RNA Velocity and Gene Regulatory Networks

Mathematical Modelling Case Study

Candidate Number: 1061996

1 Introduction

We will start this essay by introducing the biology behind what we are investigating. Once this is clear, we will introduce a model currently used to represent this process and how we plan to improve it.

1.1 Biological Background

In many animals, including humans, stem cells can differentiate into a multitude of different cell types while the DNA inside the cell does not change. The gene expression of a gene is a measurements of how active or well expressed a gene is within a cell. Gene expression profiles can give a picture of how cells are changing, for example if they are actively dividing. Furthermore, the gene expression profile of different cells can be used to predict what type of cell it is [7].

We want to be able to predict the fate of a cell (what cell type a cell is going to differentiate to after a long period of time) given a gene expression profile. Currently, there exists techniques such that we can measure the gene expression profile of a cell at any point in time but these measurements destroy the cell. This means we cannot simply track the gene expression profile of one cell through time, we can only get a static snapshot of it [7].

Vector fields are used in many areas of mathematics and physics, such as astronomy, to model the motion of objects through space. There exists models that give the rate of change of gene expression of a cell given certain data relating to its gene expression. In a similar way, these can be used to generate a vector field, evolve cells the position of cells with time and, ultimately, predict the fate of cells [7].

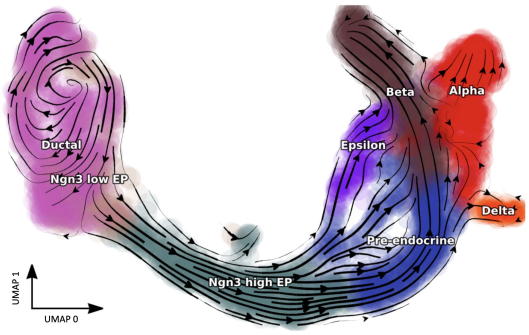


Figure 1: A 2 dimensional vector field describing cell differentiation in the pancreas created using the dynamo package.

The other biology of interest to us is that of gene regulatory networks (GRNs).

These are networks which describe how genes interact with each other. Gene A inhibits gene B if it prevents gene B from expressing itself. Therefore increasing the amount gene A expresses itself decreases the amount gene B expresses itself. If both of these genes inhibit each other we call it cross-inhibition, ultimately with one gene coming out much more expressed than the other after a long period of time. Therefore, the outcome of these cross-inhibitions can be vital in cell fate decisions.

1.2 Modelling

First, we need to introduce the concept of RNA velocity. Suppose that we are considering n genes in a cell at time t. We define the gene expression state for these genes by

$$\boldsymbol{x}(t) \coloneqq (g_1(t), g_2(t), \ldots, g_n(t))^\intercal$$

where $g_i(t)$ is the ith gene's gene expression at time t. We can then define the RNA velocity as the rate of change of the gene expression state with time. More formally,

$$\dot{\boldsymbol{x}}(t) \coloneqq (\dot{g_1}(t), \dot{g_2}(t), \ldots, \dot{g_n}(t))^\intercal.$$

As mentioned in the previous subsection, vector fields can be a useful tool in predicting the cell fate of a cell with known gene expression. The problem we have now is how to construct a vector field f over gene expression-space for our selected genes such that

$$\boldsymbol{f}(\mathbf{x}) \coloneqq (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_n(\mathbf{x}))^\intercal \text{ with } \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\mathbf{x}(t))$$

for all $t \geq 0$.

Plenty of models have been developed to approximate RNA velocity with dynamo using the model described in the paper Mapping Transcriptomic Vector Fields of Single Cells [7]. We will not go into the model here as the details are not too important. All we need to know is that given certain gene expression and splicing data for a cell the RNA velocity of the cell can be calculated. Dynamo is a package that then uses this model to generate a vector field in 2 dimensions such that the time-evolution of cells can be visualised and we can predict their fate. The path of cells through time is animated using the in-built animation function, which evolves the projected gene expression vectors in 2 dimensions using a 2 dimensional vector field.

In figure 2 we see two screenshots taken from the output of the animation function of dynamo. First, we notice the different coloured regions representing different cell types. The cells start in the ductal cells region (which are stem cells in the pancreas)

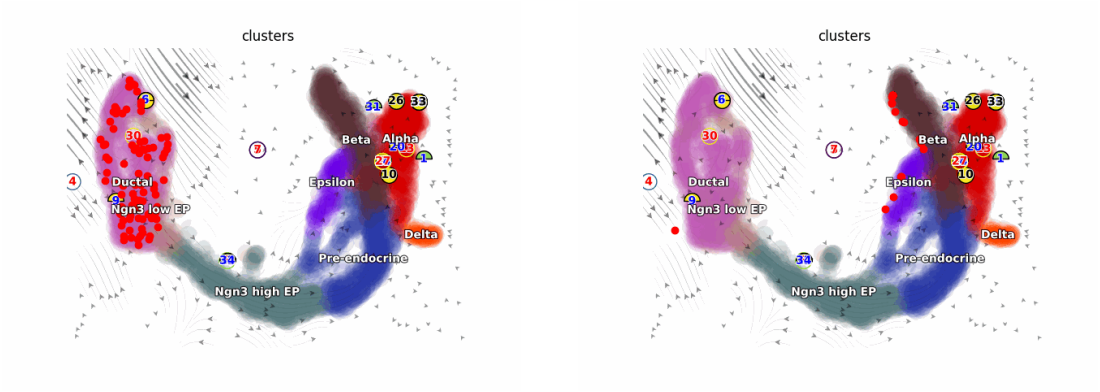


Figure 2: These are two screenshots of the animation given by dynamo for the pancreatic endocrinogenesis dataset. The left screenshot gives the initial states and the right screenshot is at a later time.

