Journal of XYZ, 2017, 1–8

doi: xx.xxxx/xxxx
Manuscript in Preparation
Paper

PAPER

SlowMoMan: A web app for discovery of important features along user-drawn trajectories in 2D embeddings

Kiran Deol[1], Griffin M. Weber[2] and Yun William Yu[3, *]

[1]University of Alberta and [2]Harvard Medical School and [3]University of Toronto

*ywyu@math.toronto.edu

Abstract

Nonlinear low-dimensional embeddings allow humans to visualize high-dimensional data, as is often seen in bioinformatics, where data sets may have tens of thousands of dimensions. However, relating the axes of a nonlinear embedding to the original dimensions is a nontrivial problem. In particular, humans may identify patterns or interesting subsections in the embedding, but cannot easily identify what those patterns correspond to in the original data. Thus, we present SlowMoMan (SLOW MOtions on MANifolds), a web application which allows the user to draw a 1-dimensional path onto the 2-dimensional embedding. Then, by back projecting the manifold to the original, high-dimensional space, we sort the original features such that those most discriminative along the manifold are ranked highly. We show a number of pertinent use cases for our tools, including trajectory inference, spatial transcriptomics, and automatic cell classification.

Software availability: https://yunwilliamyu.github.io/SlowMoMan/
Code availability: https://github.com/yunwilliamyu/SlowMoMan

Key words: Visualization; embeddings; manifolds

Introduction

The surge of big data in bioinformatics often means data sets may contain hundreds of thousands of dimensions/features [1, 2, 3]. Such data is impossible to visualize without embedding. A number of linear and non-linear embedding algorithms exist, such as PCA [4], UMAP [5], and t-SNE [6]. Visualizations of embeddings of high-dimensional data allow users to visually identify areas of significance that may not be flagged as important by quantitative metrics alone. However, understanding the behaviour of the original features within such areas is nontrivial when the number of dimensions is too large to be manually examined.

In particular, there is a need for automatic extraction of discriminative features [7] within user-defined subsections of the embedded data. Indeed, automatically determining discriminative features on labelled data (and then using those features for prediction of labels on unlabelled data points) is one of the classic foundational problems in machine learning, and has been solved in many ways using tools such as regression analysis [8], support vector machines (SVMs) [9], and deep neural networks [10] to name just a few. Traditionally, the labelled data had to be predefined, but more recently, interactive tools have been developed to allow users to label sparse data on-the-fly [11]. In interactive labelling tools, the labels are very often categorical: "is a cat present in the photo?", "is a cell cancerous?", etc. There are many techniques for prediction of ordinal (non-categorical) labels, but not as much in interactive applications. Indeed, interactive prediction of ordinal labels seems quite difficult, but simply extracting important features is more accessible.

Hence, in this manuscript, we are interested in providing automatic extraction of features on non-categorical labels corresponding to real-time user-specified trajectories/pseudo-time in datasets. What constitutes a notable area is dependent on the user's domain

Compiled on: August 19, 2022.
Draft manuscript prepared by the author.

knowledge, expectations, and purpose. For example, single cell trajectory inference seeks to identify continuous processes in single cell data, often corresponding to a notion of pseudotime in cell differentiation [12]. Often, such trajectories are near-to-obvious to the human eye when embedded in 2D, but the relationship between the 2D embedding and the original feature space may be complicated. Users can quickly identify potential trajectories, but identifying the variables that are important to that trajectory may involves going back to the original data. Furthermore, users need to remain cognizant of spurious path-like structures in an embedding, which may appear due to the optimization criteria of various embeddings—t-SNE is well-known to in certain parameter ranges to promote long thread-like regions, even when the underlying data do not strongly support it [13].

Thus, we present SlowMoMan, an in-browser tool which allows users to draw a custom, 1-dimensional path over notable areas of their embedded data. By examining the visualization of the embedding and the distribution of the original dimensions along possible trajectories in the embedding, users can quickly draw potential trajectories which they can compare and analyze. Once the user has drawn a manifold over a notable area, we run a nearest-neighbours algorithm on the points of the manifold to locate a respective back-projection in the original space (see. Fig. 1d). Along this back-projection, we examine the behaviour of each of the original features and for each of these features, we compute a user-chosen metric for discriminative ability.

Importantly, because the user has drawn a path, rather than provided a binary label, 'discrimination' here corresponds to measuring how important that feature seems to be to determining the position along the path. Users may choose from autocorrelation or the harmonic sum of the magnitudes of the fast Fourier transform, two metrics we have found to be both quick-to-compute and that correspond well with a human intuition of path-importance. Based on the metric scores, we sort the original features. Using metrics to sort the original features is a type of filter feature selection method which ensures SlowMoMan is fast and does not impose any assumptions on the data set (as opposed to wrapper or embedded feature selection methods, which rely on learning algorithms to rank features' importance) [3]. The features ranked highly are then listed and their exact distributions along the manifold are displayed for the user, who can then manually validate the feature selection.

Methodology

Overview

SlowMoMan runs locally in-browser using the Javascript engine and relies on the user's machine for all computation and storage. Thus, users do not need to install any additional software and can process their data sets without any third-party servers ever seeing their data. We have tested SlowMoMan on commodity personal laptops using both Firefox and Chrome. A 2017 Lenovo laptop with an Intel i7-8550U CPU and 16 GB of RAM is sufficiently fast to run SlowMoMan interactively with only a few seconds of loading delay for large data files, and any modern browser works. SlowMoMan will accept CSV files of the user's 2D embedding and the original data, where columns contain the features and rows contain the samples. Once uploaded, the user will see the visualization of the embedded data. On this visual, the user can draw their own 1-dimensional manifold over any areas of interest (Fig. 1c). A nearest neighbours algorithm is then used to back-project the user's custom manifold to the original space (Fig. 1d). Next, we observe the behaviour of each of the original dimensions along the back-projection of the user's manifold (Fig. 1e).

