Model Reduction and Trajectory Inference

BY ZHIJUN ZENG

2023年7月27日

1 Model Reduction

1.1 MDS

The basic idea of MDS is to project the high dimensional data points into low dimension space while preserve the similarity between each other.

Given N data points $\{x_i\}_{i=1}^N, x_i \in \mathbb{R}^{1\times m}$, one can obtain the distance matrix $D \in \mathbb{R}^{N\times N}$. Suppose we have projected our data into n dimension space $\{z_i\}_{i=1}^N, z_i \in \mathbb{R}^{1\times n}$,Z$\in\mathbb{R}^{n\times N}$.

$$d_{\mathrm{ij}} = \|z_i\|^2 + \|z_j\|^2 - 2z_i^\top z_j.$$

We assume that

$$\sum_{i=1}^N z_i = 0$$

then summing up both side gives

$$\sum_{i=1}^N d_{\mathrm{ij}} = \sum_{i=1}^N \|z_i\|^2 + N\|z_j\|^2, \sum_{j=1}^N d_{\mathrm{ij}} = \sum_{j=1}^N \|z_j\|^2 + N\|z_i\|^2$$

and further

$$\sum_{i=1}^N \sum_{j=1}^N d_{\mathrm{ij}} = 2N\sum_{i=1}^N \|z_i\|^2$$

Let $B = Z^\top Z \in \mathbb{R}^{N \times N}$,

$$b_{\mathrm{ij}} = -\frac{1}{2}(d_{\mathrm{ij}}^2 - d_{i,.}^2 - d_{.,j}^2 + d_{.,.}^2) \tag{1}$$

Then we can compute B using existing D. To implement model reduction, we decompose B

$$B = V\Lambda V^\tau \, (\mathbb{R}^{N \times N}, \mathbb{R}^{N \times N}, \mathbb{R}^{N \times N})$$

We only preserve the n largest eigen value

$$Z = \sqrt{\tilde{\Lambda}}\tilde{V}^T \, (\mathbb{R}^{n \times n}, \mathbb{R}^{n \times N})$$

1.2 ISOMAP

The difference between ISOMAP and MDS is that, ISOMAP construct the distance matrix via nearest neighbourhood connection and Shortest Path.

1. Setting the nearest neighbourhood number k, construct connection graph and adjecant matrix.
2. Constructing the distance matrix via shortest path searching.
3. Compute$B$ via (1).
4. Compute Λ, V.
5. Obtain Z.

1.3 Diffusion Map

Diffusion maps are a non-linear technique. It achieves dimensionality reduction by re-organising data according to parameters of its underlying geometry.

The connectivity between two points x, y is defined as the probability of jump from x to y, which is often model as

$$\operatorname{con}(x, y) \propto k(x, y)$$

where $k(x, y) = \exp\left(-\frac{|x - y|^2}{\alpha}\right)$.

The neighbourhood of x is those y such that $k(x, y) > \varepsilon$ for some $1 > \varepsilon > 0$.

The diffusion kernel satisfies

1. k is symmetric
2. k is positive preserving: $k(x, y) \geqslant 0$.

A scale parameter d_X such that

$$\frac{1}{d_X} \sum_{y \in X} k(x, y) = 1.$$

Then the final connectivity is defined as

$$\mathrm{con}(x,y) = p(x,y) = \frac{1}{d_X} k(x,y)$$

Then one can compute a diffusion matrix(Probability transfer) $P_{\mathrm{ij}} = p(X_i, X_j)$. The t step transform matrix

$$P^t = (P)^t$$

Then we define a diffusion metric based on this structure. The metric measures the similarity of two points in the observation space as the connectivity (probability of "jumping") between them.

$$D_t(X_i, X_j)^2 = \sum_{u \in X} |p_t(X_i, u) - p_t(X_j, u)|^2 = \sum_k |P_{\mathrm{ik}}^t - P_{\mathrm{jk}}^t|^2.$$

The Euclidean distance between two mapped Y_i, Y_j is

$$\|Y_i - Y_j\|_E^2 = D_t(X_i, X_j)^2$$

Since $P = D^{-1}K$ is the normalized diffusion matrix, let

$$Y_i{}' = \begin{bmatrix} \lambda_1^t \phi_1(i) \\ \dots. \\ \lambda_n^t \phi_n(i) \end{bmatrix}$$

where $\phi_1(i)$ indicates the i-th element of the first eigenvector of P, the eigenvalue indicates the importance of each dimension. The reduction is achieved by retaining m dimensions associated with the dominant eigenvectors, which ensures that $\|Y_i{}' - Y_j{}'\|$ approximates the diffusion distance.