and, as time evolves, they develop into other cells. This shows how dynamo predicts how cells will differentiate. Second, it is important to note that some cells disappear straight off the edge of the plot. This is due to an error in the program or model as it makes no sense physically what is happening here. Lastly, we have problems with the 2 dimensional projection. In figure 1 we see that cells can change from epsilon to beta cells, but this is physically impossible [8, 9]. We also have the inherent problem that if we project the gene expression down to 2 dimensions certain places in gene expression-space may be projected on top of each other. This looks to be what has happened in figure 1 with beta being projected over a path between epsilon and alpha. This can make our 2 dimensional vector field and therefore cell predictions inaccurate, especially since the gene expression is only being evolved in 2 dimensions in dynamo.

1.3 Aims

The main goal we had was to train a neural network to work as the vector field rather than using the model described in the previous subsection. As an input we would have a vector with each entry being the gene expression of a gene and as an output we would have the RNA velocity for the corresponding genes. The current accuracy of dynamo was not great so we thought that using a neural network might mitigate some of the problems of the model and give us more accurate results.

Another goal was training a neural network with no hidden layers, biases or activation functions so that we could use the weights to model a GRN. We hoped this would work as the neural network would simply be a linear map between gene expression-

space and the RNA velocity-space. Therefore the greater the magnitude of a weight linking two genes, the greater the effect changing the input gene expression would have on the output gene's RNA velocity. The advantage of doing this is that it allows us to check specific cross-inhibitions which we know should be present and therefore provides an insight into whether our neural network is predicting the RNA velocity accurately. It would also enable us to predict other genes that are cross-inhibiting.

We also wanted to use a larger neural network with an activation function to create a GRN as we predicted the larger network would be more accurate. This will however involve different techniques which we will discuss later. We wanted to be able to use our new model to get more accurate cell fate decisions and we wanted to be able to evolve our cells position in gene expression-space in higher dimensions to mitigate the effects of crossing trajectories when projecting to 2 dimensions. This would also allow more information to be present when evolving the position in gene expression-space and, if we can choose the dimension we evolve it in, we can include specific genes which we know play an important role in the cell fate decision. We could also choose to include more genes if the predictions we are getting are poor which should improve the accuracy of predictions.

Finally, we wanted to compare the results of our plots and GRNs to dynamo's ones and to have made improvements on the work they did.

2 Dynamo

To begin with each of the group forked the dynamo repository from GitHub so that we would all have a copy of the package to work with on our laptops. We then needed to get the dynamo package working so we could understand exactly what it does, how it does it and what plots it can generate. This should have been a simple step but in fact took us around four weeks to get the package running without fault. Because the package was still under active development and did not have a large team behind it much of the documentation was outdated. Therefore we found that we got errors or inaccurate plots when using the dependencies stated in the documentation. As a result we spent the first few weeks changing dependencies to different versions until we eventually got everything running as it should be. Ultimately, we only got the package working on two of our laptops but we decided that would be enough for us to develop our code and therefore decided not to spend any more time testing different combinations of dependencies.

Next we needed to choose a dataset to work with. The pancreatic endocrinogenesis

dataset [1] was ideal for us: it was working with dynamo however there was clear error in the way the points were evolved as shown in figure 1. We saw that ε cells became β cells but physically we know this is not possible as it defies a known cross-inhibition [6]. We hypothesised that this was a result of dynamo projecting down to 2 dimensions before evolving the points and that we may be able to prevent this happening with our neural network approach. Points also disappeared off the edge of the animation shown in figure 2 which makes no sense physically. Furthermore, since we know a cross-inhibition should be occurring and where it should be occurring in gene expression-space we can test if our neural network and GRN are working correctly on this dataset.

Ultimately, the cells from the pancreatic endocrinogenesis dataset have four major cell fates: α cells, β cells, δ cells and ε cells. We know that the genes PAX4 and ARX cross-inhibit each other, particularly as the cell is developing [6]. In addition, we know that α and ε cells have high ARX gene expression and β and δ cells have high PAX4 gene expression [8, 9]. We therefore expect this cross-inhibition to govern part of the cell fate decision and we should be able to see where it occurs on our plots and test whether or not we find this cross-inhibition with our GRN.

Last in this section we could discuss the workflow of dynamo, however in the end, we did not use dynamo very much and we are lacking space in the essay so we will not go into it here. We used the preprocessing function from it to get our data in the correct form for us to use and we used it to calculate the RNA velocity of the cells in our dataset which we need to train our neural network later, but that is all. Now, we will investigate how to develop this neural network.

3 The Neural Network

The first thing to consider here is how can a neural network be used in this situation. Dynamo evolves the positions of these vectors in the 2 dimensional projected space by using the model considered at the start of the essay and a forward Euler scheme. However, as stated in our aims, we want to evolve the positions of our cells in higher dimensions so as to avoid trajectories crossing and also to allow more information to be used when we decided how the cells will evolve. We hope this will give us a more accurate cell fate decision. We therefore use a neural network with say an n dimensional input and an n dimensional output with the input corresponding to the gene expression of n genes and the output corresponding to the RNA velocity of the same genes. We can then evolve the position in gene expression-space using a forward

Euler scheme then use UMAP [3] to project down to a lower dimensional space to visualise the evolution, but we will discuss this last part more next section.

