1 Introduction

The modelling of temporal networks continues to be an important and unresolved problem in the field of network analysis. Discovering the underlying equations of these dynamical systems proves to be an even more challenging problem. This paper proposes a process with which we can find a set of equations that allow us to predict the how a temporal network will behave.

Nodes within the network influence eachother and so we might wish to draw on the rich literature of differential equation modelling, which have been useful in modelling inumerable dynamical systems. Attempting to directly model the temporal adjacency matrix presents another challenge; changes in the network are discrete jumps, not smooth curves which can be nicely modelled by differential equations.

We propose using random dot product graphs, which are well established in statistics, and singular value decomposition to resolve this issue[citation]. Creating a singular value decomposition of each of our network observations allows us to view the temporal network as a sample, from which we can model the dynamics of edge existence probabilities across time, now a continuous function. Because the progression of the system can now be viewed as a continuous transformation, we can now look to model the change of this system. To regenerate the network from the embedding, we can simply multiply our embedding matrices together. This will give us a random dot product graph, where the entries (i,j) of the matrix will be the probability of an edge from i to j existing[citation]. In this form, the problem can be viewed as a dynamical system We believe this approach will be useful when applied to areas of medicine, especially protein interaction networks; population dynamics for network ecology; and social network modelling.

By using this process, we also open the possibility of using the endlessly flexible toolbox of machine learning. In this paper we explore the usefulness of neural ODEs, a neural network that is trained to differential equation of temporal data, for discovering the governing function of a set of simple, synthetic temporal networks. This results in a black box DE, which is not very helpful for understanding our systems. To give more insight into what is really going on, we then use symbolic regression to find a function whose output is similar to that of the neural network. By doing this we may further understand the important factors that govern our temporal network.

2 Methodology

We used a singular value decomposition of each matrix to embed our network in a low dimensional space.

$$A = U\Sigma V'$$

Where $\Sigma_{ii} = \sigma_i$ and $\sigma_i > sigma_{i+1}$ and each σ_i is a non-negative square root of the matrices $AA', A'A$. U is a matrix made from the orthonormalised eigenvectors of AA' associated with σ_i, and V is a matrix made from the orthonormalised eigenvectors of $A'A$ associated with σ_i. Each row i of U and V represents the in and out position of node i respectively. This can be truncated at an arbitrary number of eigenpairs to form a low dimentional approximation to A.

$$A \approx \hat{U}\hat{\Sigma}\hat{V}'$$

Where $\hat{U}, \hat{\Sigma}, \hat{V}$ are the first d columns of U, Σ, V [1]. $\hat{U}\sqrt{\hat{\Sigma}}$ is still associated with the in edges of our nodes, and $\sqrt{\hat{\Sigma}}\hat{V}'$ is still associated with the out edges of our nodes, but now we have simplified our system whilst keeping as much information as possible due to our descending order of singular values. These two matrices form our d dimensional embedding.

The approximation given by this formula can be interpreted as a graph where the entry (i, j) of the matrix are the probability that an edge exists from $i \to j$.This approach creates a random dot product graph[3], a class of Latent Position Models [2]. Finding the embedding is not trivial; we need to find the eigenpairs of each of AA' and $A'A$. Because of the computational complexity we use the SVD function in Arpack.jl to approximate each matrix. This then changes the problem of discovering edge formation and destruction into learning a dynamical system.
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 alligned, learning will be effectively impossible [¡- show this]. Since our approximation method is an iterative approach, to align each SVD, we can simply pick the same initial vector guess for each [probably worth getting a reference].

Once we have this embedding of points, we look to model the trajectories of each point. That is, with some input data, point location, distance to neighbours, etc, we want to model the ODE of each point. To do this we make use of the SciML ecosystem of packages in the Julia programming language [4]. To explore which architectures perform best for this system we then test various types.

