), k)$ to find the direction vectors of the k nearest neighbours to $u(t)$. We use the function K to limit the complexity of our model.

$$\frac{du}{dt} = NN(t, K(\hat{L}, k), \theta) \tag{18}$$

Because we are only modelling the movement of a single node, we assume that we have access to the rest of the network structure at every time step for the prediction. This would likely not be the case in a real world application, but will allow us to observe some of the strengths and weaknesses of this framework in a controlled setting. If we wanted to predict the whole network we would extrapolate out every point and use the dot product of each point to predict the edges.

Here we train a neural network to approximate the function $g(u(t), t) = NN(u(t), t, \theta)$. Where θ is the parameters of the neural network.

To obtain a suitable set of parameters θ, we make use of the SciML ecosystem of packages in the Julia programming language[28].

3. Symbolic Regression

With our trained NNDE, we have a black box that approximates the system that governs the movement of the target node in the embedded space. To make this

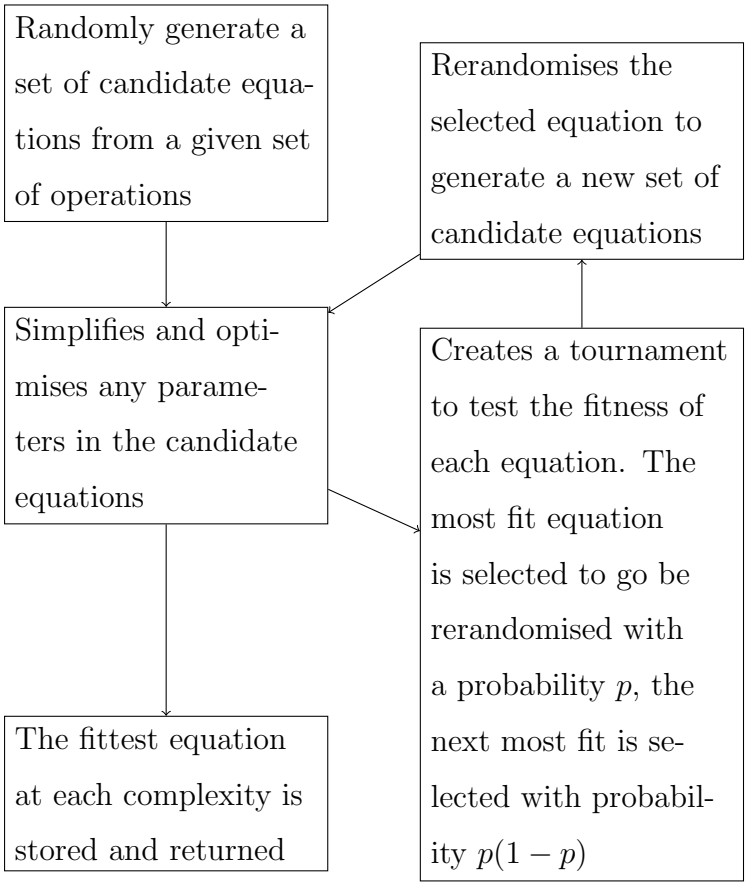


FIGURE 2. Illustration of the genetic programming symbolic regression process.

more interpretable, we then look to approximate this black box into a combination of known functions using symbolic regression[25].

We use the genetic programming algorithm found in [25]. The process for which can be seen in fig. 2.

The fitness of a candidate function is determined by comparing the predicted value to the given data, as well as the derivatives of the candidate equations to numerically found partial derivatives of the data variable.

For this process we need to define a set of functions which the algorithm will use to generate a tree of functions to find the best approximation to our NNDE with. When using this method we are usually required to make two assumptions: that we have paired observations of $u(t), \frac{du}{dt}$, and that the tree of expressions we will need is shallow. We will train the symbolic regression on the predictions of the neural network. By doing this, we can remove these assumptions[39].

4. Reconstructing the Temporal Network

u the first row of the matrix $\hat{L}$. By taking $u\hat{R}'$, this is equivalent to finding $\hat{A} = \hat{L}\hat{R}'$ and looking at $\hat{A}_{1\cdot}$. Hence, to recover a final prediction of the edges of u, we take:

$$p = u\hat{R}' \tag{19}$$

This gives a vector p of probabilities of connections between the target node and the rest of the network. As the results can be interpreted as a probability, the most likely graph is given by setting any entry with a value of 0.5 or higher to one (predict that the edge exists), and setting any entry with a value of less than 0.5 to zero (predict that an edge does not exist).

CHAPTER 4

Data

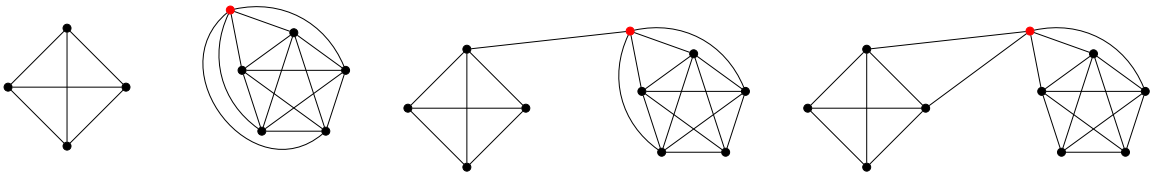
In this paper we simulate three temporal sequences of undirected graphs. For each sequence we will be modelling the node in red, we will refer to this as the target node. For each sequence we train the model on the first 25 time steps and compare the predictions for the next 15 time steps. For clarity, figs. 1, 3 and 4 have been simplified and illustrates the movement of the target node in each sequence.