Using the behaviour of each of the original dimensions along the back-projection of the user's manifold, SlowMoMan will apply a number of possible metrics to sort the original dimensions, such that the dimensions most discriminative along the user's manifold are ranked highest for further examination. Users will see a graph of the value of each of those recommended discriminative variables plotted against the time parameterization of the drawn trajectory. Users can also select specific variables to plot, if they have expert knowledge on which of the highest ranking variables are likely to be important.

Canvas and back-projection

In our version 1 implementation, we fixed a canvas size of 512x512 pixels, and the 2D embedding is resized to fit into that space. This design choice was made for a combination of efficiency and reproducibility: (1) by setting the visual display to match the 2D embedding, determining the appropriate back-projection is made easier, and (2) we provide the user-drawn path as a series of key points in the canvas in a text box, and fixing the canvas size for all users ensures that those key points remain correct and portable.

However long the user-drawn path, we internally convert the path into 512 evenly spaced points on the canvas approximating that path. This is necessary because the mouse-over events vary across browsers, OS mouse sensitivity, and the mice themselves. If a user slowly drags a path, there will be many more mouse-over events than if a user draws the same path quickly; hence the normalization. The choice of $n = 512$ was largely arbitrary, but chosen to match the canvas size, and as a power of 2 for efficiency of the FFT computation.

Because SlowMoMan internally discretizes both the path and the canvas, back-projection via nearest neighbors is fairly straightforward. For each point along the path, we find the nearest data point on the canvas, and store the original features for that data point. Because of the discretization, there may sometimes be multiple such points, in which case we simply average features across all of those points. One might argue that it would be better to arbitrarily choose a point instead of averaging features, but in practice, it does not seem to make a difference as this is a sufficiently rare event.

Feature-ranking metrics

Depending on the data or the goals of the user, which features are most discriminative may vary. We explored a variety of possible metrics for users to choose from, though most performed fairly poorly (e.g. variance)—while variance is an accessible metric, its inability to account for the time-dependency along a path yields inconsistent results. The metrics that were most helpful were those that find periodicity along the path. As such, the metrics available in SlowMoMan include autocorrelation and the harmonic sum of the fast Fourier transform magnitudes (*FFT Score*).

Autocorrelation

The autocorrelation of a variable x can intuitively be thought of as the level of correlation that x has with a delayed version of itself [15]. Formally, let the autocorrelation be defined as

$$r_k = \frac{\sum_{t=k+1}^{n}(x_t - \bar{x})(x_{t-k} - \bar{x})}{\sum_{t=1}^{n}(x_t - \bar{x})^2} \tag{1}$$

where $\bar{x}$ is the mean of x and k is the lag value. Thus, SlowMoMan computes r_k for each of the original dimensions along the back-projection of the embedding, using a lag value of $k = 22$.

Intuitively, using autocorrelation picks out features that repeat

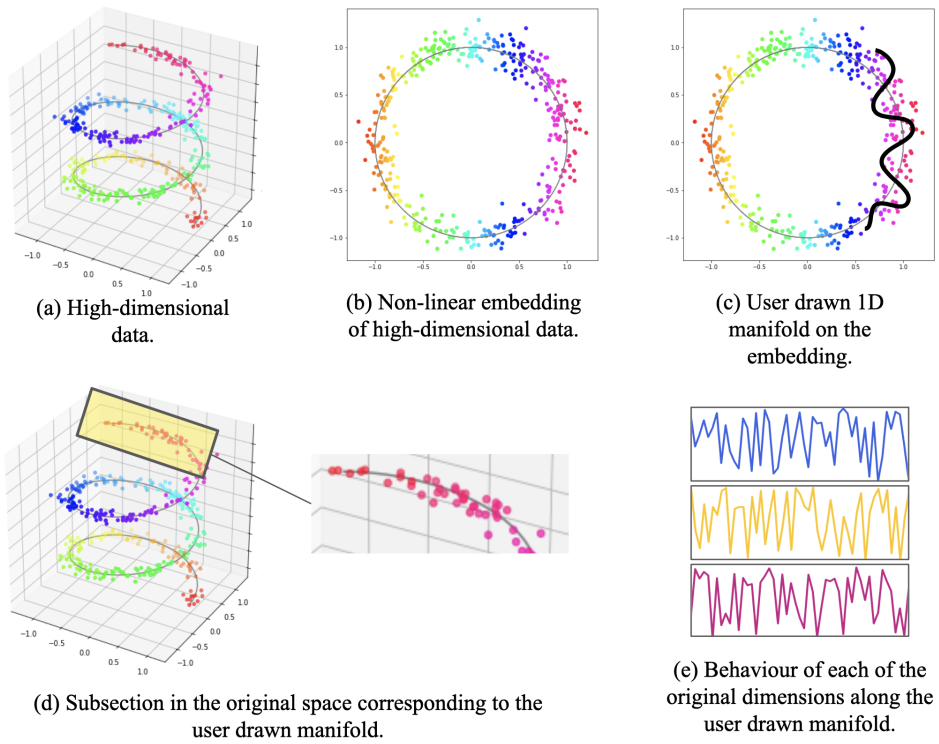

(a) High-dimensional data.
(b) Non-linear embedding of high-dimensional data.
(c) User drawn 1D manifold on the embedding.
(d) Subsection in the original space corresponding to the user drawn manifold.
(e) Behaviour of each of the original dimensions along the user drawn manifold.