Now the problem becomes how to train the neural network. Each input required for the network is an n dimensional subset of the genes in gene expression-space. We can find the RNA velocity for the respective gene expression by using dynamo (which uses the model from [7]) and use this to train the network. Therefore we can assign each cell in the pancreatic endocrinogenesis dataset to be either training or test data and use the training data to train the neural network. The only other thing we will need before we can start training is a loss function. We will use the mean squared error to compare the predicted RNA velocity from the neural network and the actual RNA velocity (the one given by the model in section 1.2). We then give the accuracy on one vector by

$$a(\mathbf{v}_{true}, \mathbf{v}_{pred}) = 1 - \frac{||\mathbf{v}_{pred} - \mathbf{v}_{true}||_2^2}{||\mathbf{v}_{true}||_2^2} \tag{1}$$

where $\mathbf{v}_{pred}$ is the RNA velocity predicted by our neural network and $\mathbf{v}_{true}$ is the RNA velocity given by the model in [7]. We then calculate the mean of this over all our test data for the accuracy.

The first neural network we used (model 1) was a basic one given to us by our supervisor. It had no hidden layers, biases or activation functions, it was just a linear map between an 18 dimensional gene expression-space and an 18 dimensional RNA velocity-space. Training a neural network with the gene expression of every gene we have data for as an input and every gene we have data for as an output would be very time consuming and not feasible with the amount of data we have. Furthermore, most of the genes are dormant so their expression does not change or changes very little. As such, we only use 18 highly variable genes to describe the motion of the cell. This is a modelling assumption we must make to keep the problem feasible. Our supervisor chose to use an 18 dimensional space as these are the 18 genes given by cross-referencing a list of known transcription factors (genes which we know enhance or decrease the expressions of other genes) with the list genes we get by finding the top PCA factors of our dataset (a list of highly variable genes). The reason for this network architecture was to achieve one of the aims of our project: to model the GRN using the weights of this network. We hypothesised that this might work as the greater a weight joining node i to node j, the more we would assume that gene i would activate gene j, and similarly for larger negative weights but with inhibition.

We trained the network using the Adam optimisation algorithm then tested it to find that it had an accuracy of 0.78 which seems quite promising given that this is

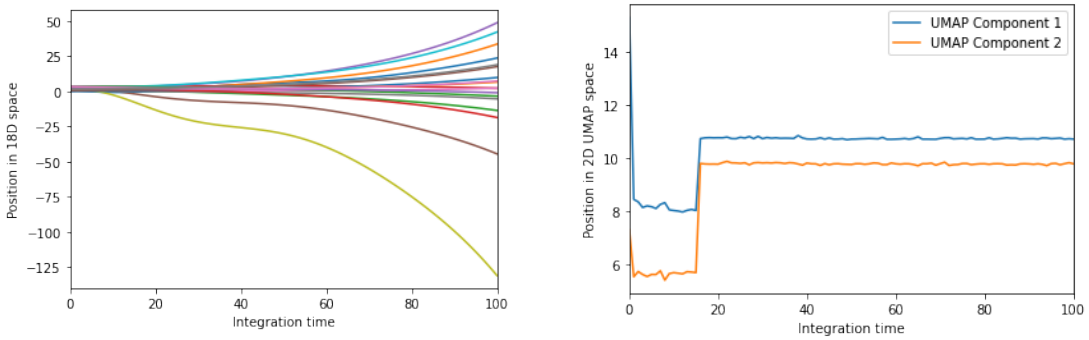


Figure 3: The gene expression of a point in 18 dimensional gene expression-space and 2 dimensional UMAP-space as it is evolved using model 1. The gene expression of each of our 18 genes is represented by a line in the figure on the left.

only a linear map. However, this may be less promising than it seems on first sight as our accuracy can be quite large even when the relative error in the result is quite large. For example, consider if we have $v_{true} = 0.6$ and $v_{pred} = 0.3$, then equation (1) would give us $a = 1 - 0.3^2/0.6^2 = 0.75$. Therefore there is still a lot of room for improvement with the neural network so clearly a linear model is too basic. This is further evidenced by figure 3 as despite the position in 18 dimensional gene-space changing we have no meaningful evolution in the 2 dimensional UMAP-space. Hence, we need to add activation functions and layers to our network. This was expected as the relationship between gene expression and RNA velocity is rarely linear, more so it often takes more than one gene to inhibit or activate another gene so just accounting for the relationship between two genes given by the weight connecting them may not be very telling. In the end, we decided not to use the weights as a GRN as the prediction of the network was not very good so we did not think it would tell us anything. In addition, we would be calculating the same GRN for every point in the 18 dimensional gene expression-space which is not realistic as the amount genes inhibit each other depends on how well expressed each of them are.

We know by the universal approximation theorem that a network with a large enough hidden layer is able to approximate any continuous function to arbitrary precision. Our supervisor therefore suggested using a network with a single hidden layer of 1800 neurons, biases and sigmoid activation functions (model 2). After training with Adam we made a considerable improvement to the accuracy: we got it to 0.82. The problem we had is that we were not sure if we could trust this value and how much the predictions had actually improved. Thus, we decided to plot the points from our dataset in 18 dimensions projected down to 2 dimensions using PCA along

with the velocities at some of these points, again projected to 2 dimensions using PCA.