Algorithm 1 Basic Diffusion Mapping Algorithm

```
INPUT: High dimensional data set $X_i$, $i = 0 \dots N-1$.

    1. Define a kernel, $k(x, y)$ and create a kernel matrix,
       $K$, such that $K_{i,j} = k(X_i, X_j)$.

    2. Create the diffusion matrix by normalising the
       rows of the kernel matrix.

    3. Calculate the eigenvectors of the diffusion matrix.

    4. Map to the $d$-dimensional diffusion space at time
       $t$, using the $d$ dominant eigenvectors and -values as
       shown in (9).

OUTPUT: Lower dimensional data set $Y_i$, $i = 0..N-1$.
```

1.4 PCA

Assume the dataset consist of n data sample $\{x_i\}_{i=1}^n$, $x_i \in \mathbb{R}^p$. A linear projection is

$$y_i = \mathrm{Wx}_i \in \mathbb{R}^q, W = [w_1, \dots w_q]^T \in \mathbb{R}^{q \times p}.$$

Let $\bar{x} = \sum_i x_i / n$,$\mu = \sum_i w^T x_i / n$ for some w, $\mathrm{cov}(X) = \Sigma = \frac{1}{n} \sum_i (x_i - \bar{x})(x_i - \bar{x})^T$.Then the variance of projected data is

$$\sigma^2 = \omega^T \Sigma w$$

We want

$$\hat{w} = \mathrm{argmax}_w w^\top \Sigma w, s.t. w^\top w = 1$$

By lagrange multiple

$$\Sigma w = \lambda w$$

which means that w should be the largest eigenvector of Σ.

Then we consider the eigen decomposition

$$\Sigma = G \Lambda G^\top$$

where $G \in \mathbb{R}^{p \times p}$ is orthogonal with

$$G = [g_1, \dots g_p]$$

then $W = [g_1, \dots, g_q]^\top$,

$$Y^\top = \mathrm{WX}^\top$$

where $X = \begin{bmatrix} x_1^T \\ \dots. \\ x_n^T \end{bmatrix}$.

1.5 t-SNE

To model the distance between samples, t-SNE use the transition probability

$$P(j|i) = \frac{S(x_i, x_j)}{\sum_{k \neq i} S(x_i, x_k)}, j \neq i$$

where $S(x_i, x_j)$ is the similarity between i and j. For the samples in the projection space,

$$Q(j|i) = \frac{S'(z_i, z_j)}{\sum_{k \neq i} S'(z_i, z_k)}, j \neq i$$

We want to find projected samples $(z_1, \ldots, z_n)$ to minimize the KL divergence of both distributions

$$\{z_i^*\} = \mathrm{argminL}(\{z_i\}) = \mathrm{argmin} \sum_{i=1}^{n} \sum_{j \neq i} P(j|i) \log\left(\frac{P(j|i)}{Q(j|i)}\right)$$

here the similarity of t-SNE is given by

$$S(i, j) = \exp(-\|x_i - x_j\|^2 / (2\sigma_i)^2).$$

$$S'(i, j) = [1 + \|z_i - z_j\|^2]^{-1}$$

1. Compute $p_{\mathrm{ij}} = P(j|i)$ by definition
2. Generate $z_1, \ldots, z_n$ as low-dim data,denoted as initial value $Z^{(0)}$
3. Compute $q_{\mathrm{ij}} = Q(j|i)$ and obtain gradient

$$\frac{\partial L}{\partial Z^{(0)}} = \left(\frac{\partial L}{\partial Z_1^{(0)}}, \frac{\partial L}{\partial Z_2^{(0)}}, \ldots, \frac{\partial L}{\partial Z_n^{(0)}} \right)^T$$