As a way of limiting the complexity of our model, we limit the input of the neural network to the euclidean distances between the target node and its k nearest neighbours. The target node is the node that we are looking to model the movement of and stays the same throughout each of the temporal networks.

$$\begin{aligned} \mathbb{P}(i \to j) &\coloneqq L_i \cdot R_j \\ \mathbb{P}(\hat{i} \to j) &\coloneqq L_{\hat{i}} \cdot R_j \end{aligned} \tag{1}$$

Where (i, j) are nodes in the network and $\hat{i}$ is another node at the same timestep to i. This implies that the distance between these two nodes with respect to node j is given by

$$\begin{aligned} ||\mathbb{P}(i \to j) - \mathbb{P}(\hat{i} \to j)||_2 &= ||L_i \cdot R_j - L_{\hat{i}} \cdot R_j||_2 \\ &= ||(L_i - L_{\hat{i}}) \cdot R_j||_2 \end{aligned} \tag{2}$$

If we look at the expectation of this, we see

$$\mathbb{E}_j(||(L_i - L_{\hat{i}}) \cdot R_j||_2) = ||L_i - L_{\hat{i}} \cdot \frac{1}{k} \sum_{j \neq i} R_j||_2 \tag{3}$$

Where k is the number of neighbours we chose for the neural network input. Because the distance between points is discribed by the euclidean distance, we use this as our loss metric.

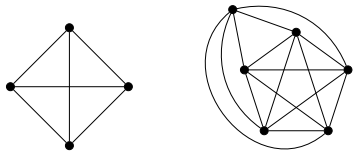
!!expand on sym reg!! With our trained NODE, we have a black box that approximates the system that governs the movement of Twitter users. To make this more interpretable we then look to approximate this black box into a combination of known functions using symbolic regression[citation (use the one from symreg package)]. 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 NODE with. When using this method we usually are required to make two assumptions: that we have paired observations of $y(t)$, $\frac{dy}{dt}$, 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[5].

3 Data

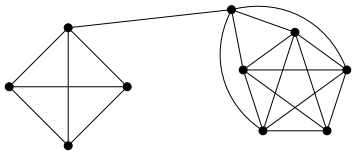
In this paper we explore this process using three synthetic networks, one which consists of the node we are interested in modelling, the target node, transitioning from one large group to another large group. In the other network the target node slowly moves along a long chain of connected nodes.

In the 2 community network the communities are of size 40 and 50 and are each fully connected. The target node is initially fully connected to the 50 node community; at each time step one edge is removed between the target node and the 50 node community and replaced with an edge between the target node and the 40 node community 1. This temporal network has 40 time steps in total, we use the first 20 to train the neural network and the last 20 to test it.

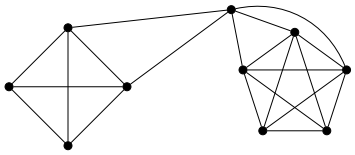
Figures obtained from [6].



(a) Simplification of the 2 Community network at $t = 1$.



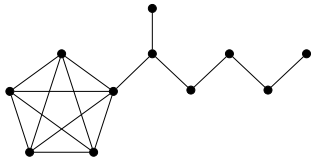
(b) Simplification of the 2 Community network at $t = 2$.



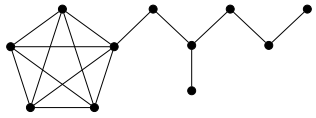
(c) Simplification of the 2 Community network at $t = 3$.

Figure 1: First 3 time steps of synthetic 2 community network.

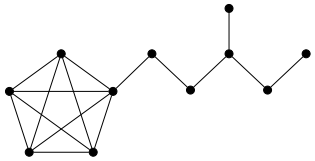
For the long tail network we have a large (50 nodes) fully connected component at one end. This is used to orient the embedding and model. The target node starts attatched to the node that is attatched to this large componet. At each time step the target node move one node further down the tail.



(a) Simplification of the long tail network at $t = 1$.