Each model was trained on a neural network with 4 hidden layers (64,8,8,8).

1. Two Community System

The 2 community network has communities of size 40 and 50, each of which are fully connected. The target node is initially fully connected to the 50 node community. In our simplified diagram fig. 1a the 50 node community is represented by the 5 fully connected nodes, while the 40 node community is represented by the 4 node component. At each time step one edge is removed between the target node and the larger community and replaced with an edge between the target node and the smaller community fig. 1b. This process is then continued fig. 1c for thirty-five time steps to generate our training and test data. Once we have this data in the form of a sequence of adjacency matrices, we find the singular value decomposition using the framework described in section 1.

Figures obtained from [42].



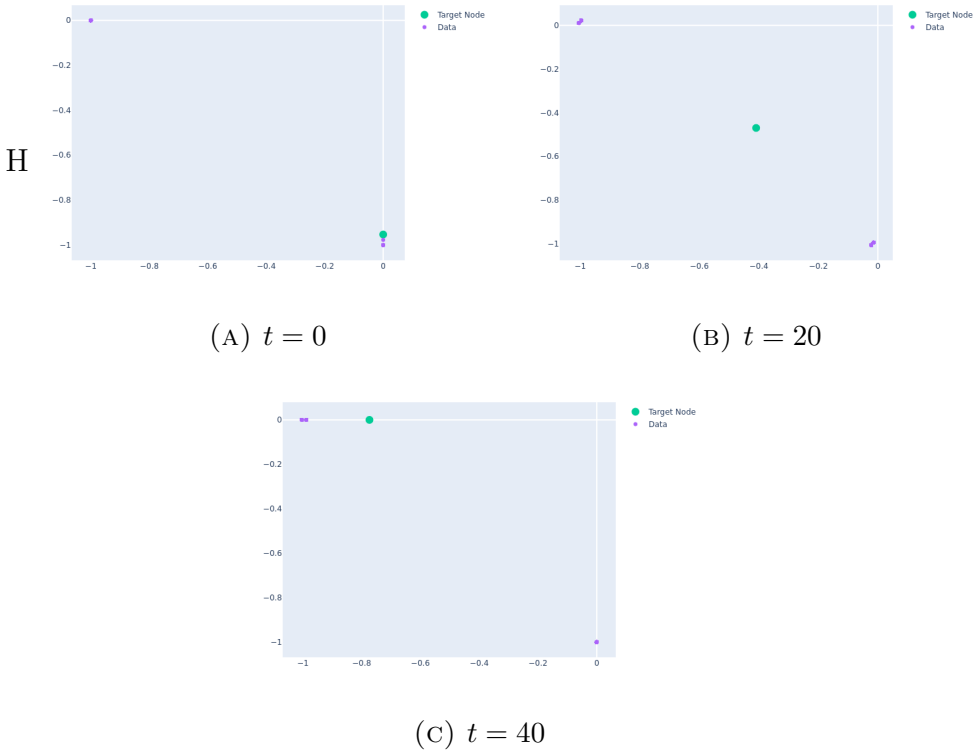
(A) $t = 1$ (B) $t = 2$ (C) $t = 3$

FIGURE 1. First 3 time steps of the simplified, synthetic 2 community network. The red node represents the target node that we are aiming to model. The 4 node and 5 node fully connected components represent the 40 and 50 node communities of the sequence respectively. At each time step, an edge from the target node to the larger community is replaced with and edge from the target node to the smaller community.

The 2 community temporal network was selected to test whether this framework can be used to model a node changing communities in systems where there are no other processes occurring. Notably, the movement in the two community system may be difficult for the neural network to learn because opposite sets of inputs need to result in the same output. In a simplified version of this system with only the nearest node as input, the input for the model begins as the position of the larger community at roughly $(0, -1)$ fig. 2. This input needs to move the target towards the top left. However, as the sequence progresses, the nearest neighbours will be the small community at roughly $(-1, 0)$, which also needs to move the target node towards the top left.

2. Long Tail System

In the long tail network fig. 3, we have a long chain with 50 nodes and a fully connected component with forty nodes at one end of the chain. This is used to orient the embedding and model. The SVD does not take into account the index of the

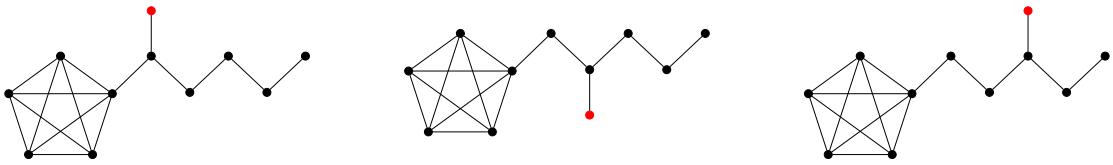


(A) $t = 0$ (B) $t = 20$

(C) $t = 40$

FIGURE 2. 2 Community test series. This series shows the movement of the target node within the SVD embedding.

individual node in the adjacency matrix; it only distinguishes based on their distance to each other. Because of this a chain of nodes, even when aligned, could be flipped when it is embedded. That is, between one time step and the next the first node in the chain could be aligned with the last node. With a large, connected group at one end however, that group will always be aligned with itself and so the rest of the chain will also be aligned properly. The target node starts attached to the node that is attached to this large component fig. 3a. At each time step the target node move one node further down the tail fig. 3b. Again, this is repeated to generate the training and test data fig. 3c.