4. Given learning rate $\eta,$ update

$$Z^{(1)} = Z^{(0)} + \eta \frac{\partial L}{\partial Z^{(0)}}$$

5. Repeat 3-4

2 Trajectory Inference

2.1 Principal Curve

We observe $n \times p$ matrix $\boldsymbol{X}$ with row $\boldsymbol{x}_i$. Our goal is to extract a lower-dimensional set of factors $\boldsymbol{y} = (y_1, \ldots, y_n)^T$. We can think of the following model:

$$\boldsymbol{x}_i = f(y_i) + \boldsymbol{\varepsilon}_i$$

where $\boldsymbol{\varepsilon}_i$ are errors with mean 0 and variance $\sigma^2 \boldsymbol{I}$ and $\boldsymbol{f}: \mathbb{R}^1 \to \mathbb{R}^p$ is a continuous function called a curve.

Curves are often parametrized by arc length , i.e. $y_1 - y_0$ gives the distance from $\boldsymbol{f}(y_1) - \boldsymbol{f}(y_0)$ along the curve.

For curve $\boldsymbol{f}$, the projection index $y_f(\boldsymbol{x})$ maps the observation $\boldsymbol{x} \in \mathbb{R}^p$ to the point on $\boldsymbol{f}$ that is closest, returning the score. If there are several such points, it returns the largest.

$$y_f(\boldsymbol{x}) = \sup_y \left\{ y \colon \|\boldsymbol{x} - \boldsymbol{f}(y)\| = \inf_\mu \|\boldsymbol{x} - \boldsymbol{f}(\mu)\| \right\}$$

The principle curves estimates the following least squares objective function

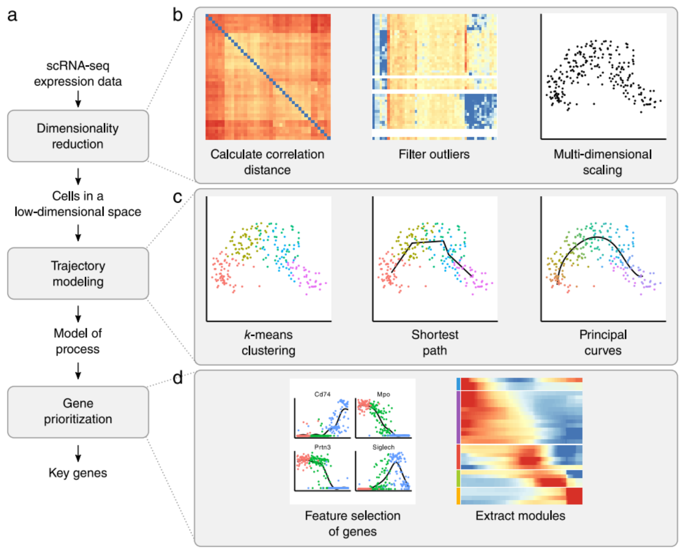
$$\min_{\boldsymbol{f}} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{f}(y_f(\boldsymbol{x}_i))\|^2.$$

Principle curve algorithm:

1. Initialize: Let iteration counter $h = 1$,set $y^{(0)} = \boldsymbol{X}\boldsymbol{u}$ where $\boldsymbol{u}$ is the first PC vector
2. Smoothing step: with $\hat{\boldsymbol{y}}^{(h-1)}$ fixed, estimated $\hat{\boldsymbol{x}}_i^{(h)}$ with a smoother.
3. Projection step: with $\hat{\boldsymbol{X}}^{(h)}$ fixed , use projection index to update the score $\boldsymbol{y}^{(h)}$ so that they have unit speed.
4. Loop.Increment h and return Step 1 while change in the objective function si above some threshold

2.2 SCORPIUS

Comparable to other TI methods, SCORPIUS assumes that a given dataset contains the genome-wide expression profiles of hundreds to thousands of cells, which were uniformly sampled from a linear dynamic process.



The main step of SCORPIUS is dimensionality reduction, trajectory modeling and gene prioritization.

- DR: Compute the spearman correlation distance between all pairs of cells, and remove outliers. Then we perform MDS to reduce the dimension to n components.

- TM: An initial trajectory is constructed by k-means clustering and finding the shortest path through the centers.This initial trajectory is subsequently refined in an iterative way using the principal curves algorithm.

- GP:infers the degree to which a gene and its expression is involved in the dynamic process of interest. This is achieved by ranking the genes according to their ability to predict the ordering of cells from the expression data, using the Random Forest algorithm.