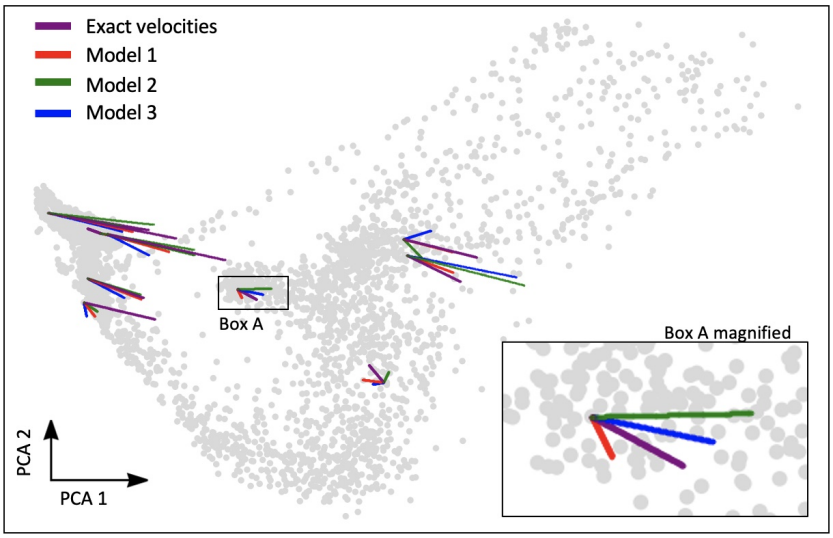


Figure 4: The grey dots in this plot are the gene expressions of all the cells in the dataset projected to 2 dimensions from our 18 dimensional space using PCA. We plot the velocities (projected to 2 dimensions using PCA) calculated at some of these points by each of the different neural networks and the exact velocities at these points calculated using dynamo [7].

From figure 4 we see that as the accuracy increases the neural network's prediction of the velocity gets better. This is shown by the more accurate model's velocities tending to better approximate the exact velocities. In fact, we calculated the average angle between the exact velocity and each model's velocity (in 18 dimensions) over all points in our dataset. Hence we can see our network is becoming more accurate as the average angles for models 1 and 2 are 0.470 and 0.377 radians, respectively.

We did not however stop with this neural network. We added another layer such that we had a network with the ReLU activation function and two hidden layers, the first of 1800 neurons and the second of 900 neurons (model 3). This was trained with Adam and a dropout of 0.4 between the hidden layers and gave an accuracy of 0.86. Furthermore, we can see an actual improvement in the predicted velocity looking at figure 4 and noting the average angle drops to 0.327 radians.

4 Implementation

We now have a working neural network that given any point in our 18 dimensional gene expression-space will find us the corresponding RNA velocity in the 18 dimensional RNA velocity-space. As we briefly mentioned at the start of the previous section we can then evolve the points in 18 dimensional space using a forward Euler scheme. More specifically, suppose we have a point in reduced gene expression-space at time t_k given by $\mathbf{x}(t_k) \in \mathbb{R}^{18}$ and the RNA velocity predicted by our neural network at this point is given by $\mathbf{v}_{pred}(t_k) \in \mathbb{R}^{18}$. Then we can predict $\mathbf{x}(t_{k+1})$ with $t_{k+1} = t_k + \Delta t$ by $\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \mathbf{v}_{pred}(t_k)\Delta t$.

One of the advantages of using our method rather than the one presented in dynamo is that we evolve the points in higher dimension. Not only does this mean that we should not have trajectories crossing, but also we will allow more information to used when evolving the point therefore hopefully giving us more accurate trajectories.

We can project down from the gene expression of the cells in our dataset to the 18 dimensional gene expression-space by selecting the 18 genes highly variable genes found in the last section. We can then evolve the points in this 18 dimensional space using a Forward Euler scheme. The problem now is being able to make use of these 18 dimensional gene expression vectors: we want to be able to visualise them so we can predict the cell fate decisions.

In the dynamo package they used PCA to map to around 30 dimensions followed by a UMAP map to 2 dimensions. Since we are only at 18 dimensions we will go straight for using a UMAP map. The way UMAP works technically is not too important here, for our purposes all we need to know is that it is a dimensionality reduction technique and packages to implement it exist that we can use. One important thing to note however is that in reducing dimensions we will almost always lose information on the points we have. What we mean by this is that UMAP is not bijective, different points will get mapped to the same point and points which are near each other in the higher dimensional data may not be near to each other in projected space. This can be seen by looking at figure 5 and noting that some regions which are connected in the 3 dimensional plot may no longer be connected in the 2 dimensional plot: it is almost as though the mammoth has been run over by a car. Nonetheless, in general the intrinsic dimension of the data is less that 18 and therefore UMAP will still give us a useful visual representation of the data.

To apply UMAP to our data we first had to train the mapper on the 18 dimensional data we got by reducing the dataset. Once this mapper is trained a function has been

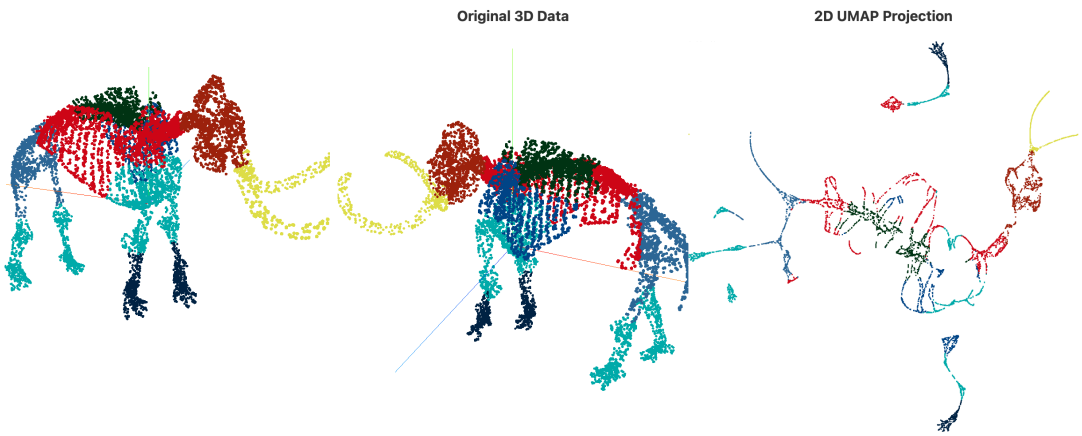


Figure 5: On the left hand side and centre we have a 3 dimensional plot of points which looks similar to an mammoth shown from two different perspectives. On the rights hand side we have a 2 dimensional projection of this using UMAP. We got these plots from [2]