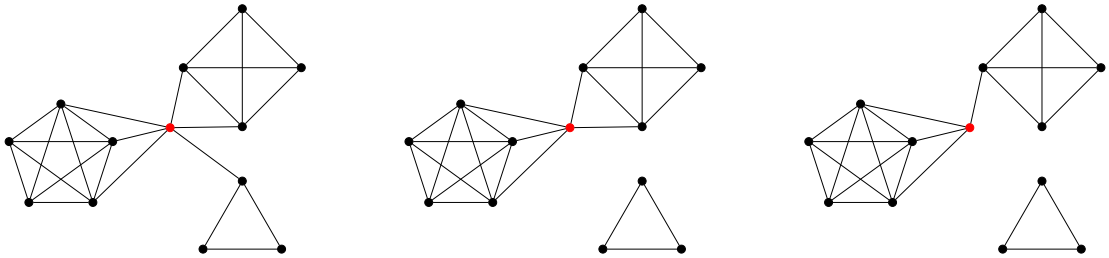
(A) $t = 1$ (B) $t = 2$ (C) $t = 3$

FIGURE 3. First 3 time steps of the simplified, synthetic long tail network. The red node represents the target node that we are aiming to model. The fully connected, 5 node component represents a 50 node community at the end of the long tail. At each time step the target node moves one step further along the chain, away from the large community.

The long tail network was selected because the SVD performs poorly on highly diagonal matrices; that is matrices with long chains. The long tail problem is difficult for the embedding because a large portion of the network is along the off diagonals; meaning that, to get an accurate embedding, the dimension of the embedding would need to be close to the length of the tail. As such we wish to see how the framework performs when it is presented with a system that the SVD cannot capture.

3. Three community System

We also simulate a sequence with three communities. The communities have sizes 40, 35, 30 and have 25, 24, and 23 edges to the target node respectively. In fig. 4 the communities have been simplified to have sizes of 5, 4, and 3. The target node starts connected to all of these communities with each community having a different number of edges fig. 4a. The community with the fewest edges between the target node will have one edge removed at each time step fig. 4b, and this process is repeated to create the required number of time steps fig. 4c.



(A) $t = 1$ (B) $t = 2$ (C) $t = 3$

FIGURE 4. First 3 time steps of the simplified, synthetic 3 community networks. The red node represents the target node which we aim to model. The 3, 4, and 5 node components represent the 30, 35, and 40 node communities respectively. At each time step the one edge is removed between the target node and the community it has the fewest connections with.

The three community was chosen as a best case scenario system to model. It has been constructed to avoid the challenges that are posed by the 2 community and long tail systems. That is, because in this system the target node is moving away from its least similar community, not moving from one community to another, it is not required to overcome the issue of symmetrical input. And because there are no long tails in this system, the SVD is able to capture a lot of the system with a low dimensional.

CHAPTER 5

Results and Discussion

1. Embedding Prediction

For each of these systems, we took embedded the temporal network in two dimensions at each time step. The temporal embeddings were then divided into a training set of 20 and a testing set of 15. The output of the trained neural networks was then used to train a symbolic regression model that could use a simple set of addition, subtraction, and multiplication.

We predicted the evolution of the 2 community, long tail, and 3 community systems for 15 time steps after the training period using both a NNDE model and a symbolic regression model.

To compare our models, we look at the distance of the predictions to the embedding of the target node. Note that this does not include the loss from the SVD, which con be reduced by increasing the number of singular values taken in for the SVD.

To find the predicted probability of the edges of node i, $\mathbb{P}(i \leftrightarrow j : j \in V \setminus \{i\})$, we use:

$$ {}_tA_{i,\cdot} \cong (\bar{p})_t\hat{R} \tag{20}$$

Where $\bar{p}$ is the predicted location of the target node i in the embedding at time t.

Sequence	Neural Network Prediction Loss	Symbolic Regression Prediction Loss
2 Community	0.224	0.145
Long Tail	0.093	0.331
3 Community	43.064	2.768

TABLE 1. A summary of the mean loss of the predictions at the fifth to ninth time steps. The neural network and symbolic prediction loss comes from the RDPG being reconstructed but with the position of the target node replaced with the location of the respective prediction. Loss is calculated as the mean squared error between the prediction predicted edge probability and the edge probability of the embedding.