Here we define $\mathbb{C}$ as the collection of all cells, the distance is defined as

$$D_\rho(X,Y) = 1 - \frac{C(X,Y)}{2}$$

The outlierness of a cell is defined as

$$\mathrm{Out}_D(X) = \sum_{Y \in 10NN(X)} \frac{D(X,Y)}{10}$$

We assume the outlierness of all cells to be normally distributed and iteratively removes cells with largest outlierness and fit normal distribution.And then we perform MDS to reduce the dimension into n dimension space S

$$S = K \Lambda K^T$$

where Λ is a vector containing n largest eigenvalue of double centred D_ρ, K is a matrix containing the corresponding eigenvectors.

For trajectory inference, cells are clustered into k clusters with k-means in S . An initial trajectory is constructed by linking cell clusters through their shortest path using a custom distance function..

$$D(C_i, C_j) = D_{\text{Euclidean}}(C_i, C_j) \times D_{\text{KNN}}(C_i, C_j)$$

$$D_{\text{KNN}}(C_i, C_j) = \sum_{n=0}^{n<100} \text{Out}\left(C_i + (C_i - C_j) \times \frac{n}{100}\right)$$

Finally, SCORPIUS further optimizes this initial path using the Hastie and Stuetzle principal curves algorithm.

2.3 Component1

This method returns the first component of a principal component analysis (PCA) dimensionality reduction as a linear trajectory. This method is especially relevant as it has been used in a few studies already

2.4 Embeddr

Embeddr use laplacian eigenmaps to reveal the manifold of scRNA-seq data. First, high-variance genes are selected using spike-ins if available or by fitting coefficient of variation - mean curves.Given a cell-to-cell similarity matrix W_{ij}, one attemp to minimise the quantity

$$\sum_{\mathrm{ij}} W_{\mathrm{ij}} \| y_i - y_j \|^2$$

where y_i is a p-dimensional embedding of a cell i. In other words, if two cells are transcriptionally similiar, we attempt to place them nearby. It can be shown that minimisation of equation subject to $y^T y = 1$ is equivalent to solving the eigenvalue problem

$$\mathrm{Ly} = \lambda y$$

where $L = D - W$, D is the degree matrix and is the diagonal matrix made up of the row sums of W.

A robust choice of W is that $W_{\mathrm{ij}} = 1$ if i is among the k nearest neighbourhood of j, $W_{\mathrm{ij}} = 0$ otherwise.

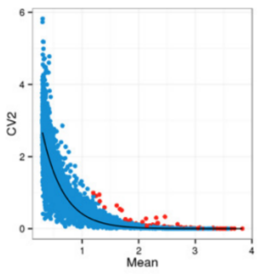
$k = \ln(n)$ is a typical choice. To ensure the $\mathrm{eigenvalues}$ are real and eigenvectors are orthogonal

$$\widetilde{W_{\mathrm{ij}}} = \frac{1}{2}(W_{\mathrm{ij}} + W_{\mathrm{ji}})$$

Once this low dimensional embedding is constructed, principal curves are used to trace through the centre of the manifold and assign a pseudotime to each cell.

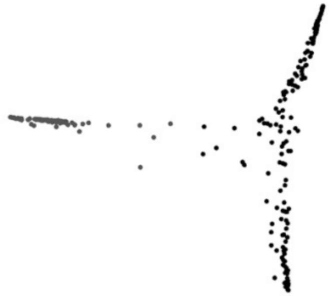
Once the principal curve has been fit, the orthogonal projection of each cell onto it may be found.Subsequently, the arc-length from the beginning of the curve to the projected point estimates the pseudotime of the cell. All pseudotimes are then rescaled to be in [0,1] by

$$\tilde{t} = \frac{t - \min(t)}{\max(t) - \min(t)}$$

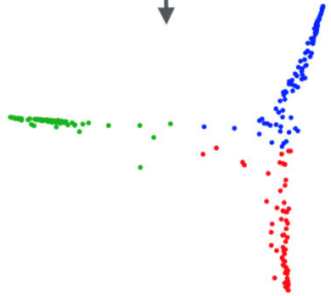


1: Select high variance genes

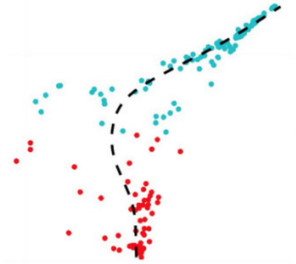
2: Build nearest neighbour correlation graph