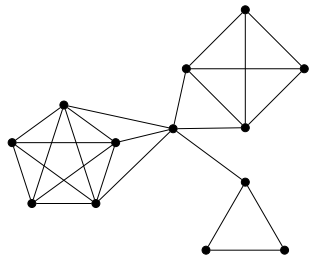
(b) Simplification of the long tail network at $t = 2$.



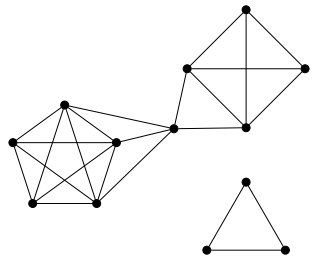
(c) Simplification of the long tail network at $t = 3$.

Figure 2: First 3 time steps of synthetic long tail network.

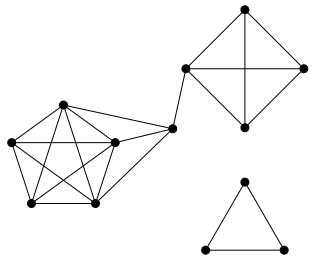
We also create a set of three communities. The target node starts connected to all of these communities with each community having a different number of edges. The community with the fewest edges between the target node will have one edge removed at each timestep. The communities have sizes 40, 35, 30 and have 25, 24, and 23 edges to the target node respectively.



(a) Simplification of the 3 Community network at $t = 1$.



(b) Simplification of the 3 Community network at $t = 2$.



(c) Simplification of the 3 Community network at $t = 3$.

Figure 3: First 3 time steps of synthetic 3 community network.

The 2 community temporal network was selected to illustrate that this process can be used to model a node changing communities in ideal conditions, and the long tail network was selected because we wanted to present these models with a range of problems. The two community problem is difficult for the neural network to learn because the same set of distances as input need to result in different outputs. 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. The three community problem should be the easiest to solve because it has been dileberately constructed to avoid the problems the other two models face.

4 Results

For each of theses 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, division, and multiplication.

4.1 2 Community

In fig 4 the green points are the embedded coordinates of the node in each of the communities. Each cluster is one separate community. The orange point is the true coordinate of the embedded target node at each time step. The blue point is our model prediction from the true target node at the first time step. As the time progresses, we see the true node move from one community cluster to the other. This is expected from the embedding; 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). As such, we see the target node move towards the second community.

In this case we see the symbolic regression overshoot the location of the target node and move far ahead of its location by the end of the training data. In contrast, the neural network moves away from the target node at first, then maintain that distance throughout the training data. This system was set up and trained multiple times, and each time we saw the symbolic regression model behave more erratically than the neural network model predictions.

4.2 Long Tail

In fig 5 we see the true embedding of the target node jump from one arm to the other at each timestep. It is clear that neither the the symbolic regression model nor the neural network model capture this movement, but the symbolic regression model does not move away from the data as quickly as the neural network model does.

In contrast to this, the symbolic regression model remains close to the true target node throughout the test data.