To find calculate the loss of our prediction at time t, we use:

$$\bar{j} = V \setminus \{i\} \tag{21}$$

$$L = \frac{1}{||\bar{j}||} \sum ({}_t\hat{A}_{i,\cdot}\hat{R}_{\cdot,\bar{j}} - (\bar{p})_t\hat{R}_{\cdot,\bar{j}})^2 \tag{22}$$

That is the mean squared error between the probabilities given by the true embedding and the prediction.

When we do this for the SVD, NN, and symbolic regression, we get table 1.

As can be seen in table 1, the symbolic regression model performed better for both the 2 and 3 community system. However, for the long tail system the symbolic regression model performed notably worse.

The loss of one system should not be compared with another. That is the 2 community losses should not be compared to the long tail. This is because the long tail system will only ever have one edge to predict, whereas the 2 community will have 50 edges that need to be predicted.

Instead, we may compare predictions within the same system. However, we do not necessarily get a good understanding of the behaviour of our models with just this summary. We plot our predictions at every third time step to further understand the behavior of the models generated by our framework.

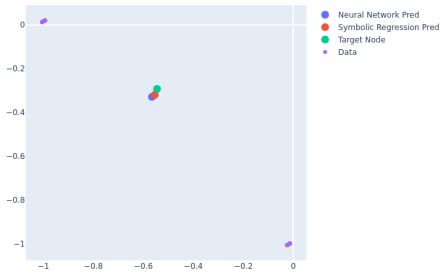
2. Further Exploration

In this section, we study the long tail, and 2 and 3 community systems. In figs. 1 to 3 we identify the embedded coordinates of each non-target node with a purple point in the Cartesian plane. Each cluster is one separate community. The green point identifies the true coordinate of the embedded target node at each time step. The blue point identifies the coordinate of the neural network model prediction at each time step, and the orange point identifies the symbolic regression prediction.

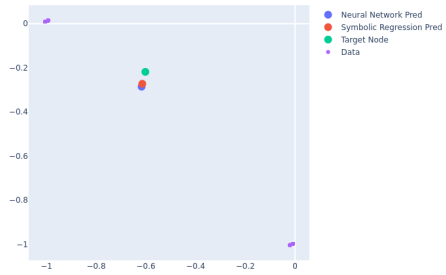
To illustrate how the changing structure of the network is represented in the embedding, we give a brief description of the movement of the target node throughout the 2 community system below.

In fig. 2, as the time progresses, we see the true node move from one community cluster to the other. This is because, at the beginning of the training phase when $(t = 0)$, the target node starts as very similar to the first community (as it has many connections with nodes in that community) and as time progresses, it gradually becomes less and less similar to the first community and more similar to the second (as the edges between the target node and the first community are replaced with edges to the second community). Because the SVD embeds nodes with many of the same connections close to each other, we see the target node move towards the second community.

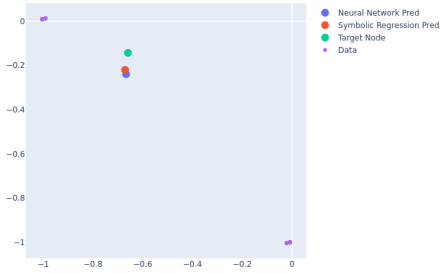
2.1. 2 Community. In fig. 1, we see both the neural network model and the symbolic regression models follow quite closely to the target node throughout the



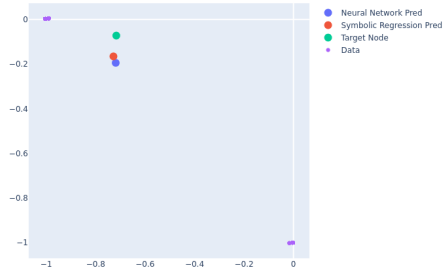
(A) $t = 28$



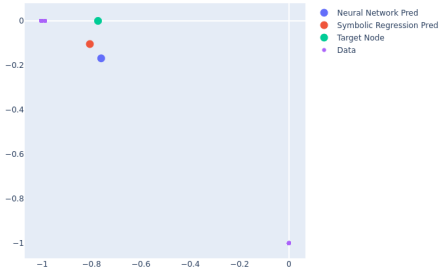
(B) $t = 31$



(C) $t = 34$



(D) $t = 37$



(E) $t = 40$

FIGURE 1. 2 Community test series. This series shows the comparison of the neural network model, the symbolic regression model trained on the neural network model, and the true solution of the two community system.

test process. As we saw in table 1, the NN model performs slightly worse than the symbolic regression model. This improvement of symbolic regression was present throughout the test period.

