1 Introduction

The modelling of temporal networks is an important task in many real world applications including symptom interactions for mental health, epidemiology, and protein interactions [6, 7, 9, 10, 12]. Temporal networks can be seen as dynamical systems: that is a system in which we have points, in our case nodes in a network, whose states, the edges connecting them, that vary dependent in time. Discovering the underlying equations governing these dynamical systems proves challenging, 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 timestep then appearing at the next. Here, we proposes a hybrid statistical and deep learning framework that allows us to model temporal networks as continous-time dynamical systems, discover a fitting set of differential equations describing it, and, exploiting that discovery, predict the time evolution of a network.

Differential equations are useful for modelling systems where the state of one variable can effect the trajectories of other variables. We observe this behaviour in temporal networks; nodes' connections within the network can influence the formation and decay of edges between other nodes, for example the phenonmenon of preferential attatchment observed in [3, 5]. With this in mind we might wish to draw on the rich mathematical literature of differential equation modelling.

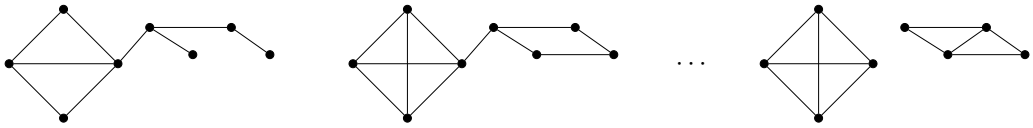
In the common representation of networks as binary-valued adjacency matrices, the events recorded in a temporal sequence of networks correspond to topological events, such as the appearence or the disappearance of link. Because of the discrete nature of events, directly modelling the temporal networks as dynamical systems would require to handle discrete jumps. The topological nature of temporal networks, and the discontinuous character of their temporal evolution, make it challenging to use differential equations techniques.

Here, we overcome the discretness problem by interpreting networks as Random Dot Product Graphs; a well established statistical model for complex networks, that embeds nodes in a low-dimensional metric space[8]. In this way we translate the hard problem of modelling discrete events in the space of networks to the easier problem of modelling continous change in the embedding space. Then, we define and use systems of Neural Network Differential Equations (NNDE) 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 to predict forward in time the evolution of the temporal networks.

In this manuscript, we prove that the temporal network prediction problem can be successfully reinterpreted as a dynamical system modelling problem. In particular, 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.

2 Methodology

We are given a sequence of graphs, and our goal is to predict the next graph in the sequence. To do this we read each graph as a Random Dot Product Graph (RDPG); a visual representation of this can be seen in Figure 1. To obtain the RDPG, we take each network observation Figure 1a and represent them in the form of an adjacency matrix Figure 1b. We then use a truncated singular value decomposition to reduce the dimensionality of our adjacency matrices[ref]. The rows of the output of the SVD can then be viewed as vector points in a d-dimensional space[ref], where d is the number of singular values we take for our SVD. In this space nodes that are similar, ie, nodes have many common neighbours, will be mapped to similar points in the space.



(a) Example sequence of networks.

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

(b) Sequence of sparce adjacency matrices associated with the networks in Figure 1a.

Figure 1: Visual illustration of the proposed framework.

We used a Singular Value Decomposition of each adjacency matrix to embed our network in a low dimensional space. To execute the SVD[1] we use the Julia package [cite]. This package uses implicitly restarted Lanczos iterations[2] to approximate the eigen vectors and singular values of each of our adjacency matrices. From the "svds" function, we obtain two matrices, L and R where:

$$L = \hat{L}\sqrt{\hat{\Sigma}} \tag{1}$$

$$R = \hat{R}\sqrt{\hat{\Sigma}} \tag{2}$$

$$A \approx \hat{L}\hat{\Sigma}\hat{R}' \tag{3}$$

$$\approx LR' \tag{4}$$

$L, \hat{L}, R, \hat{R}$ are each real valued $n \times d$ matrices, where the columns of $\hat{L}$ and $\hat{R}$ are the first d eigenvectors of AA' and $A'A$ respectively ordered by the size of their associated singular values. $\hat{\Sigma}$ is a $d \times d$ diagonal matrix whose entries are the singular values associated with each eigenvector.
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]; this will cause the outputs of the Arpack function to each be aligned with eachother XXX with some requirements of the matrix/initial vec?XXX. Altenatively we can use Procrustes alignmet[cite package] to align the matrices after they have been generated.
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[8], a class of Latent Position Models [4]. 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 [11].