4.3 Three Community (Please Review)

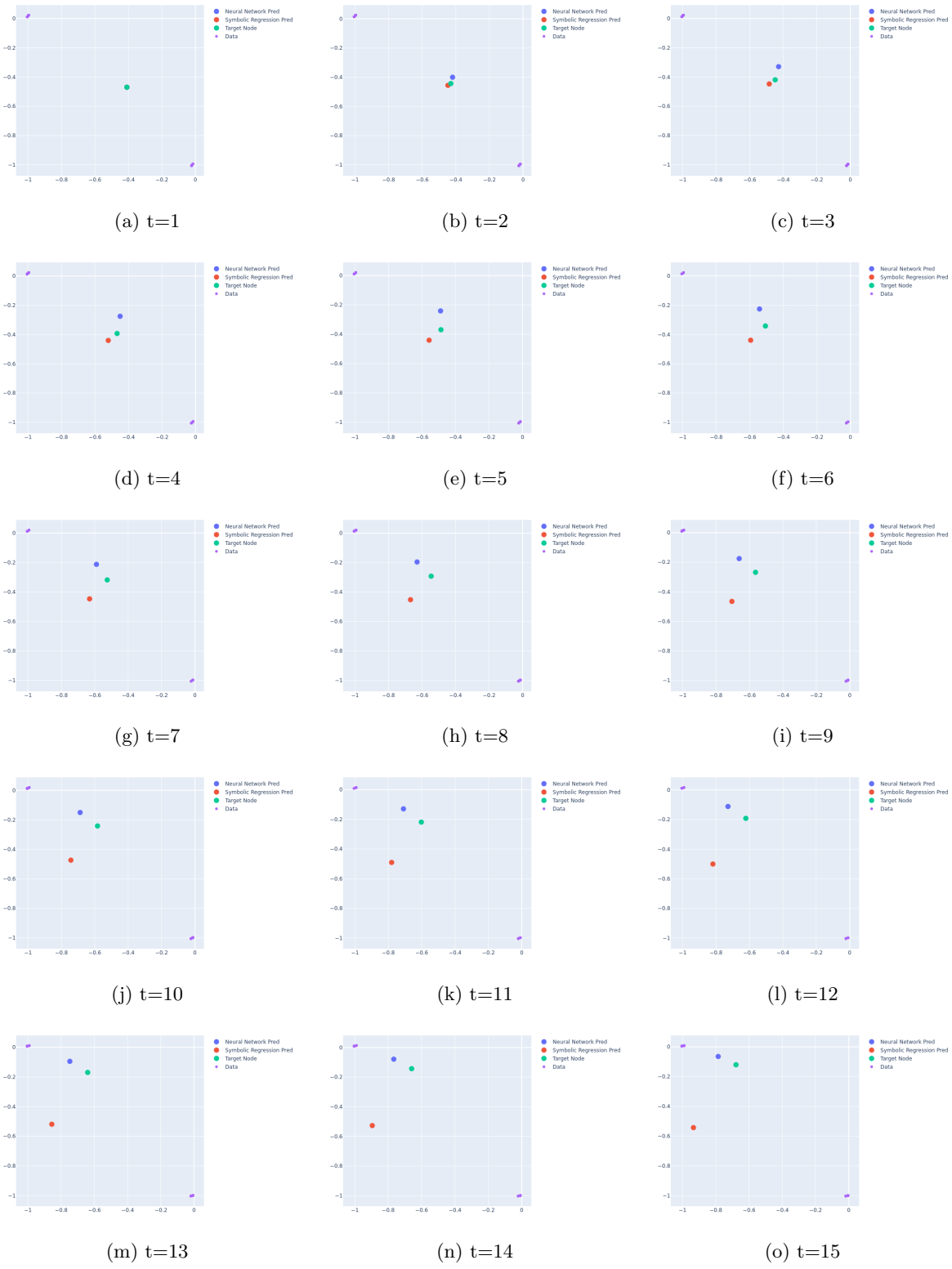
In fig 6 we again, as the edges between the community with the fewest edges are removed, we see the target node move away from it in the embedding and towards the other two clusters of points (communities).

In the neural network only model, we see the prediction follow the true target node very closely at first. We then see the node drift further and further from the true target node. In contrast, the symbolic regression model remains much closer to the true position of the target node as the test data progresses. We see the symbolic regression move slightly ahead of the true location at first, but this distance remains relatively constant thereafter.

5 Discussion

5.1 Two Community

It is interesting to see the symbolic regression model perform so much more poorly than the neural network model in this system. When we view the training section of the data, we see that there is a downward curve in the neural network predictions towards the end of the training data. It seems that the symbolic regression model picked this up and continued the downward trajectory throughout the test data. This downward curve may be because the problem that we are trying to solve is symmetrical. That is the same set of distances as input into the neural network, the target node needs to move away from its nearest neighbours in the first half, then move towards its nearest neighbours in the second half. Potential ways of breaking this symmetry