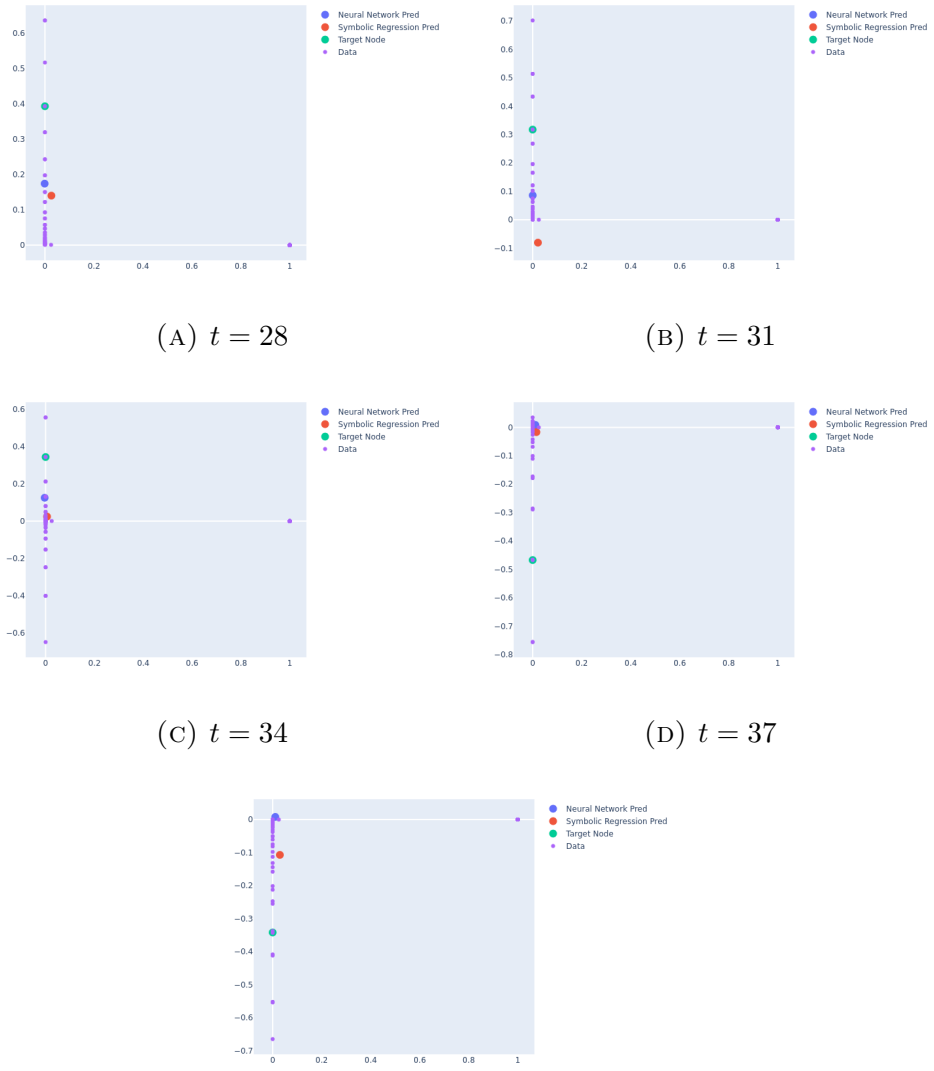
In spite of the simple movement of the target node, these close predictions were unexpected. When creating this system, we were looking to test how the model would perform with symmetrical input data. We hypothesised that this may be difficult for the NN model to learn, as it would need to learn to give the same output with opposite input.

In this case however, it seems that predicting the movement of the target node was not affected by this symmetry.

2.2. Long Tail. In fig. 2 we see the true embedding of the target node move from one arm to the other between the time steps 9 and 12. It is clear that neither the symbolic regression model nor the neural network model capture this movement.

From table 1, we see that the NN model had a higher accuracy than the symbolic regression. However, the NN model does not follow the true node; instead it stays in relatively the same spot, as opposed to the symbolic regression model, which does move.

We see that neither the neural network model capture the movement of the target node to any real extent. One reason for this may be that by the time the system progresses to the test section the number of nearest neighbours (k=15) means that the input for the model no longer includes the large community used to orient the system. Because of this, the model will essentially be given the same input at every time step, but will have to somehow produce the alternating behaviour of the target node. This again is an issue of symmetry.



(A) $t = 28$ (B) $t = 31$

(C) $t = 34$ (D) $t = 37$

(E) $t = 40$

FIGURE 2. Long Tail test series. This series shows the comparison of the neural network model, the symbolic regression model trained on the neural network model, and the true solution of the long tail system.

To solve this problem we could break the symmetry by including the absolute position of the target node as input for the model, or to have multiple time steps as input.

Another potential reason for the poor behavior could be that the embedding of the temporal network does not contain enough information to train a NNDE model. This is discussed in section 2.

2.3. Three Community. In fig. 3 we see the symbolic regression model move in roughly the same direction as the target node. The NN model however, moves quickly towards the bottom left, away from the target node.