learnt that maps from 18 dimensions to a lower dimension such that we can map any point in 18 dimensions to a point in UMAP-space. The reason UMAP has to be "trained" is that we want a specific function that represents our data in lower dimensions as well as possible, that is the lower dimensional space should retain meaningful properties of the higher dimensional space. For example, points that are close in the higher dimensional space should be close in lower dimensional space.

Another advantage of evolving our points in higher dimensional space then projecting to a lower dimensional space after to visualise it is that we can pick the dimensions of the space we are reducing down to as a parameter of the UMAP function. This means we can project to 3 dimensions rather than 2. Therefore we have achieved another one of our aims: we can plot the trajectory of a cell in 3 dimensions.

If someone then wanted to find the evolution of a specific cell and they had the gene expression profile of it they could project to 18 dimensions by selecting the gene expressions of the selected genes and use our model to evolve its position. To see what we achieved let us look at some plots.

Figure 6 shows the evolution of a point from the dataset with time in 3 dimensions. Being able to create these plots in 3 dimensions is useful as it allows us to visualise what is happening, but since the points are evolved in 18 dimensions now anyway the problem of crossing trajectories has already been solved so it does not actually provide any mathematical benefit to us.

The last thing we want to check is if we can achieve our main goal, predicting the

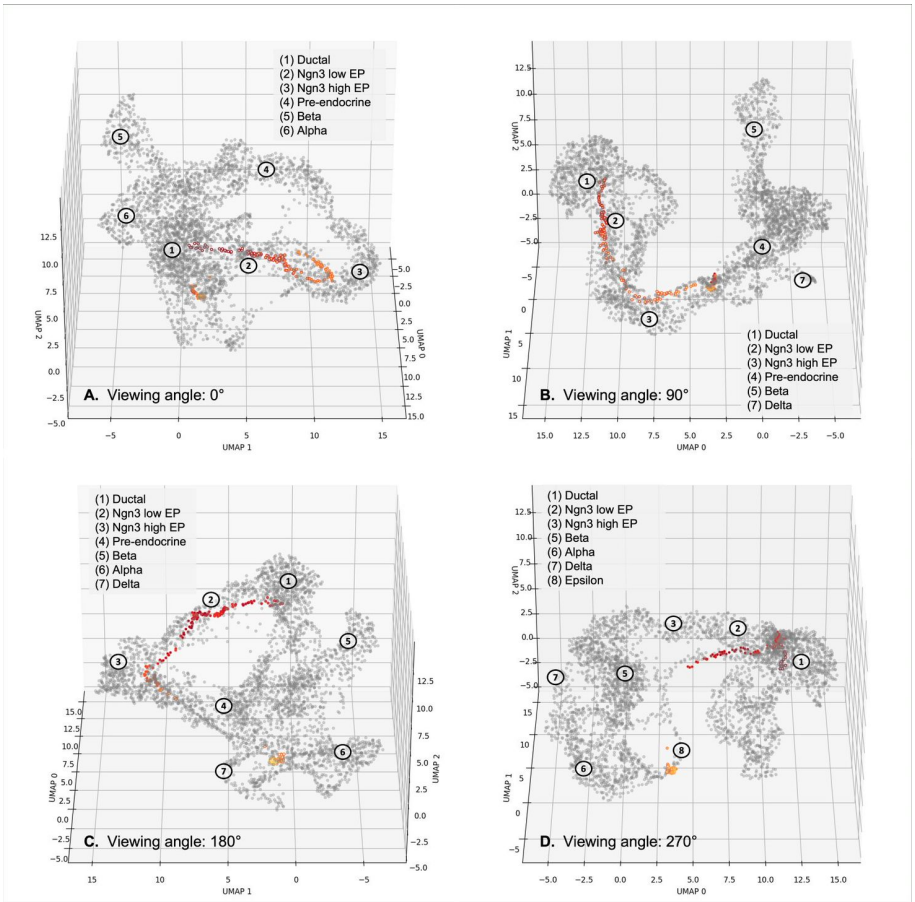


Figure 6: The 3 dimensional plot of a trajectory given by model 3. The point starts at the darker red colours and evolves to the lighter orange colours.

fates of cells using our model. It turns out that with our model we can predict the cell fates but all the cells from our initial states become ε cells. This is an improvement from dynamo's cell fate predictions as all of the cells become beta cells or completely disappear off the plot when it is used on the same dataset. However it is not perfect as we wanted to see cells differentiating to all the different cell types.

Let us look at which of the aims from the start we have achieved in the last two sections. We started by developing a neural network to predict the RNA velocity of certain genes given their gene expression profile. We used a basic two layer neural network with no biases or activation function first however the network was not very accurate and we decided that the idea of using the weights as a GRN was flawed. We then developed a more complex neural network increasing the accuracy from 0.78 to 0.86 along the way. To allow us to evolve points using our neural network we implemented a forward Euler scheme to use with the input and output of the neural

network. We then trained a UMAP function to project our data down to 2 or 3 dimensions so we can visualise it and test our code using the initial states from our dataset. Ultimately, we did not get cells differentiating to all the different cell types when using our initial states but we think our prediction was slightly better than dynamo's. The only aims we have left now are to create a GRN and compare it to dynamo's.