Figure 1. In most use cases, data with hundreds or thousands of dimensions would be best suited for SlowMoMan. (a) However, to visualize an example of high-dimensional data, we present a toy 3-dimensional dataset (the typical Swiss Roll dataset with 3 dimensions/features). (b) A non-linear embedding allows visualization in 2 dimensions, but it is not clear how these axes relate to the original features. (c) Users may use SlowMoMan to draw a path in the 2D embedding, which ideally corresponds to something close to a 1D manifold in the original space (d). Then, we can analyze the behavior of the original features along that path, highlighting and sorting by the features that vary the slowest or are most autocorrelated.

themselves along the path. That kind of periodicity may suggest that the feature is one of the determinants of the path. However, one disadvantage of using autocorrelation is that we need to specify a lag value in advance. This choice of lag value of $k = 22$ was found empirically to work reasonably well in our experimentation, so that is the default lag, but that varies by application. Due to the need to set a parameter, autocorrelation should only be used by advanced users with a reasonable guess as to the lag that should be set.

Fast Fourier Transform

That of course leads us to desire a parameter-less way to find features that have some periodicity. To this end, we turn to standard signal processing tools and the Fourier transform. The fast Fourier transform is an efficient algorithm for computing the discrete Fourier transform [16]. The discrete Fourier transform decomposes a sequence of values into a sum of discrete sinusoidal waves, each with a distinct magnitude and frequency. Formally, if we have a sequence of k values $\{x_n\}$, we can transform it into a new sequence $\{F_m\}$ of sinusoids.

$$F_m = \sum_{n=0}^{k} x_n * [cos(\frac{2\pi}{k} * n * m) - i * sin(\frac{2\pi}{k} * n * m)] \quad (2)$$

Each of these new sinusoids will have a respective wavelength magnitude, which is the absolute distance between waves. Across all of the magnitudes m_n of each of the sinusoids for a specific sequence $\{x_n\}$, we can take a harmonic sum S over descending m_n

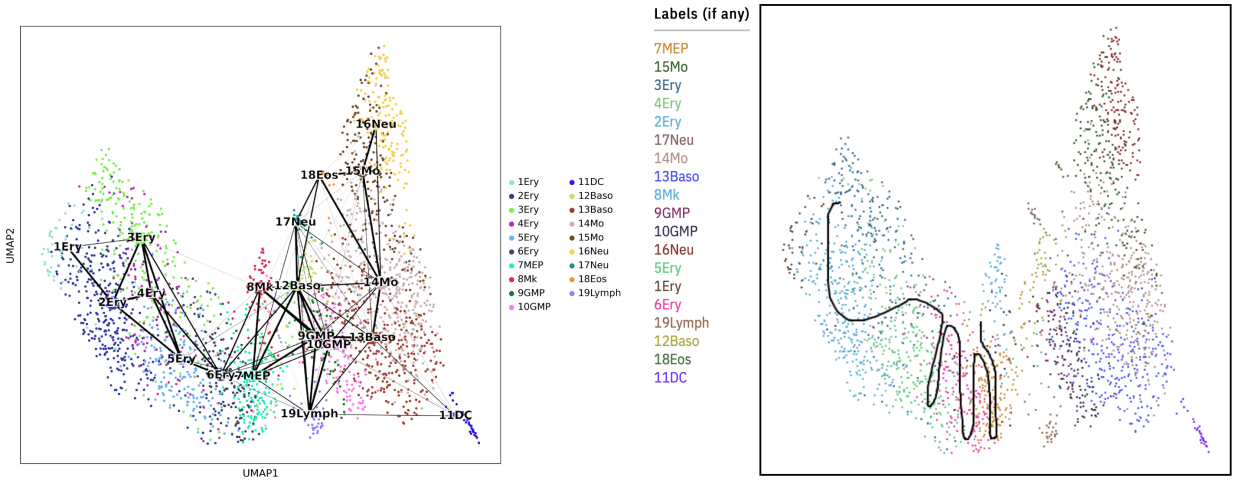
$$S = \sum_{n=1}^{k} m_n * \frac{1}{n} \quad (3)$$

which is then used as the custom "FFT Score" metric. The harmonic sum inherently lends greater weight to the sinusoids with lower frequencies—although, we naturally omit the constant m_0 term as we are only interested in periodic behavior.

Intuitively, we are not as interested in high-frequency variables because they are not strongly correlated with position on our path. Instead, low-frequency variables (i.e. slow motions) are features where if you know their values, you can roughly predict your position along the path. This allows us to simultaneously capture periodicity at all scales, while more heavily weighting longer periods, overcoming the shortcoming of having to set a lag parameter in autocorrelation.

Implementation details

Although SlowMoMan is presented as a website, there are no server-side components. All of SlowMoMan is implemented in client-side HTML5, Javascript, and CSS. The choice to use web technologies was made to take advantage of ready-to-use user interface elements, including the HTML5 Canvas, and Mouseover events for drawing. For the FFT computation, we use a 2014 version of the Project Nayuki Javascript FFT library [18], adapted by Cannam [19]. The autocorrelation computation does not use any libraries and is done directly in-app.



(a) PAGA Graph Overlaid onto UMAP Embedding.

(b) Custom Trajectory as used in SlowMoMan.

Figure 2. The results of the PAGA algorithm on the hematopoiesis data of Paul et al. [14] highlight strong connectivities between clusters 1-6. We use SlowMoMan to explore important features along a custom manifold between clusters 1-7. (a) The UMAP embedding of the hematopoiesis data of Paul et al. [14] with the PAGA connectivities between clusters represented as black edges. Thicker edges indicate stronger connectivities. Clusters correspond to cell type, with labels corresponding to those of Paul et al. [14]. Note also the prominent connectivities between clusters 1-7. (b) A screen capture of SlowMoMan with the user-drawn manifold used in the analysis.