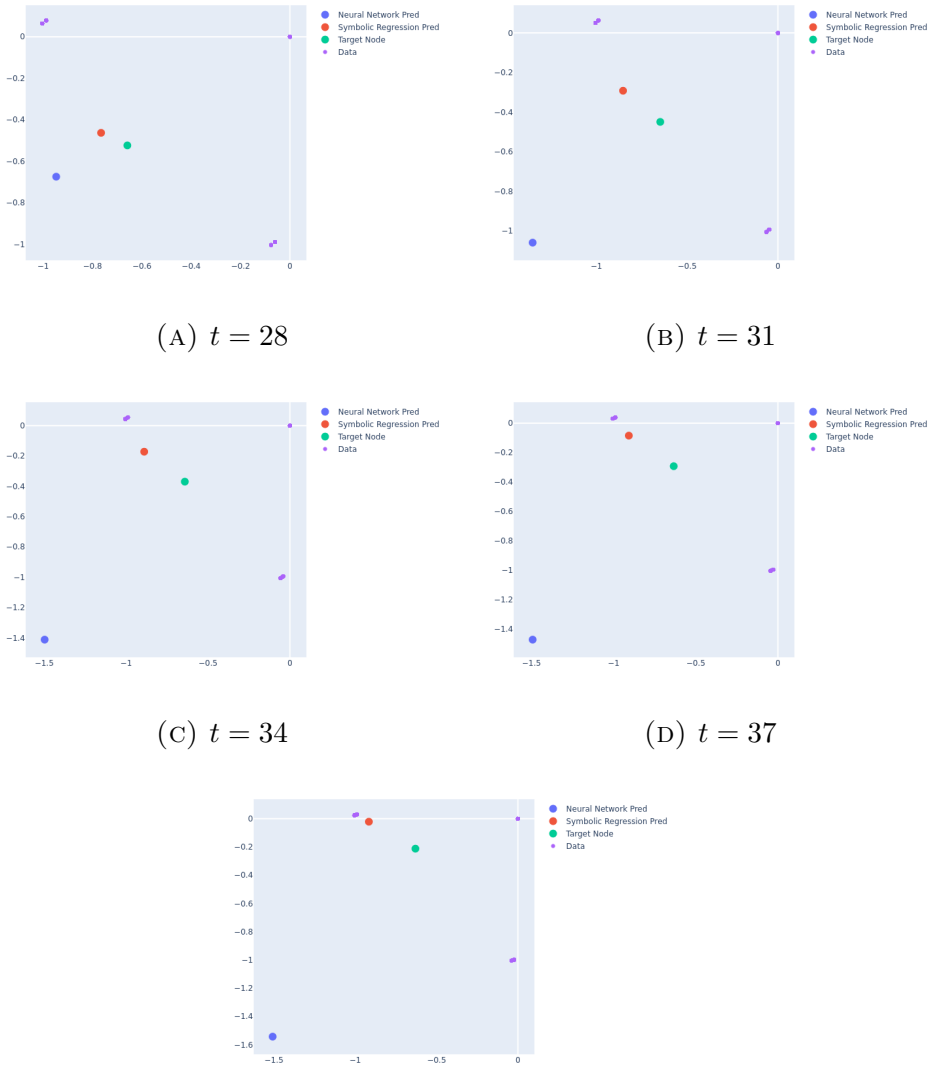
In contrast, after the third time step, the symbolic regression model prediction maintains a relatively constant distance to the target node.

This system was constructed to avoid the symmetrical input of the other two systems. Because of this, it is interesting to see the poor predictive performance of the NN model.

We see that the symbolic regression model significantly improved the predictions, which supports the notion that symbolic regression, can give more robust predictive models than a NNDE alone.

3. Summary

In the 2 and 3 community systems, we see that symbolic regression improved the accuracy of the predictions across all time steps when compared to the NN model. This is especially prominent in the 3 community system. In the long tail system, where the symbolic regression model performed worse than the NN model, neither performed well, with the NN model remaining in relatively the same spot throughout the test period.



(A) $t = 28$ (B) $t = 31$

(C) $t = 34$ (D) $t = 37$

(E) $t = 40$

FIGURE 3. 3 community test series. This series shows the comparison of the neural network model, the symbolic regression model trained on the neural network model, and the true solution of the 3 community system.

CHAPTER 6

Code

1. Summary

The idea of this code is to provide a foundation for a streamline research into temporal network modelling. To that end it is planned to be compatible with popular Julia packages such as Graphs, the SciML ecosystem, and EcologicalNetworks.

The core of the code developed for this project is the TemporalNetworkEmbedding (TNE) structure. This structure was developed not only for me to be able to interface with data quickly and easily, but also so the SciML ecosystem could interface with it during training of the NN. In particular, the ability to index the TNE with a float was essential for the continuous time evolution of the UDEs and NNDEs to function.

2. Statement of Need

Throughout this thesis, we have demonstrated that temporal networks are found in many different areas, that, on small networks, temporal progression of embedded nodes can be modelled with NNDEs, and that there are still problems to be overcome.

With this in mind, the code has been developed to:

1: Allow for interface with popular packages

2: Allow for further exploration of optimal network structures for deep learning

3: Provide opportunities for collaboration of experts from other fields

3. Package Methods

In the package [41], I created code for constructing: The TemporalNetworkEmbedding structure. This structure stores the temporal network as tensors of the $\hat{L}, \hat{R}$ matrices. This structure has many methods for streamlining interactions with the TemporalNetworkEmbedding. These include indexing with integers and floats, where the floats will linearly interpolate between time steps. Other methods include viewing and or removing specific nodes from the network.

As well as constructRDPG, which takes a temporal network embedding and optionally the time steps of interest, and outputs the RDPG from the embedding.

I also added the nearestNeighbours method which takes a vector point in the dimension of the embedding, the time step, a TemporalNetworkEmbedding and the number of nearest neighbours required. The method then outputs the indices of the nearest nodes of the network to the vector in descending order.