(a) t=1 (b) t=2 (c) t=3

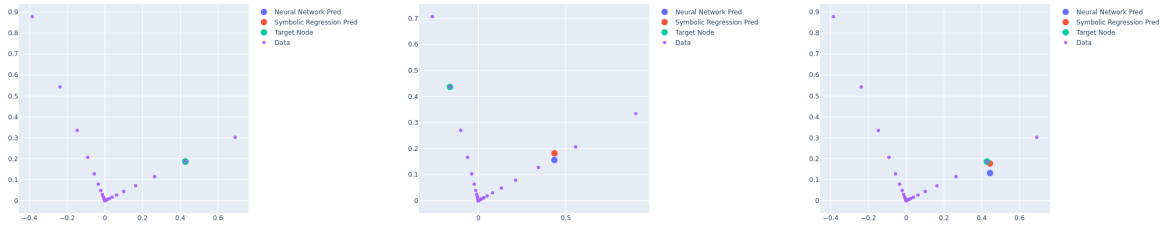
(d) t=4 (e) t=5 (f) t=6

(g) t=7 (h) t=8 (i) t=9

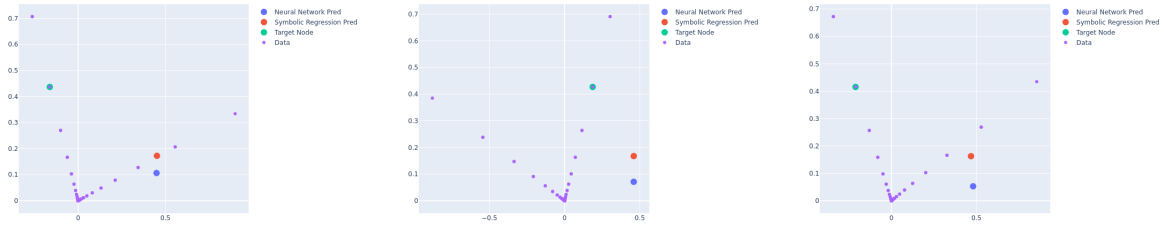
(j) t=10 (k) t=11 (l) t=12

(m) t=13 (n) t=14 (o) t=15

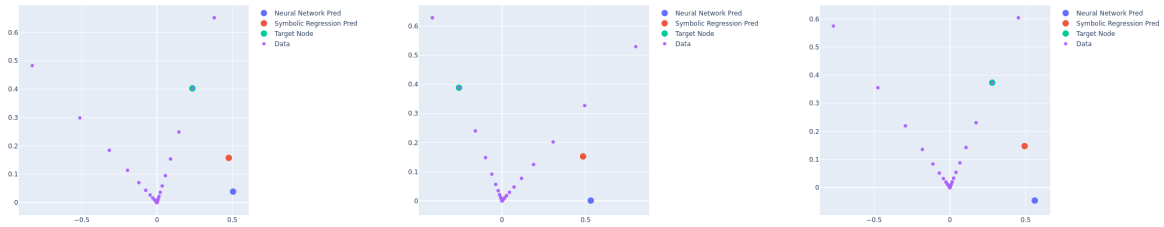
Figure 4: 2 Community test series. This series shows the comparison of the neural network model, the symbolic regression model trained on the neural network model, and the true solution of the two community system.