5 Gene Regulatory Networks and Communities

5.1 Cross-Inhibition and Gene Regulatory Networks

First, let us recap what inhibition and activation mean. Gene A inhibits gene B if it prevents gene B from expressing itself. Therefore increasing the amount gene A expresses itself decreases the amount gene B expresses itself. If both of these genes inhibit each other we call it cross-inhibition. On the other hand, gene A activates gene B if it causes gene B to become more expressed. If gene A and B activate each other we call it mutual activation. Another important definition for this section is that a progenitor cell is a descendent of a stem cell that differentiates further to form a specialised cell.

As we mentioned in section 2, α and ε cells have high ARX gene expression and β and δ cells have high PAX4 gene expression. In addition, PAX4 and ARX cross-inhibit each other. This cross-inhibition is particularly prevalent in the progenitor stage of the cell just pre-differentiation as cells are more likely to be expressing both genes a little rather than one a lot more than the other [6]. Creating a GRN that has this cross-inhibition would be great as it would not only provide evidence that our network is working correctly, but also it would mean we could predict further inhibitions and activations between genes. This could then be checked or tested scientifically to see if the relationships exist.

We estimate the cross-inhibition and mutual activation at position $\mathbf{x}$ in gene expression-space using the formula

$$I(\mathbf{x}) \coloneqq \frac{\partial f_{ARX}}{\partial x_{PAX4}}(\boldsymbol{x}) + \frac{\partial f_{PAX4}}{\partial x_{ARX}}(\boldsymbol{x}) \tag{2}$$

where f_{ARX} and f_{PAX4} give the RNA velocity of ARX and PAX4, respectively, and x_{ARX} and x_{PAX4} give the gene expression of ARX and PAX4, respectively. We can therefore estimate the cross-inhibition of all pairs of genes at $\mathbf{x}$ by calculating

$J(\mathbf{x}) + J^\intercal(\mathbf{x})$ and setting the diagonal entries to zero where J is the Jacobian of $\mathbf{f}$ with respect to $\mathbf{x}$.

We can find the point in gene expression-space where the most cross-inhibition of ARX and PAX4 occurs by minimising I over all $\mathbf{x}$ where cells can exist in gene expression-space. Dynamo gives a function to calculate this Jacobian for any $\mathbf{x}$ allowing us to calculate $I(\mathbf{x})$ using dynamo. We can also calculate the Jacobian for any $\mathbf{x}$ using our model by differentiating the neural network function with respect to the gene expression of each of the genes by using autodifferentiation functions in the TensorFlow package. Therefore we can find a GRN at any given $\mathbf{x}$ for each of these models and compare them.

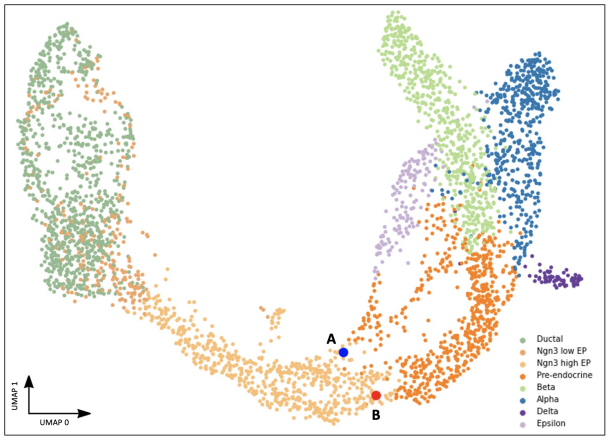
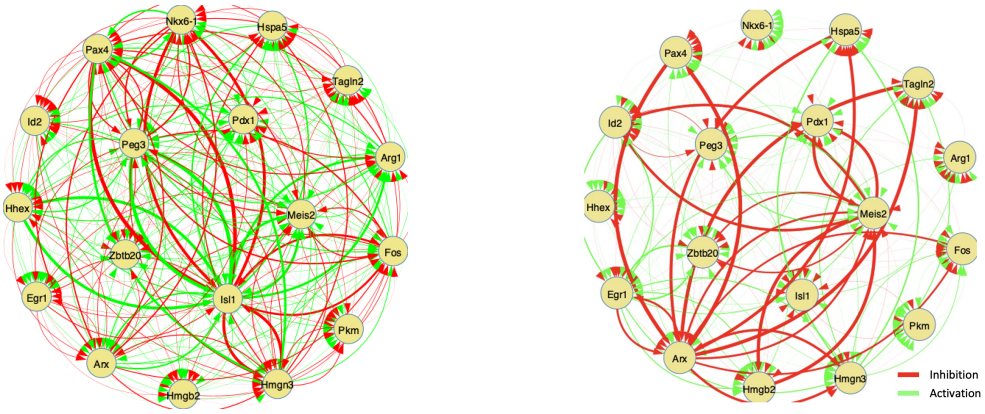


Figure 7: The dataset projected to 18 dimensions by selecting the corresponding genes then projected to 2 dimensions using UMAP. Points A and B are where the estimations of the cross-inhibition of ARX and PAX4 are maximised over cells in the NGN3 high EP region with the estimations calculated using our neural network and dynamo, respectively.