Evaluation

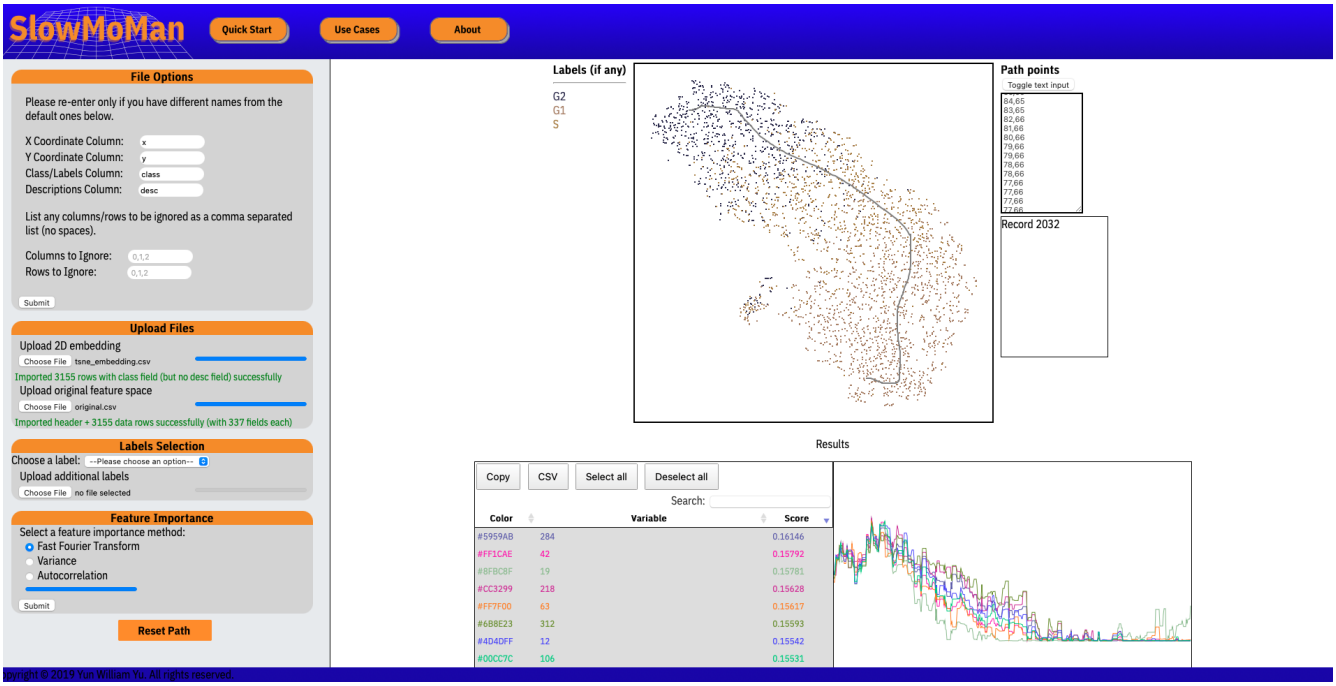
We demonstrate the utility of SlowMoMan through case studies and empirical analysis. In the first task, our prototypical use case, we show how SlowMoMan can be used to help interpret trajectories drawn on a 2D embedding. In this task, we consider two different types of trajectories, one where the features are actual genes in single-cell transcriptomic data, and another where the features are the activation space of a neural network on cell image classification. In the second task, we demonstrate using SlowMoMan on a more classic task related not to trajectories, but to simple clusters instead—we find important features disambiguating between two clusters of spatial transcriptomic cell data. In the last task, we use SlowMoMan to better characterize the nature of errors in automatic cell classification using single-cell transcriptomic data.

Trajectory Inference/Pseudotemporal Ordering

We apply SlowMoMan to datasets taken from Paul et al. [14] (Fig. 2) and Eulenberg et al. [17] (Fig. 3). Physical limitations in the technologies used to obtain gene expression data often make it impossible to track dynamic processes in single cells, such as the cell cycle or cell differentiation. Trajectory inference methods seek to infer such processes by analyzing similarities in the gene expression of cells to order them along a continuous interval. A large number of algorithms exist for such a purpose and Saelens et al. [21] benchmark 45 of these methods on their accuracy, scalabilty, stability, and usability. SlowMoMan inherently complements such methods due to its focus on trajectories, as opposed to standard tools which often focus on analyzing entire clusters. In particular, SlowMoMan may be used to identify important features along a given trajectory or to quickly explore alternative, user-defined trajectories.

Trajectory Inference in Hematopoiesis of Mouse Bone Marrow Cells
Data collected by Paul et al. [14] was retrieved through the *paul15()* method of Wolf et al. [22]. The raw data contains 3,451 gene counts across 2,730 labelled bone marrow cells of mice. The clusters correspond to cell type. Numbers are prefixed to the cluster labels to distinguish between cells that may belong to the same cell type, but are at different levels of hematopoietic differentiation. In order to identify potential trajectories we use PAGA [23], one of the best overall trajectory inference methods in Saelens et al. [21]. From a k-nearest neighbours (kNN) graph of single-cells and the associated clusters (partitions) of these cells, PAGA will compute a new graph representing the connectivities of the partitions. Obtaining the data, preprocessing, and running PAGA on this dataset were all accomplished through the Python package Scanpy, which contains methods for trajectory inference [22]. In particular, we use UMAP with the euclidean distance metric to compute a single-cell connectivity graph with a neighbourhood size of 20. We use the existing clusters of Paul et al. [14] as the partitions. We then run PAGA to obtain the graph of connectivities between partitions, as shown in Fig. 2.