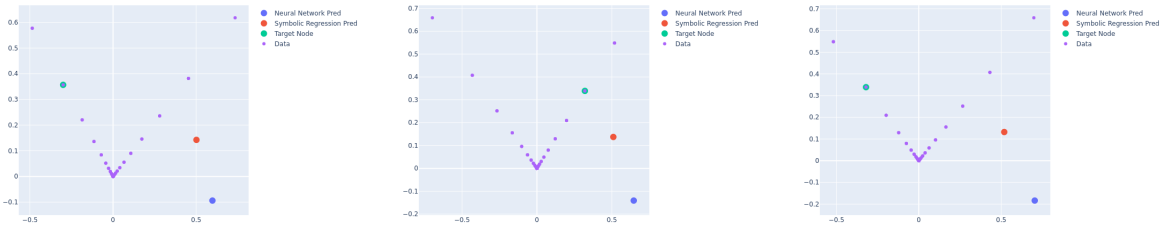
(a) t=1 (b) t=2 (c) t=3



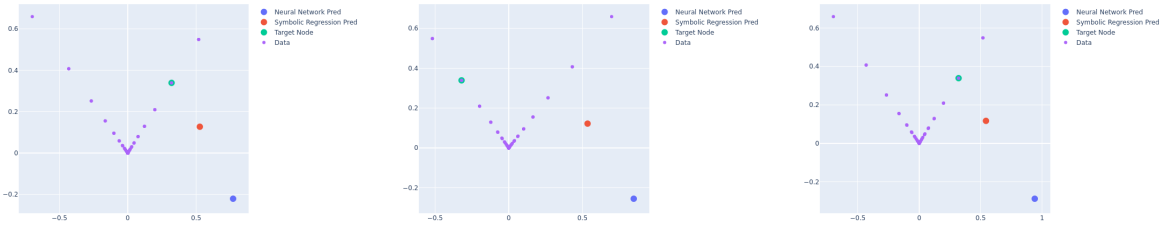
(d) t=4 (e) t=5 (f) t=6



(g) t=7 (h) t=8 (i) t=9

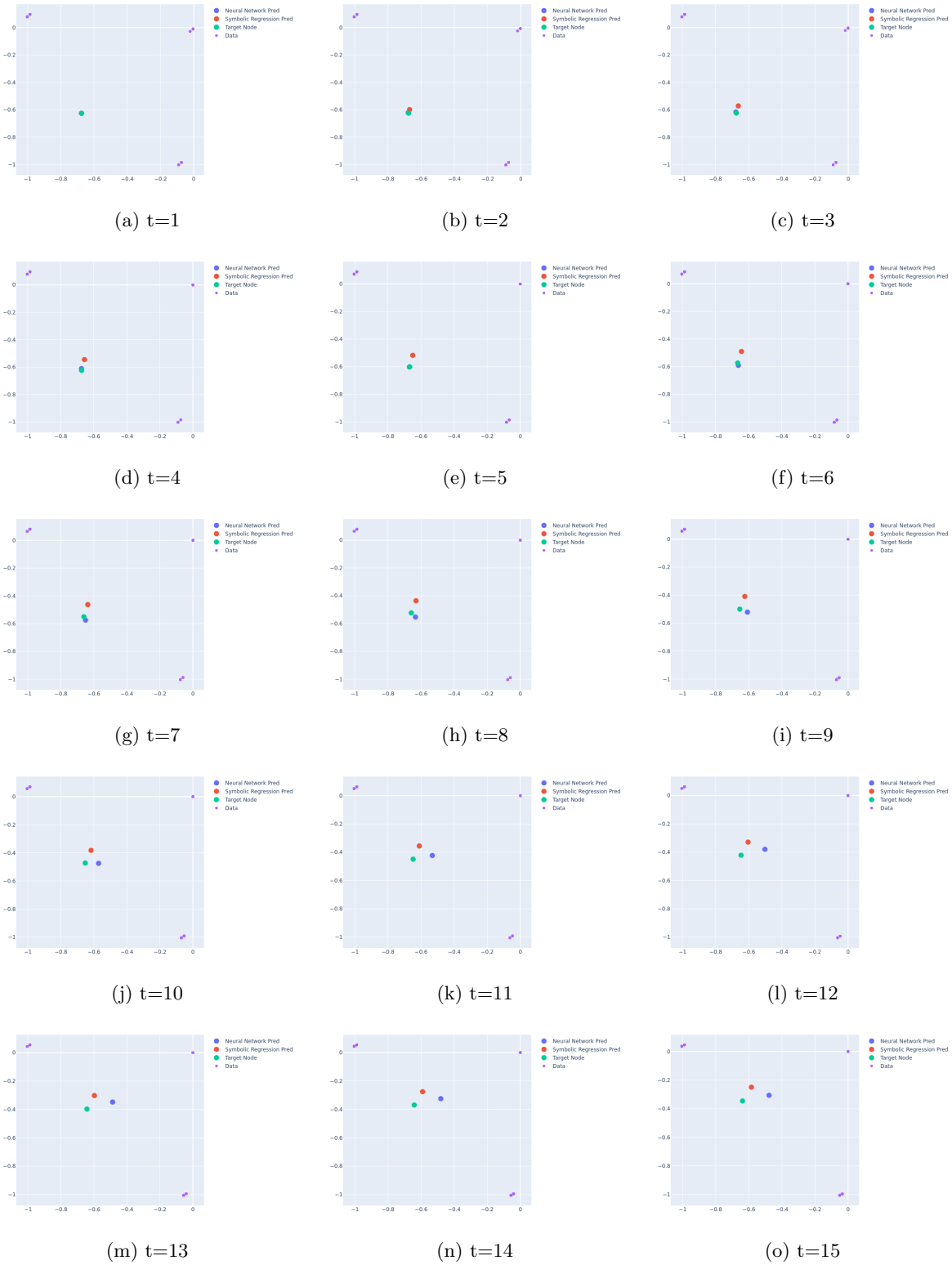


(j) t=10 (k) t=11 (l) t=12



(m) t=13 (n) t=14 (o) t=15

Figure 5: 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 long tail system.



(a) t=1 (b) t=2 (c) t=3

(d) t=4 (e) t=5 (f) t=6

(g) t=7 (h) t=8 (i) t=9

(j) t=10 (k) t=11 (l) t=12

(m) t=13 (n) t=14 (o) t=15

Figure 6: 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 3 community system.

may be to include information of the previous timestep, or to include the absolute position of the target node as input into the neural network.

5.2 Long Tail

We see that neither the neural network model capture the movement of the target node to any real extent. One reason for this may again be an issue of symmetrical input. The distances from the k nearest neighbours to the target node do not seem to change much as the target node jumps from one arm to the other, but the neural network somehow needs to learn to jump left then right at alternating timesteps. The solutions to this would be the same as for the two community system.

5.3 Three Community

This was by far the largest improvement between the neural network and the symbolic regression. This could be because this system had the least difference between the training predictions and the true position of the target node. At the end of the test data we see the true target node move towards the origin. This is because it now only has edges to one community and so at each timestep it removes one of these. If this had continued, the target node would have no edges. Nodes with no edges are mapped to the origin. When this happened, the symbolic regression model prediction began to move further from the true node.