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.

$$\mathbb{P}(i \to j) \coloneqq L_i \cdot R_j \tag{5}$$

$$\mathbb{P}(\hat{i} \to j) \coloneqq L_{\hat{i}} \cdot R_j \tag{6}$$

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

$$\begin{aligned} ||\mathbb{P}(i \to j) - \mathbb{P}(\hat{i} \to j)||_2 &= ||L_i \cdot R_j - L_{\hat{i}} \cdot R_j||_2 \\ &= ||(L_i - L_{\hat{i}}) \cdot R_j||_2 \end{aligned} \tag{7}$$

If we look at the expectation of this, we see

$$\mathbb{E}_j(||(L_i - L_{\hat{i}}) \cdot R_j||_2) = ||L_i - L_{\hat{i}} \cdot \frac{1}{k} \sum_{j \neq i} R_j||_2 \tag{8}$$

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.

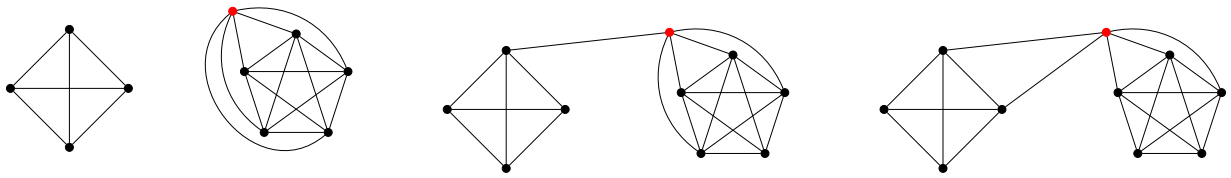
With our trained NODE, we have a black box that approximates the system that governs the movement of Twitter users. To make this more interpretable we then look to approximate this black box into a combination of known functions using symbolic regression[citation (use the one from symreg package)]. For 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[13].

3 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 20 timesteps and compare the predictions for the next 15 timesteps. For clarity, each diagram Figure 2 Figure 3 Figure 4 has been simplified and illustrates the movement of the target node in each sequence.

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 Figure 2a that is represented by the five node community. 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 Figure 2b. This process is then continued Figure 2c for thirty five timesteps to generate our training and test data.

Figures obtained from [14].



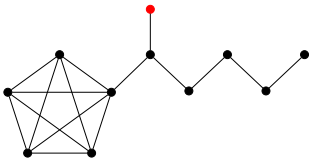
(a) Simplification of the 2 Community network at $t = 1$.

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

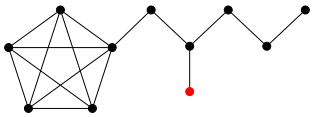
(c) Simplification of the 2 Community network at $t = 3$.

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

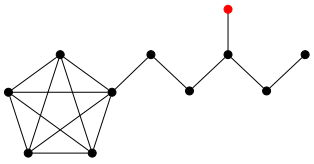
In the long tail network Figure 3, we have a long chain with fifty nodes and a fully connected component with forty nodes at one end of the chain. This is used to orient the embedding and model. The target node starts attatched to the node that is attatched to this large componet Figure 3a. At each time step the target node move one node further down the tail Figure 3b. Again, this is repeated to generate the training and test data Figure 3c.



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



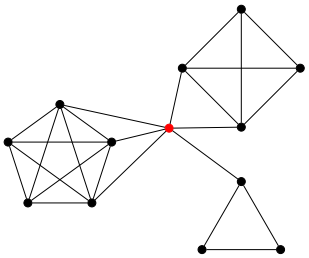
(b) Simplification of the long tail network at $t = 2$.