A notable set of connectivities are those among clusters 1 to 7, which Paul et al. [14] hypothesize represent a continuum of erythrocyte differentiation. SlowMoMan is used to draw a trajectory between these clusters (see 2) and using the FFT score to sort, obtain a list of the 100 most important features out of 3451. Notably, a number of key transcription factors identified in Fig. 1C of Paul et al. [14] are ranked within the top 20 genes of this trajectory by SlowMoMan, includinig *Apoe*, *CPOX*, *Hba-a2*, and *Car2*. Within the top 100 genes, SlowMoMan also identified *Gata2* and *Car1*. Although not previously associated with erythropoiesis, Paul et al. [14] noted the significance of *Cited4* and *Phf10* in clusters 1-7. Interestingly, SlowMoMan also ranks these genes within the top 50 most important genes of the trajectory. Although SlowMoMan was able to identify a number of key genes, due to these genes being spread across the list of top 100 features, a user without prior knowledge may not realize their significance. Indeed, users with prior knowledge as which features are most important may focus on using SlowMoMan to experiment with possible trajectories that capture as many of the important features as possible, as illustrated in the current example. In contrast, users without prior expectations on the features may instead use SlowMoMan is to gather a list of important features along a specific trajectory which may be investigated further with more conclusive methods, as we show in



(a) In-app screen capture of SlowMoMan

Figure 3. SlowMoMan identifies the most informative nodes within a neural network trained to process images of Jurkat cells in different stages of the cell cycle [17]. The t-SNE embedding of the neural network activation space representations of each cell is presented near the center of the screen capture. Notice the clusters already have a fairly clear trajectory and are indeed in their correct order. The top 7 most important nodes identified by SlowMoMan are displayed in the left half of the lower two boxes. The behaviour of the top 7 nodes along the path of the trajectory is displayed in the right half of the lower two boxes. Notably, all of the nodes move in a fairly similar pattern, as to be expected with a good trajectory.

the next example.

Deep Learning for Cell Cycle Reconstruction

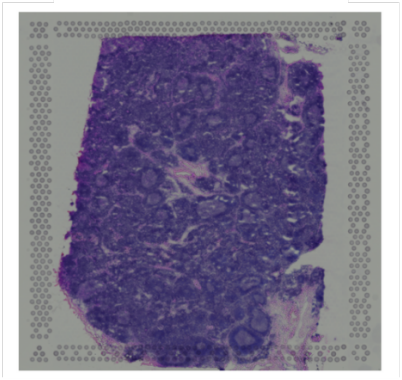
In contrast to pseudotime algorithms, Eulenberg et al. [17] apply deep learning to a dataset of 32,266 images of Jurkat cells, each in one of seven stages in the cell cycle (G1, S, G2, Prophase, Anaphase, Metaphase, and Telophase). The neural network learns a new representation of the data in which the data is much closer to being linearly separable with respect to each of the classes. This implies that distances in this new space are more biologically faithful than those of the original representation. Eulenberg et al. [17] thus use the activations in the last layer of the network as the representation of the data. A 2-dimensional embedding of the 336-dimensional activation space represention is obtained with t-SNE. The resulting clusters are distinct and appear in their correct orders, indicating a successful reconstruction of the cell cycle. However, a practitioner may wish to understand how the architecture of the network leads to such an embedding. Unlike more traditional representations where domain knowledge of the features, such as genomic biomarkers, can be used to explain the results of an embedding algorithm, the activation space representation inherently creates features which are not easily interpretable to humans. Thus, SlowMoMan stands to supplement such a method by allowing users to both identify which nodes are most important along some subsection of their data and to study how the activations of each node change along that subsection.

In order to use SlowMoMan, a subsample with 3,157 cells of the original dataset of Jurkat cells obtained by Eulenberg et al. [17] was accessed from `https://falexwolf.de/scanpy_usage/170529_images/`. This subsample contains cells only within the G1, S, and G2 phases of the cell cycle. The dataset contains the 336-dimensional activation space representations of the 3,157 Jurkat cells, along with their phase labels. We then use t-SNE to obtain a 2-dimensional embedding of the data. As expected, the clusters are quite distinct and are in their expected order (G1 to S to G2). This embedding implies a roughly linear trajectory right through each of the clusters, as illustrated in 3.

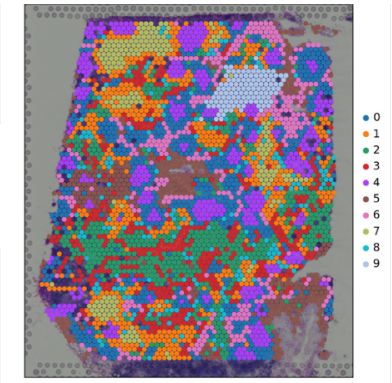
Using the FFT score, SlowMoMan identifies the top 7 most important nodes as: 284, 42, 19, 218, 63, 312, 12, and 106. For a practitioner, this suggests that the nodes identified as the least important could be removed or modified to possibly improve the results of the neural network. This also shows the practitioner which nodes are most discriminative in the network, especially along the main trajectory. SlowMoMan also allows the user to examine the distribution of the all of the nodes along the given trajectory. In Fig. 3, we see that the top 7 nodes remain fairly steady early in the trajectory, and become inactive near the end. When evaluating a "good" trajectory, we intuitively expect to rely on genes which move slowly and remain in sync along the trajectory. Indeed, this is what we see with the top 7 nodes given by SlowMoMan, confirming that our intuitions as to what the trajectory is were correct.

Spatial Transcriptomics

We apply SlowMoMan to a Human Lymph Node Spatial Gene Expression Dataset by Space Ranger 1.0.0, as provided by 10x Genomics [20] (Fig. 4). A recent advancement in bioinformatics is the ability to associate gene expression data with its spatial distribution in a sample. Specifically, spatial transcriptomics enables researchers to locate the position of individuals cells (and the respective gene expression data of each cell) in a sample. Spatial transcriptomics is a strong candidate field for SlowMoMan, as clusters in a tissue sample are unlikely to