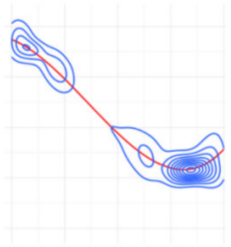
3: Low dimensional embedding using laplacian eigenmaps



4: Cluster embedding and any branches



5: Fit pseudotime using principal curves



6: Downstream analysis of genes in pseudotime

2.5 Wanderlust

Wanderlust receives as input a list of N points in D dimensions. Each point is a cell. The algorithm assume that the cells lies in a 1D developmental trajectory. In addition , Wanderlust receives an early initiator cell s that serves as a starting point.

2.5.1 Initialization step:

1. Wanderlust flags a set of cells as waypoint, the selection is random.
2. The data is then converted to a KNN graph, where node is cell coords, weight is the distance between two nodes.
3. An ensembel of l-out-of-KNN graphs (l-k-NNG) is generated by randomly picking l neighbors out of KNN for each cells.

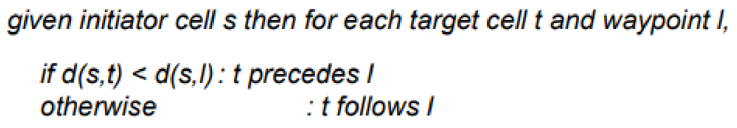
2.5.2 Trajectory calculation step:

1. First we define the shortest path distance for pair

$$\mathrm{spd}(s,t) = \min_p \sum_{e \in p} w(e)$$

   where p is the path between s and t. spd is calculated by dijkstra

2. The trajectory score is initialized to the spd with initiator s.
3. Calculate the spd with each way points,which serves as a landmarks
4. Then we reorientate the distance by



given initiator cell s then for each target cell t and waypoint l,

```
    if d(s,t) < d(s,l) : t precedes l
    otherwise          : t follows l
```

5. Since the nearby waypoints is more accurate than a distant one, we gives a weight for the waypoints and cell

$$w_{l,t} = \frac{d(l,t)^2}{\sum_m d(l,m)}$$

6. The trajectory score is updated by

$$\mathrm{traj}_t = \sum_l \frac{d(l,t)}{n_l} w_{l,t}$$

Input: data set of cells $X = \{x_1, \ldots, x_n\}$, initiator cell s, number of waypoints nl, distance function *dist*, ensemble parameters: size of ensemble ng, number of nearest neighbors k, subset size l
Output: trajectory score $S = \{s_1, \ldots, s_n\}$ for each cell x_i
Initialization:

```
    pick from $X$ $nl$ cells uniformly at random to serve as waypoints; set $s$ as first waypoint $\rightarrow$
$\{l_1 = s, \ldots, l_{nl}\}$
    calculate $k$-nearest-neighbor graph of $X \rightarrow G$
    randomly generate $ng$ $l$-out-of-$k$-nearest-neighbor graphs $\rightarrow G_{1, \ldots,} G_{ng}$
```

Trajectory calculation:

```
    for each $l$-out-of-$k$-nearest-neighbor graph
        calculate shortest-path distance from each waypoint $l_j$ to each point $x_i \rightarrow D = d_{ij}$
        set $w_{ij}$ to $|d_{ij}|^2 / \Sigma_k |d_{ik}|^2$


        realign: calculate realigned distance T = $t_{ij}$. for each waypoint $l_j$
            $t_{ij} = d_{ij}$ if $d_{1i} > d_{1j}$, $-d_{ij}$ otherwise
            $t_{ij} = t_{ij} + d_{1j}$
        set $traj_{1,i}$ to $\Sigma_j t_{ij}/nl * w_{ij}$
        repeat until convergence
            realign $t_{ij}$ using $traj_{(iter-1),i}$
                set $traj_{iter,i}$ to $\Sigma_j t_{ij}/n_l * w_{ij}$
            set the graph's trajectory to $traj$ of the last iteration
    return average over all graph trajectories in ensemble $\rightarrow$ S
```