4. Future Work

The current TNE structure is not yet GPU compatible. We plan to remedy this as well as include support for constructing a TNE from Graph.jl and MetaGraph.jl objects.

CHAPTER 7

Conclusion

In this thesis, we proposed a novel framework for modelling temporal networks. We then tested this framework on three types of small, synthetic network sequences. Throughout these tests we showed that an NNDE model can be used to approximate the movement of points in the embedded space of some temporal networks. We also demonstrated that a functional expression generated using symbolic regression can increase the accuracy of prediction on unseen data. This has been a proof of concept for this framework; we believe future work can dramatically improve the understanding and modelling of this framework.

Bibliography

[1] Vito Volterra. *Variazioni e fluttuazioni del numero d'individui in specie animali conviventi*. Vol. 2. Societá anonima tipografica" Leonardo da Vinci", 1927.

[2] George E Forsythe and Cleve B Moler. "Computer solution of linear algebraic systems". In: (1967).

[3] Gene H Golub and Christian Reinsch. "Singular value decomposition and least squares solutions". In: *Linear algebra*. Springer, 1971, pp. 134–151.

[4] Richard B Lehoucq and Danny C Sorensen. "Deflation techniques for an implicitly restarted Arnoldi iteration". In: *SIAM Journal on Matrix Analysis and Applications* 17.4 (1996), pp. 789–821.

[5] Richard B Lehoucq, Danny C Sorensen, and Chao Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, 1998.

[6] Charles S Elton. *Animal ecology*. University of Chicago Press, 2001.

[7] Mark EJ Newman. "Clustering and preferential attachment in growing networks". In: *Physical review E* 64.2 (2001), p. 025102.

[8] Peter D Hoff, Adrian E Raftery, and Mark S Handcock. "Latent space approaches to social network analysis". In: *Journal of the american Statistical association* 97.460 (2002), pp. 1090–1098.

[9] Judith D Singer, John B Willett, John B Willett, et al. *Applied longitudinal data analysis: Modeling change and event occurrence*. Oxford university press, 2003.

[10] Andrea Capocci, Vito DP Servedio, Francesca Colaiori, Luciana S Buriol, Debora Donato, Stefano Leonardi, and Guido Caldarelli. "Preferential attachment in the growth of social networks: The internet encyclopedia Wikipedia". In: *Physical review E* 74.3 (2006), p. 036116.

[11] Josh Bongard and Hod Lipson. "Automated reverse engineering of nonlinear dynamical systems". In: *Proceedings of the National Academy of Sciences* 104.24 (2007), pp. 9943–9948.

[12] Sanchit Garg, Trinabh Gupta, Niklas Carlsson, and Anirban Mahanti. "Evolution of an online social aggregation network: an empirical study". In: *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*. 2009, pp. 315–321.

[13] Ruoming Jin, Scott McCallen, Chun-Chi Liu, Yang Xiang, Eivind Almaas, and Xianghong Jasmine Zhou. "Identifying dynamic network modules with temporal and spatial constraints". In: *Biocomputing 2009*. World Scientific, 2009, pp. 203–214.

[14] Michael Schmidt and Hod Lipson. "Distilling free-form natural laws from experimental data". In: *science* 324.5923 (2009), pp. 81–85.

[15] Steve Hanneke, Wenjie Fu, and Eric P Xing. "Discrete temporal models of social networks". In: (2010).

[16] Birkan Can and Cathal Heavey. "Comparison of experimental designs for simulation-based symbolic regression of manufacturing systems". In: *Computers & Industrial Engineering* 61.3 (2011), pp. 447–462.

[17] Xiaohan Zhao, Alessandra Sala, Christo Wilson, Xiao Wang, Sabrina Gaito, Haitao Zheng, and Ben Y Zhao. "Multi-scale dynamics in a massive online social network". In: *Proceedings of the 2012 Internet Measurement Conference*. 2012, pp. 171–184.

[18] Fariba Karimi and Petter Holme. "Threshold model of cascades in empirical temporal networks". In: *Physica A: Statistical Mechanics and its Applications* 392.16 (2013), pp. 3476–3483. ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2013.03.050. URL: https://www.sciencedirect.com/science/article/pii/S0378437113002835.

[19] Naoki Masuda and Petter Holme. "Predicting and controlling infectious disease epidemics using temporal networks". In: *F1000prime reports* 5 (2013).

[20] Kosuke Yoshihara, Maria Shahmoradgoli, Emmanuel Martínez, Rahulsimham Vegesna, Hoon Kim, Wandaliz Torres-Garcia, Victor Treviño, Hui Shen, Peter W Laird, Douglas A Levine, et al. "Inferring tumour purity and stromal and immune cell admixture from expression data". In: *Nature communications* 4.1 (2013), p. 2612.

[21] Antoine Moinet, Michele Starnini, and Romualdo Pastor-Satorras. "Burstiness and aging in social temporal networks". In: *Physical review letters* 114.10 (2015), p. 108701.

[22] Avanti Athreya, Donniell E Fishkind, Minh Tang, Carey E Priebe, Youngser Park, Joshua T Vogelstein, Keith Levin, Vince Lyzinski, and Yichen Qin. "Statistical inference on random dot product graphs: a survey". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 8393–8484.