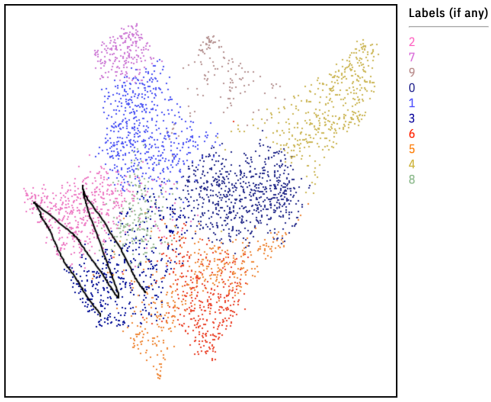


(a) Original Lymph Node Tissue Sample

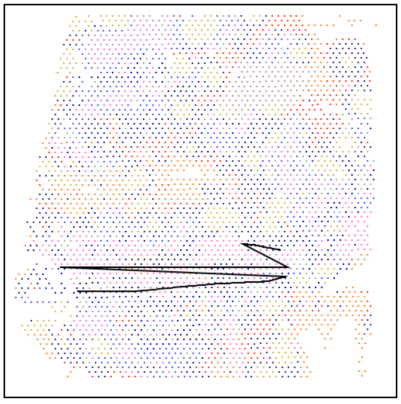


(b) Cellular Overlay on Lymph Node Sample

(c) UMAP Clustering



(d) UMAP clustering with user-drawn trajectory as used in SlowMoMan



(e) Cellular Overlay on Lymph Node Sample with user-drawn trajectory as used in SlowMoMan

Figure 4. SlowMoMan applied to a spatial transcriptomics dataset to determine important features distinguishing between two nearby clusters. (a) The original image of the human lymph node tissue sample obtained by 10x Genomics [20]. (b) The original image of the human lymph node tissue sample with the identified single cells and their respective UMAP classes overlaid. (c) The UMAP clustering of the single cells identified in the tissue sample. (d) The UMAP clustering of the single cells identified in the tissue sample, with a simplified overlay of the lines used during the analysis with SlowMoMan drawn as black lines. (e) The original image of the human lymph node tissue sample, with a simplified overlay of the lines used during the analysis with SlowMoMan drawn as white lines.

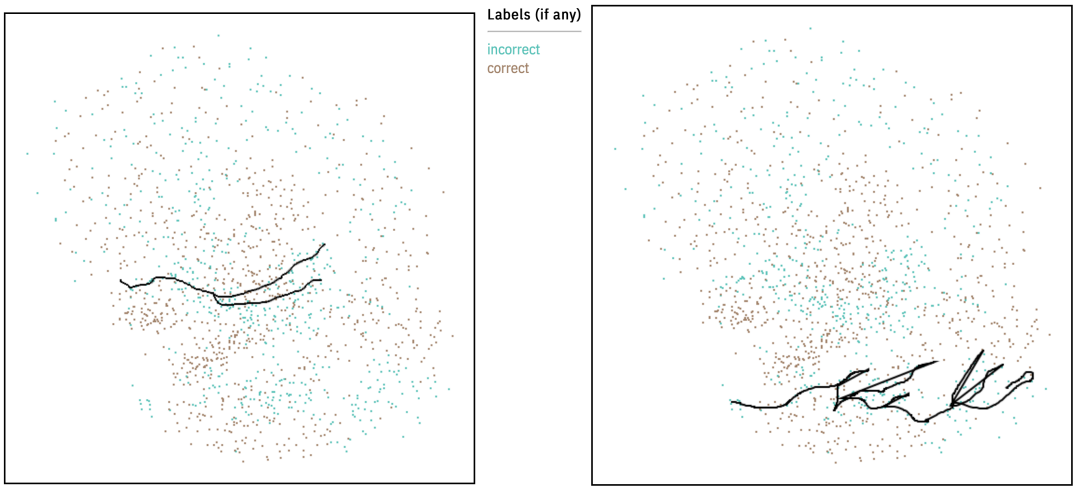
be perfectly distinct. Indeed, while the clusters of the corresponding embedding of the cells of tissue sample cells tend to remain fairly coherent, the clusters in the tissue sample itself are often intermingled are broken into a number of subclusters, as illustrated in 4. In such cases, analyzing the features of entire clusters would not be as informative as analyzing the individual subsections of clusters and comparing them.

Thus, we apply SlowMoMan to a Human Lymph Node Spatial Gene Expression Dataset, provided by 10x Genomics [20] (Fig. 4). In particular, 10x Genomics used their Visium Spatial Gene Expression tool to analyze a sample of human lymph node tissue. Obtaining the data, preprocessing, visualization, and running the embedding algorithm were all accomplished through the Python package Scanpy, which also contains methods for spatial transcriptomics [22].

This dataset from 10x Genomics contains the spatial coordinates of each of the individually identified cells in the tissue sample, along with the gene expression levels of each of these cells. From the gene expression data, we use UMAP on a 50-dimensional PCA embedding to compute a graph with neighbourhoods of size 15. This neighbourhood graph is then used in the Leiden clustering algorithm of Traag et al. [25] to identify 10 within the data. Although these clusters are distinct (see 4), we cannot yet make any claims as to what known lymph cell subtypes these clusters may translate to. However, we can identify patterns between the spatial distribution of the cells and their UMAP distribution and use SlowMoMan to explain the genetic underpinnings for such patterns.

One notable observation is that clusters 2 and 3 (green and red) not only exist beside each other in the UMAP embedding, but are also frequently seen together in the tissue sample, with red appearing almost as a border to the green sections. In contrast to the embedding, the red cluster is broken up into many separate, smaller subclusters throughout the tissue sample. Computing the feature importance of the red cluster in its entirety would fail to capture any behaviour unique to aforementioned area of the red cluster which borders the green cluster. Thus, we first use SlowMoMan without considering any subclusters and analyze the red and green clusters in the UMAP embedding. Then, we compare the results to SlowMoMan being run on the aforementioned subcluster of the spatial distribution. Notice in Fig. 4 that the path drawn goes back-and-forth between the two clusters, unlike in the previous example; this artifically creates the periodicity that SlowMoMan tries to detect, allowing it to find not just discriminating features for paths, but also clusters.