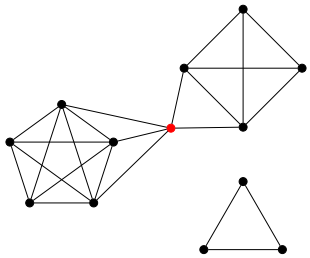
(c) Simplification of the long tail network at $t = 3$.

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

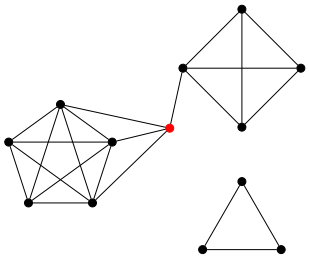
We also simulate a sequence with three communities. The communities have sizes 40, 35, 30 and have 25, 24, and 23 edges to the target node respectively. In Figure 4 the communities have been simplified to have sizes of five, four, and 3. The target node starts connected to all of these communities with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



(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 4: 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 and Discussion

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.
In fig Figure 5, Figure 6,Figure 7 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.

4.1 2 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.

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 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.

4.2 Long Tail

In fig Figure 6 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.

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.

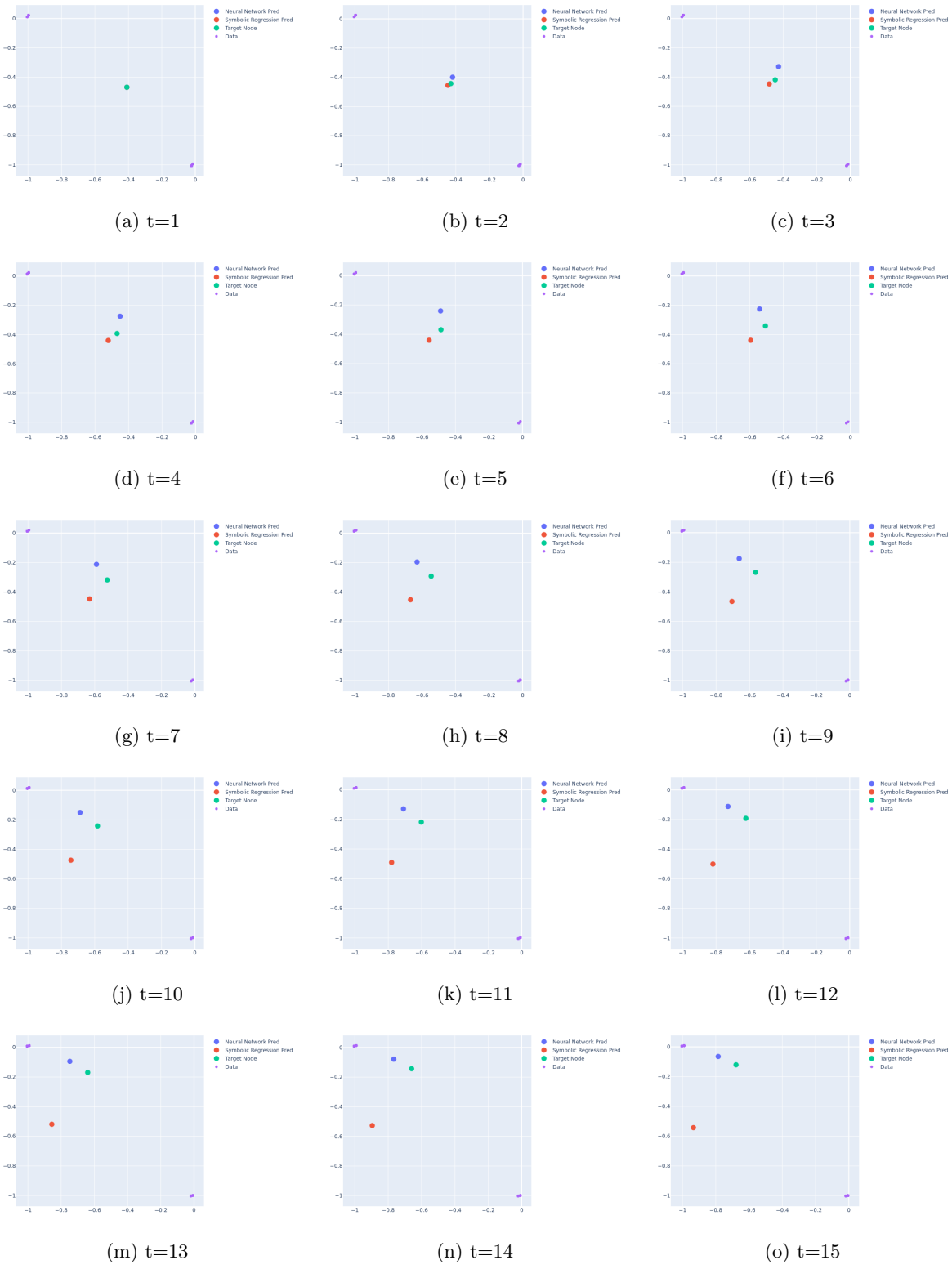
4.3 Three Community (Please Review)