Minimising I for both models over the gene expressions of all NGN3 high EP cells in our dataset gives the points in figure 7. Both of these points are on the boundary to the next cell type as we would expect because in general cross-inhibition would be greater as we get closer to differentiation. Figure 8 shows a GRN created with data from dynamo and one created with data from our neural network for point A and B from figure 7. In these diagrams red arrows represent cross-inhibition and green arrows represent mutual activation and the thickness of the arrows represents the magnitude of the inhibition or activation. Since we add a matrix and its transpose together we will get a symmetric matrix so both arrows between each pair genes will represent the same thing and be of the same thickness.



(a) This GRN was created using the Jacobian calculated by the dynamo package for point B.

(b) This GRN was created using the Jacobian calculated by differentiating our neural network for point A.

Figure 8: These are two GRNs. The plots were made using the iGraph package [4].

We see the GRN created using our model predicts the PAX4-ARX cross-inhibition much more clearly than the one created using dynamo. We could also use the network to predict other cross-inhibiting pairs of genes at this point $\mathbf{x}$ by finding other thick pairs of red lines. For example, HSPA5 and ARX in figure 8. Biologists can then check whether these cross-inhibitions actually occur.

5.2 Communities

Given that we can now estimate the GRN for any point $\mathbf{x}$ in gene expression-space we can use this network to find communities of genes which interact with each other the most. That is, we can find collections of genes which are more frequently linked to each other and with greater strength links than they are linked to genes not in this collection. Many genes are inhibited by a combination of genes or collections of genes all inhibit each other so the cell fate can depend on the amount each of these genes is being expressed. If we only consider the edges in the network representing inhibition finding communities could be helpful in finding these collections of genes.

Community detection in networks has been an active area of research for many years and it often has to be used on networks much large than ours hence there have been numerous methods developed. An assignment of nodes (genes in our case) into groups (or communities) is called a partition of a network. One of the most popular community detection methods is maximising an objective function, such as the modularity, over all possible partitions of the network. Here, the objective function

is meant to represent the quality of a partition. We will not define a specific function here as we are lacking space but the general idea of a modularity function is this. We can create random networks that are not meant to exhibit any community structure. If our network therefore has a greater density and weight of edges in a subgraph than we would expect there to be in an equivalent random subgraph we would expect there to be some community structure. The modularity function attempts to quantify this difference and allows us to find, in some senses, the best possible partition by maximising it [5].

The problem now becomes how to find the optimal partition given that there are far too many choices of partitions to check each of them individually. Greedy algorithms are often used however it would even be too computationally expensive to use them here. Thankfully many other methods have been developed and we find that the Leiden algorithm works on our network [10]. The result of applying this to our network from figure 8b with the green edges removed can be seen in figure 9 where the colours of the nodes represent communities. We would therefore expect the cell's fate to be influenced by which of these genes becomes most expressed after this cross-inhibition.

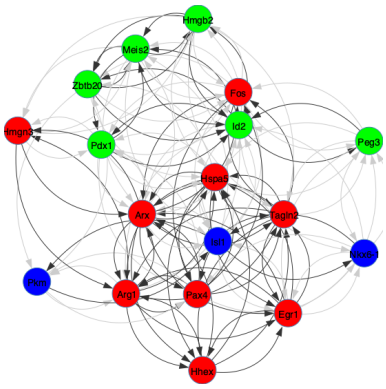


Figure 9: Partition of our GRN from figure 8b with only negative weighted edges considered. This was found using Leiden's algorithm. The plots were then made using the iGraph package [4].

6 Extension

As is often the case with neural networks it can be difficult to tell *a priori* which architecture will work the best for the problem at hand and will still be practical to train on our laptops. We wanted to run some extra numerical simulations to ensure

our neural network could not be improved by adding more layers. As such, we added another layer of 900 neurons (model 4) and trained it using Adam and a dropout of 0.4 between each of the hidden layers. We can see how the accuracy changes throughout training in figure 10. This shows that model 4 makes no significant improvement on model 3 as the lines are very similar throughout training. We therefore assume that the adding any more layers would not increase the accuracy of our neural network any more.

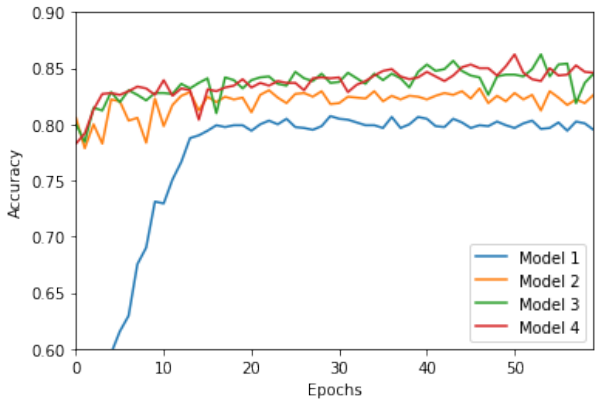


Figure 10: The accuracy against the number of epochs for each of our neural networks.

Next, we investigated a new dataset given to us by our supervisor containing data about the gene expression of cells in the thymus. It contained gene expression data for 1372 cells, however the data was not in the required form for our workflow. We therefore had to convert it to the correct form by writing it to a different data structure. After, we were able to follow the normal workflow and train our neural network. Unfortunately, the accuracy of the neural network never got above 0.5299 when we were training it hence we assume we did not have enough data to train the neural network.