Between the two runs of SlowMoMan shown in Fig. 4, we used the FFT score to obtain the top ten most important genes in both instances and found that genes IGHG1, IGHG2, IGHG3, IGHG4, and IGKC were all identified among the top 10 most important genes (out of



(a) kNN errors on pancreatic data with user-drawn path covering large erroneous region

(b) kNN errors on pancreatic data with user-drawn path covering other large erroneous region

Figure 5. SlowMoMan is used to understand the types of errors made by the kNN algorithm (with a neighbour size of 9) on an automatic cell classification task in single-cell transcriptomics [24]. We use UMAP to obtain a 2-dimensional embedding. The clusters represent which cells were classified correctly vs. incorrectly by the trained kNN model. Of the 581 incorrectly labelled cells, 78% were beta cells, 14% were gamma cells, and 8% were delta cells. None of the incorrectly identified cells were alpha cells. (a) A screen capture of SlowMoMan with the user-drawn trajectory over a fairly distinct subcluster of errors. Notable genes identified along this trajectory were NPY, PPY, and ZFP36. (b) Similar to (a), but with a user-drawn trajectory covering a different, but also fairly distinct subcluster of errors. Notable genes identified along this trajectory were G6PC2 and SCGB2A1.

19,686) in both the trajectory drawn on the UMAP embedding and the trajectory drawn on the spatial distribution. Notably, the Entrez Gene summary for each of these genes predicts that they are important to immunoglobulin receptor binding. Immunoglobulins are produced by B-lymphocytes, which are located primarily in the cortex of the lymph nodes. Given that the red and green clusters were observed mainly in one region of the tissue sample, we could the explain the importance of IGHG1, IGHG3, IGHG4, and IGKC as the result of the cells of the red and green clusters being located in the cortex of the lymph node tissue sample.

Automatic Cell Classification

We apply SlowMoMan to a single-cell transcriptomic dataset from Xin et al. [24] (Fig. 5). In single cell analysis, datasets are often too large for manual annotation and methods for automated classification of cell types are a promising alternative. A number of challenges arise when using such methods, however, such as choosing which method to use or evaluating the accuracy of a given method. SlowMoMan augments such methods by allowing users to interpret regions where multiple classification methods may differ, or by interpreting regions where a classification method fails.

For example, the pancreatic data obtained by Xin et al. [24] examines the transcriptomes of 1492 islet cells across 18 donors and includes four classes of islet cells: alpha, beta, gamma, and delta. As part of a survey of automatic cell classification methods, Abdelaal et al. [26] apply a number of classification methods to this dataset, including the kNN algorithm ($k = 9$). Replication of these methods found that kNN had an accuracy score of 59.10%, supporting the results of Abdelaal et al. [26]. SlowMoMan augments this method by enabling the interpretation of regions where kNN fails.

One explanation for the poor performance of the kNN algorithm was the imbalance in data: 60% were alpha cells, 32% were beta cells, 5% were gamma cells, and 3% were delta cells. This would also explain why all of the incorrectly identified cells were either beta, gamma, or delta cells. However, we also use SlowMoMan to understand the possible genomic causes for these errors.

In particular, we run SlowMoMan on two large and fairly distinct subclusters of kNN errors (see 5). Akin to the previous example, analyzing the feature importances across the entire cluster of errors would fail to capture any information unique to the two identified subclusters. Thus, we perform two runs of SlowMoMan, one for each of the previously identified subclusters of errors.

In the first subcluster (Fig. 5a), we identified 3 genes of interest: NPY, PPY, and ZFP36. In the second subsection, (Fig. 5b), we identified 2 genes of interest: G6PC2 and SCGB2A1. We note that in both subclusters, the aforementioned genes were all within the top 100 most important genes (out of 33890) of their respective trajectories. We also note that in this case, sorting with the autocorrelation metric led to more notable genes being identified than with the usual FFT score. The PPY gene is a known marker gene of gamma cells. Thus, its importance is to be expected, given the inability of the kNN model to recognize any of the gamma cells. The importance of the remaining four genes, however, can be explained by the discovery of four beta cell subtypes by Dorrell et al. [27]. Indeed, they identify the four subtypes as β_1, β_2, β_3, β_4. Within which there is the $ST8SIA1^+$ β_3/β_4 subtype, $ST8SIA1^-$ β_1/β_2 subtype, $CD9^+$ β_2/β_4 subtype, and the $CD9^-$ β_1/β_3 subtype. Notably, NPY and ZFP36 were found by Dorrell et al. [27] to be differentially expressed between the $CD9^-$ β_1/β_3 and $CD9^+$ β_2/β_4 subtypes. In particular, NPY had larger expression levels in the $CD9^-$ β_1/β_3 subtype, whereas ZFP36 had larger expression levels in the $CD9^+$ β_2/β_4 subtype. Thus, although the cells in the first subsection of errors (Fig. 5a), were primarily beta cells, the possibility of multiple subtypes of beta cells existing in that area may have contributed to the poor results obtained by the kNN model when classifying beta cells. Similarly, in the second subsection analyzed (Fig. 5b), the genes G6PC2 and SCGB2A1 were also identified by Dorrell et al. [27] as being differentially expressed between the $ST8SIA1^+$ β_3/β_4 and $ST8SIA1^-$ β_1/β_2 subtypes. Therefore, even with an imbalance in the data set, the analysis with SlowMoMan suggests the possibility that the unspecified subtypes within the beta cells may have contributed to the poor performance of the kNN model on classifying beta cells.

Conclusion

In this manuscript, we presented a new interactive web app, SlowMoMan, for visualization and exploration of important variables in low-dimensional embeddings. This web app runs entirely client-side using Javascript, HTML 5, and CSS, and allows for real-time usage on commodity laptops. By back-projecting user-drawn paths and ranking variables by proxies for their periodicity, users are able to identify the original features that are most informative for those paths.