[23] Nikolaos Papachristou, Payam Barnaghi, Bruce Cooper, Kord M Kober, Roma Maguire, Steven M Paul, Marilyn Hammer, Fay Wright, Jo Armes, Eileen P Furlong, et al. "Network analysis of the multidimensional symptom experience of oncology". In: *Scientific reports* 9.1 (2019), p. 2258.

[24] Alba Contreras, Carmen Valiente, Alexandre Heeren, and Richard Bentall. "A temporal network approach to paranoia: A pilot study". In: *Frontiers in psychology* 11 (2020), p. 544565.

[25] Miles Cranmer. *PySR: Fast & Parallelized Symbolic Regression in Python/Julia*. Sept. 2020. DOI: 10.5281/zenodo.4041459. URL: http://doi.org/10.5281/zenodo.4041459.

[26] Raj Dandekar and George Barbastathis. "Quantifying the effect of quarantine control in Covid-19 infectious spread using machine learning". In: *medRxiv* (2020). DOI: 10.1101/2020.04.03.20052084. eprint: https://www.medrxiv.org/content/early/2020/04/06/2020.04.03.20052084.full.pdf. URL: https://www.medrxiv.org/content/early/2020/04/06/2020.04.03.20052084.

[27] D Gage Jordan, E Samuel Winer, and Taban Salem. "The current status of temporal network analysis for clinical science: Considerations as the paradigm shifts?" In: *Journal of clinical psychology* 76.9 (2020), pp. 1591–1612.

[28] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, and Ali Jasim Ramadhan. "Universal Differential Equations for Scientific Machine Learning". In: *CoRR* abs/2001.04385 (2020). arXiv: 2001.04385. URL: https://arxiv.org/abs/2001.04385.

[29] Manuel A Roehrl, Thomas A Runkler, Veronika Brandtstetter, Michel Tokic, and Stefan Obermayer. "Modeling system dynamics with physics-informed neural networks based on lagrangian mechanics". In: *IFAC-PapersOnLine* 53.2 (2020), pp. 9195–9200.

[30] Han Gao, Luning Sun, and Jian-Xun Wang. "PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain". In: *Journal of Computational Physics* 428 (2021), p. 110079. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2020.110079. URL: https://www.sciencedirect.com/science/article/pii/S0021999120308536.

[31] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. "Physics-informed machine learning". In: *Nature Reviews Physics* 3.6 (2021), pp. 422–440.

[32] John A Keith, Valentin Vassilev-Galindo, Bingqing Cheng, Stefan Chmiela, Michael Gastegger, Klaus-Robert Müller, and Alexandre Tkatchenko. "Combining machine learning and computational chemistry for predictive insights into chemical systems". In: *Chemical reviews* 121.16 (2021), pp. 9816–9872.

[33] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. "Characterizing possible failure modes in physics-informed neural networks". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 26548–26560.

[34] Maxime Lucas, Arthur Morris, Alex Townsend-Teague, Laurent Tichit, Bianca Habermann, and Alain Barrat. "Inferring cell cycle phases from a partially temporal network of protein interactions". In: *Cell Reports Methods* (2021).

[35] Mohammadamin Mahmoudabadbozchelou, Marco Caggioni, Setareh Shahsavari, William H Hartt, George Em Karniadakis, and Safa Jamali. "Data-driven physics-informed constitutive metamodeling of complex fluids: A multifidelity neural network (MFNN) framework". In: *Journal of Rheology* 65.2 (2021), pp. 179–198.

[36] Rogini Runghen, Daniel B Stouffer, and Giulio V Dalla Riva. "Exploiting node metadata to predict interactions in large networks using graph embedding and neural networks". In: *bioRxiv* (2021). DOI: 10.1101/2021.06.10.447991. eprint: https://www.biorxiv.org/content/early/2021/06/11/2021.06.10.447991.full.pdf. URL: https://www.biorxiv.org/content/early/2021/06/11/2021.06.10.447991.

[37] Francesco Sanna Passino, Anna S Bertiger, Joshua C Neil, and Nicholas A Heard. "Link prediction in dynamic networks using random dot product graphs". In: *Data Mining and Knowledge Discovery* 35.5 (2021), pp. 2168–2199.

[38] Elaheh Kalantari, Samaneh Kouchaki, Christine Miaskowski, Kord Kober, and Payam Barnaghi. "Network analysis to identify symptoms clusters and temporal interconnections in oncology patients". In: *Scientific Reports* 12.1 (2022), p. 17052.

[39] Patrick Kidger. "On neural differential equations". In: *arXiv preprint arXiv:2202.02435* (2022).

[40] Thi Nguyen Khoa Nguyen, Thibault Dairay, Raphaël Meunier, and Mathilde Mougeot. "Physics-informed neural networks for non-Newtonian fluid thermo-mechanical problems: an application to rubber calendering process". In: *arXiv preprint arXiv:2201.13389* (2022).

[41] Smith C Dalla Riva G. *DotProductGraphs.jl*. https://github.com/gvdr/DotProductGraphs.jl. 2023.

[42] E. Hogan. *Tikz Diagrams*. https://github.com/roonil-wazlib/tikz_diagrams. 2023.