7 Conclusion

Let us look back at our initial aims and check whether or not we have achieved them. The main goal of the project was to train a neural network to model a vector field such that we could make more accurate cell fate decisions. In order to do this we developed a neural network which takes a position in a chosen reduced gene expression-space as an input and predicts the corresponding RNA velocity of these genes. We started with a simple two layer neural network with no biases or activation

functions so that we could use the weights to model a GRN. Ultimately, we found the predictions from the network to be poor and the idea of one GRN to represent all possible gene expressions to be flawed hence we did not plot it. We then made this network more complex adding layers, biases and activation functions in order to increase the accuracy. Finally, we implemented a forward Euler scheme so that we could plot the trajectory of points through time. Here, we found that we could predict the fate of cells but for all our initial states the cells became ε cells. We believed the cell fate decisions would be more accurate as we could now evolve the points in a higher dimension than dynamo therefore allowing trajectories to cross and more information to be present when deciding the direction and size of each evolution. It could be argued that we improved dynamo as all of their cells either become β cells or disappear off the plot. Nonetheless, we did not find cells differentiating to all the cell types as we had hoped.

The second most important aim was to be able to use our neural network to create GRNs for our data and compare them to dynamo's to see which better predicts the known cross-inhibitions. We did this by differentiating the neural network to find the relevant Jacobian and using this to estimate the cross-inhibition of all relevant genes. We then minimised the entry relating to the cross-inhibition of PAX4 and ARX over the set of gene expressions of the NGN3 high EP cells from our dataset and saw that the GRN created by dynamo when the cross-inhibition was minimised over the same set had some cross-inhibition between the pair, but not as much as our model. We think this is good evidence for our model being more accurate than dynamo. Finally, we found communities of genes that interacted with each other via cross-inhibition as we hoped these communities would give collections or combinations of genes that cross-inhibit each other.

This lays the groundwork for lots of possible further investigation. We could try developing our neural network for a different number of input and output genes and see how this affect our predictions. More so, we could try adding and removing certain transcription factors to see how they affect the differentiation of cells. We could apply this framework to different datasets and see if we can get accurate results from the cell fate prediction and GRN. We could try to predict other genes that cross-inhibit using our neural network and test if they do scientifically, which would also be a great way to test if our GRN is working well. Ultimately, we believe that we have achieved most of the aims of our topic, but that there are still plenty of avenues left to explore with this new model.

References

[1] Bastidas-Ponce A, Tritschler S, Scheibner K Dony L, Tarquis-Medina M, Salinno C, Schirge S, Burtscher I, Böttcher A, Theis FJ, Lickert H, and Bakhti M. Comprehensive single cell mrna profiling reveals a detailed roadmap for pancreatic endocrinogenesis. *The Company of Biologists Ltd*, pages 1–14, June 2019. URL `\url{https://cob.silverchair-cdn.com/cob/content_public/journal/dev/146/12/10.1242_dev.173849/9/dev173849.pdf?Expires=1654558912&Signature=J94JQmf0hh~lBJhPUW8VCNt50l-Y5vQ5gWuUv-A8hGWz7n47x84ucCrgW4E0nB6N8I-L~Wtp4im9w_&Key-Pair-Id=APKAIE5G5CRDK6RD3PGA}`.

[2] Andy Coenen and Adam Pearce. Understanding umap. `https://pair-code.github.io/understanding-umap/`. Accessed on 1st May 2022.

[3] Andy Coenen and Adam Pearce. Umap: Uniform manifold approximation and projection for dimension reduction. 3:1–57, September 2020.

[4] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. URL `https://igraph.org`.

[5] Santo Fortunato. Community detection in graphs. *Physics Reports 486, 75-174 (2010)*, 2:1–16, 27–41, Jan 2010. URL `\url{https://arxiv.org/abs/0906.0612v2}`.

[6] Collombat P, Mansouri A, Hecksher-Sorensen J, Serup P, Krull J, Gradwohl G, and Gruss P. Opposing actions of arx and pax4 in endocrine pancreas development. *Genes and Development*, pages 1–12, Oct 2003. URL `\url{https://www.ncbi.nlm.nih.gov/pmc/articles/PMC218152/}`.

[7] Xiaojie Qiu, Yan Zhang, Jorge D. Martin-Rufino, Chen Weng, Shayan Hosseinzadeh, Dian Yang, Angela N. Pogson, Marco Y. Hein, Kyung Hoi (Joseph) Min, Li Wang, Emanuelle I. Grody, Matthew J. Shurtleff, Ruoshi Yuan, Song Xu, Yian Ma, Joseph M. Replogle, Eric S. Lander, Spyros Darmanis, Ivet Bahar, Vijay G. Sankaran, Jianhua Xing, and Jonathan S. Weissman. Mapping transcriptomic vector fields of single cells. 4:1–60, Feb 2022. URL `\url{https://www.sciencedirect.com/science/article/pii/S0092867421015774}`.

[8] B. Sosa-Pineda, K. Chowdhury, and M. Torres. The pax4 gene is essential for differentiation of insulin-producing beta cells in the mammalian pancreas. *Nature*, pages 1–4, Mar 1997. URL \url{https://www.nature.com/articles/386399a0#citeas}.

[9] Mastracci TL, Wilcox CL, Arnes L, Panea C, Golden JA, May CL, and Sussel L. Nkx2.2 and arx genetically interact to regulate pancreatic endocrine cell development and endocrine hormone expression. *Developmental Biology*, pages 1–10, Nov 2011. URL \url{https://www.sciencedirect.com/science/article/pii/S0012160611011730}.

[10] V.A. Traag, L. Waltman, and N.J van Eck. From louvain to leiden: guaranteeing well-connected communities. *Sci Rep*, pages 1–11, Mar 2019. URL \url{https://www.nature.com/articles/s41598-019-41695-z#citeas}.