The major advance of our work is in allowing users to drawn paths, rather than simply apply categorical labels, which we showed in our two trajectory analysis examples. As shown in our latter two examples, users are still able to examine features important for binary classifications by simply drawing appropriate paths doubling back on themselves. However, the paths lend themselves particularly well in cases where there is some underlying notion of paths and time, such as in cell differentiation and cell cycles. We hope tools like SlowMoMan will assist scientists in interpreting and analyzing low-dimensional embeddings.

Acknowledgments

K.D. was supported by a University of Toronto Data Science Institute Summer Undergraduate Data Science (SUDS) Research fellowship. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), (NSERC grant RGPIN-2022-03074).

References

1. Hilario M, Kalousis A. Approaches to dimensionality reduction in proteomic biomarker studies. Briefings in bioinformatics 2008;9(2):102–118.
2. Moore JH, Asselbergs FW, Williams SM. Bioinformatics challenges for genome-wide association studies. Bioinformatics 2010;26(4):445–455.
3. Townes FW, Hicks SC, Aryee MJ, Irizarry RA. Feature selection and dimension reduction for single-cell RNA-Seq based on a multinomial model. Genome biology 2019;20(1):1–16.
4. Ringnér M. What is principal component analysis? Nature biotechnology 2008;26(3):303–304.
5. Becht E, McInnes L, Healy J, Dutertre CA, Kwok IW, Ng LG, et al. Dimensionality reduction for visualizing single-cell data using UMAP. Nature biotechnology 2019;37(1):38–44.
6. Van der Maaten L, Hinton G. Visualizing data using t-SNE. Journal of machine learning research 2008;9(11).
7. Guyon I, Gunn S, Nikravesh M, Zadeh LA. Feature extraction: foundations and applications, vol. 207. Springer; 2008.
8. Freund RJ, Wilson WJ, Sa P. Regression analysis. Elsevier; 2006.
9. Noble WS. What is a support vector machine? Nature biotechnology 2006;24(12):1565–1567.
10. LeCun Y, Bengio Y, Hinton G. Deep learning. nature 2015;521(7553):436–444.
11. Berg S, Kutra D, Kroeger T, Straehle CN, Kausler BX, Haubold C, et al. Ilastik: interactive machine learning for (bio) image analysis. Nature methods 2019;16(12):1226–1232.
12. Ji Z, Ji H. TSCAN: Pseudo-time reconstruction and evaluation in single-cell RNA-seq analysis. Nucleic acids research 2016;44(13):e117–e117.
13. Wattenberg M, Viégas F, Johnson I. How to use t-SNE effectively. Distill 2016;1(10):e2.
14. Paul F, Arkin Y, Giladi A, Jaitin DA, Kenigsberg E, Keren-Shaul H, et al. Transcriptional Heterogeneity and Lineage Commitment in Myeloid Progenitors. Cell 2015;163(7):1663–1677. https://www.sciencedirect.com/science/article/pii/S0092867415014932.
15. Gubner JA. Probability and random processes for electrical and computer engineers. Cambridge University Press; 2006.
16. Cooley JW, Tukey JW. An algorithm for the machine calculation of complex Fourier series. Mathematics of computation 1965;19(90):297–301.
17. Eulenberg P, Köhler N, Blasi T, Filby A, Carpenter AE, Rees P, et al. Reconstructing cell cycle and disease progression using deep learning. Nature Communications 2017 Sep;8(1):463.
18. Nayuki P, Free small FFT in multiple languages;. Accessed: 2022-08-01. https://www.nayuki.io/page/free-small-fft-in-multiple-languages.
19. Cannam C, Javascript DSP Tests;. Accessed: 2019-08-01. https://code.soundsoftware.ac.uk/projects/js-dsp-test/repository/show/fft.
20. 10x Genomics, Human Lymph Node Spatial Gene Expression Dataset by Space Ranger 1.0.0;. Accessed: 2022-08-01. https://support.10xgenomics.com/spatial-gene-expression/datasets/1.0.0/V1_Human_Lymph_Node?
21. Saelens W, Cannoodt R, Todorov H, Saeys Y. A comparison of single-cell trajectory inference methods. Nature Biotechnology 2019 May;37(5):547–554. https://doi.org/10.1038/s41587-019-0071-9.
22. Wolf FA, Angerer P, Theis FJ. SCANPY: large-scale single-cell gene expression data analysis. Genome Biology 2018 Feb;19(1):15. https://doi.org/10.1186/s13059-017-1382-0.
23. Wolf FA, Hamey FK, Plass M, Solana J, Dahlin JS, Göttgens B, et al. PAGA: graph abstraction reconciles clustering with trajectory inference through a topology preserving map of single cells. Genome Biology 2019 Mar;20(1):59. https://doi.org/10.1186/s13059-019-1663-x.
24. Xin Y, Kim J, Okamoto H, Ni M, Wei Y, Adler C, et al. RNA Sequencing of Single Human Islet Cells Reveals Type 2 Diabetes Genes. Cell Metabolism 2016;24(4):608–615. https://www.sciencedirect.com/science/article/pii/S155041311630434X.
25. Traag VA, Waltman L, van Eck NJ. From Louvain to Leiden: guaranteeing well-connected communities. Scientific Reports 2019 Mar;9(1):5233.
26. Abdelaal T, Michielsen L, Cats D, Hoogduin D, Mei H, Reinders MJT, et al. A comparison of automatic cell identification methods for single-cell RNA sequencing data. Genome Biology 2019 Sep;20(1):194. https://doi.org/10.1186/s13059-019-1795-z.
27. Dorrell C, Schug J, Canaday PS, Russ HA, Tarlow BD, Grompe MT, et al. Human islets contain four distinct subtypes of β cells. Nature Communications 2016 Jul;7(1):11756. https://doi.org/10.1038/ncomms11756.