We see a large difference in the training predictions between the 2 and 3 community systems. In the 2 community system, even in the training period, the predictions tended to wander and be less accurate. Whereas the 3 community training predictions remained very close to the true target node. The movement of the target node in the embedding was very similar between the two systems, and so the difference seems to be that the 2 community has symmetrical inputs. Using symbolic regression to approximate the neural network output seems to increase the stability of the extrapolation. This is most pronounced

6 Conclusion

Not sure what to write for this

References

1. Golub, G. H. & Reinsch, C. in *Linear algebra* 134–151 (Springer, 1971).
2. Hoff, P. D., Raftery, A. E. & Handcock, M. S. Latent space approaches to social network analysis. *Journal of the american Statistical association* **97,** 1090–1098 (2002).
3. Athreya, A. *et al.* Statistical inference on random dot product graphs: a survey. *The Journal of Machine Learning Research* **18,** 8393–8484 (2017).
4. Rackauckas, C. *et al.* Universal Differential Equations for Scientific Machine Learning. *CoRR* **abs/2001.04385.** arXiv: `2001.04385`. `https://arxiv.org/abs/2001.04385` (2020).
5. Kidger, P. On neural differential equations. *arXiv preprint arXiv:2202.02435* (2022).
6. Hogan, E. *Connor Can't Tikz* `https://github.com/roonil-wazlib/connor_cant_tikz`. 2023.