In fig Figure 7 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.

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.



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

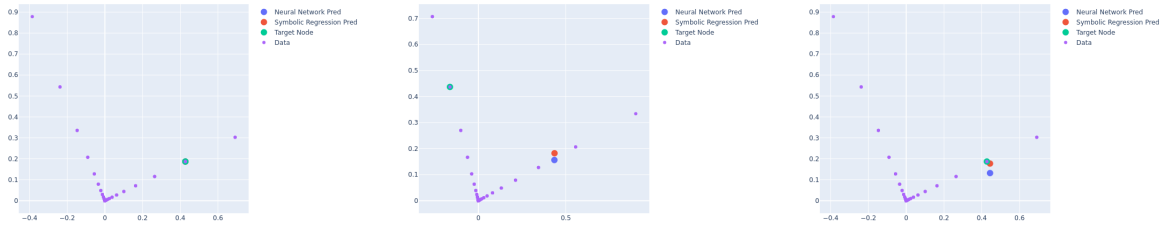
(d) t=4 (e) t=5 (f) t=6

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

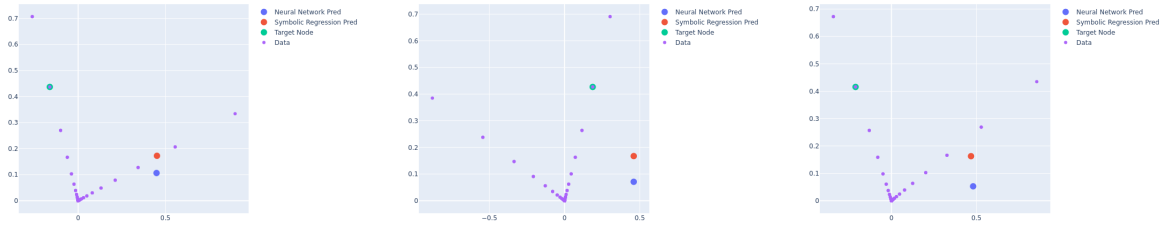
(j) t=10 (k) t=11 (l) t=12

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

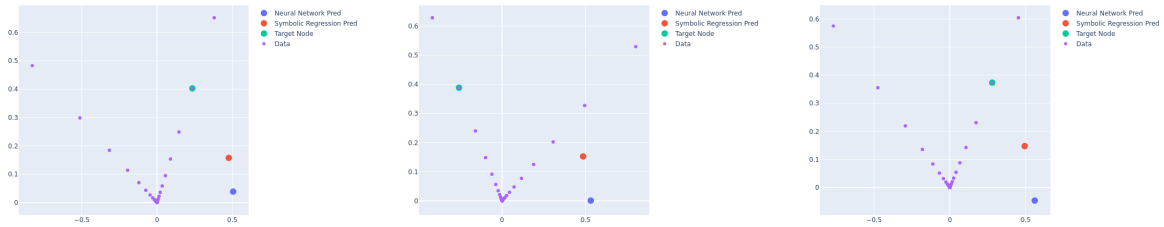
Figure 5: 2 Community test series. This series shows the comparison of the neural network model, the symbolic regression model trained on the neural network model, and the true solution of the two community system.



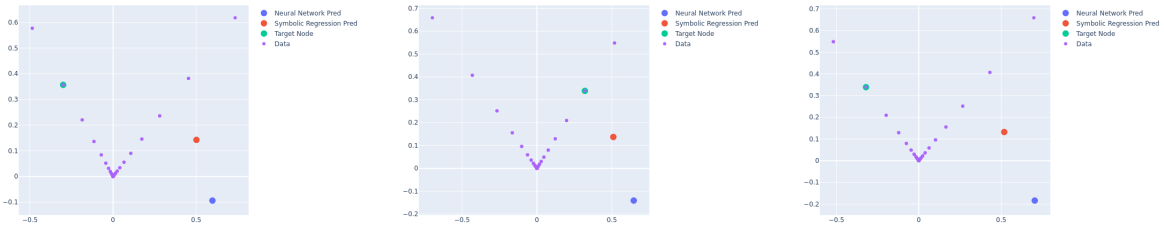
(a) t=1 (b) t=2 (c) t=3



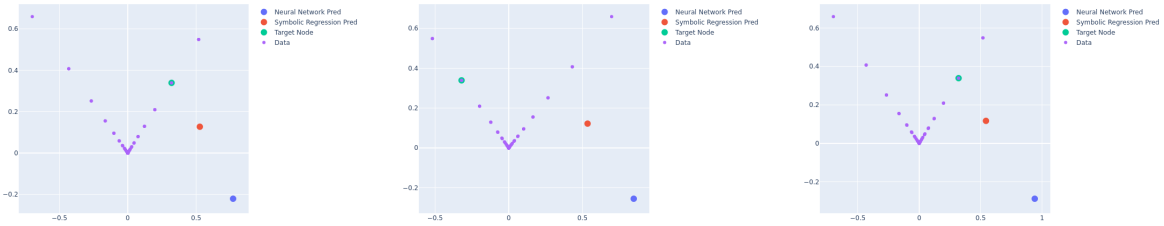
(d) t=4 (e) t=5 (f) t=6



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

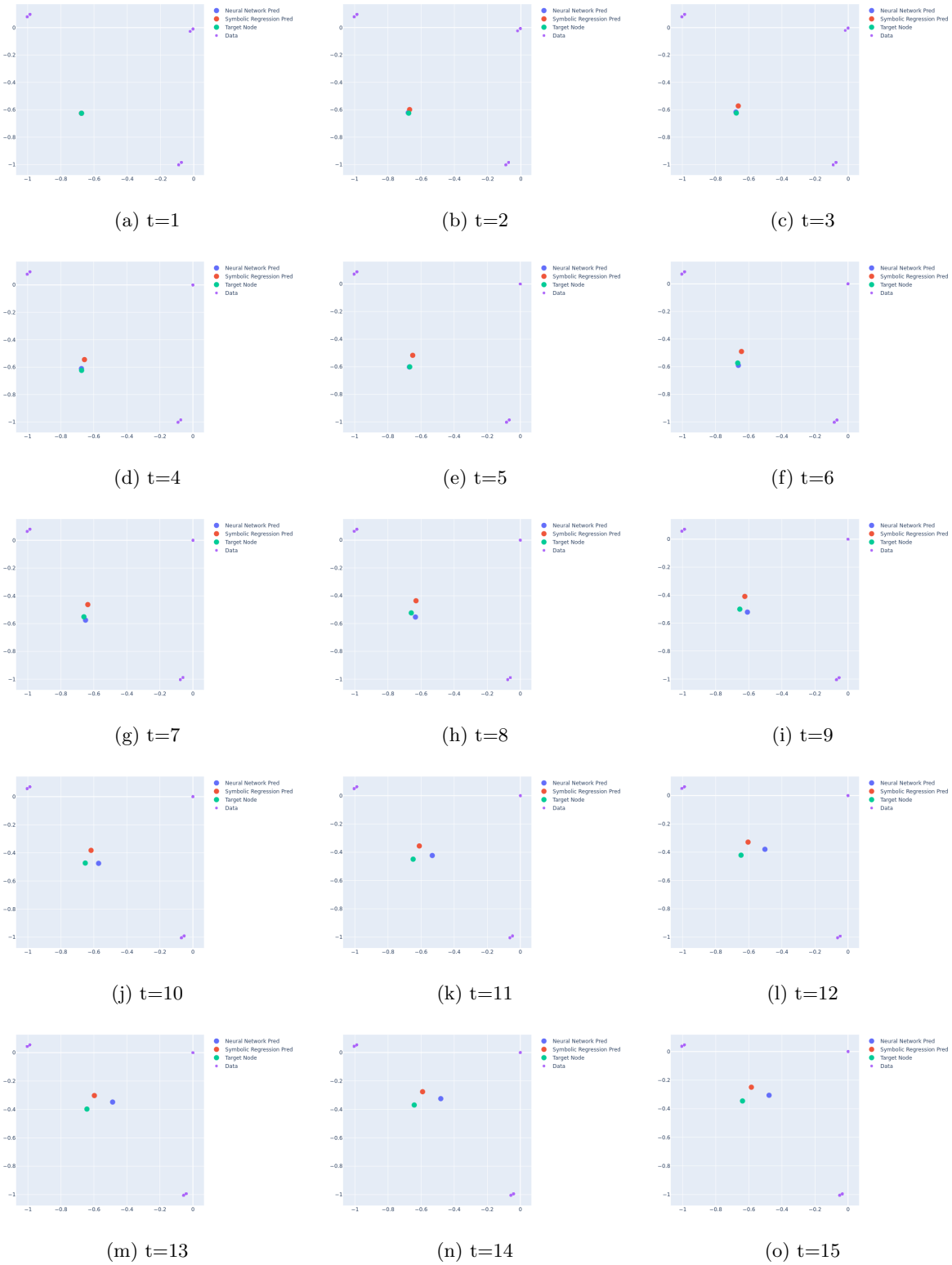


(j) t=10 (k) t=11 (l) t=12



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

Figure 6: 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 7: 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.

5 Conclusion

Not sure what to write for this
Using symbolic regression to approximate the neural network output seems to increase the stability of the extrapolation. This is most pronounced

References

1. Golub, G. H. & Reinsch, C. in *Linear algebra* 134–151 (Springer, 1971).
2. Lehoucq, R. B. & Sorensen, D. C. Deflation techniques for an implicitly restarted Arnoldi iteration. *SIAM Journal on Matrix Analysis and Applications* **17,** 789–821 (1996).
3. Newman, M. E. Clustering and preferential attachment in growing networks. *Physical review E* **64,** 025102 (2001).
4. 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).
5. Capocci, A. *et al.* Preferential attachment in the growth of social networks: The internet encyclopedia Wikipedia. *Physical review E* **74,** 036116 (2006).
6. Jin, R. *et al.* in *Biocomputing 2009* 203–214 (World Scientific, 2009).
7. Masuda, N. & Holme, P. Predicting and controlling infectious disease epidemics using temporal networks. *F1000prime reports* **5** (2013).
8. Athreya, A. *et al.* Statistical inference on random dot product graphs: a survey. *The Journal of Machine Learning Research* **18,** 8393–8484 (2017).
9. Contreras, A., Valiente, C., Heeren, A. & Bentall, R. A temporal network approach to paranoia: A pilot study. *Frontiers in psychology* **11,** 544565 (2020).
10. Jordan, D. G., Winer, E. S. & Salem, T. The current status of temporal network analysis for clinical science: Considerations as the paradigm shifts? *Journal of clinical psychology* **76,** 1591–1612 (2020).
11. 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).
12. Lucas, M. *et al.* Inferring cell cycle phases from a partially temporal network of protein interactions. *Cell Reports Methods* (2021).
13. Kidger, P. On neural differential equations. *arXiv preprint arXiv:2202.02435* (2022).
14. Hogan, E. *Connor Can't Tikz* https://github.com/roonil-wazlib/connor_cant_tikz. 2023.