t-SNE and Self-Organizing Maps

Data Visualization, Dimension reduction and Classification

Adrija Bhar
Srijani Das

Indian Statistical Institute

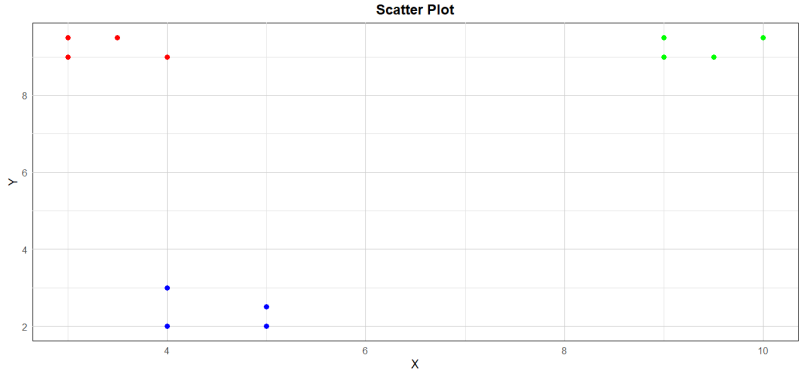
November 6, 2024

Introduction

- Visualization of high dimensional data is an important problem in many different domains.
- The aim of dimensionality reduction is to preserve as much of the significance structure of the high dimensional data as possible for the low dimensional map.
- t-SNE is a technique that visualizes high-dimensional data by giving each data point a location in a two or three-dimensional map.
- It was developed by Lawrens van der Maaten and Geoffrey Hinton in 2008.
- The technique is a variation of stochastic neighbor embedding.

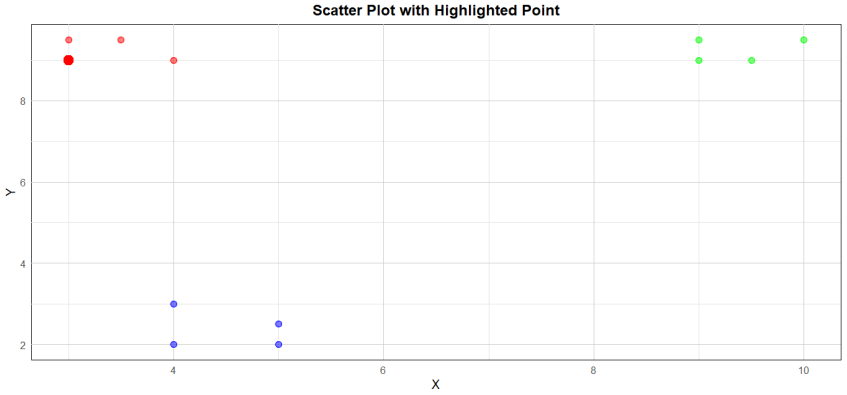
Stochastic Neighbour Embedding to t-Stachastic Neighbour embedding: Symmetric SNE

- Lets discuss the initial steps of the idea with a simple two -dimensional example.



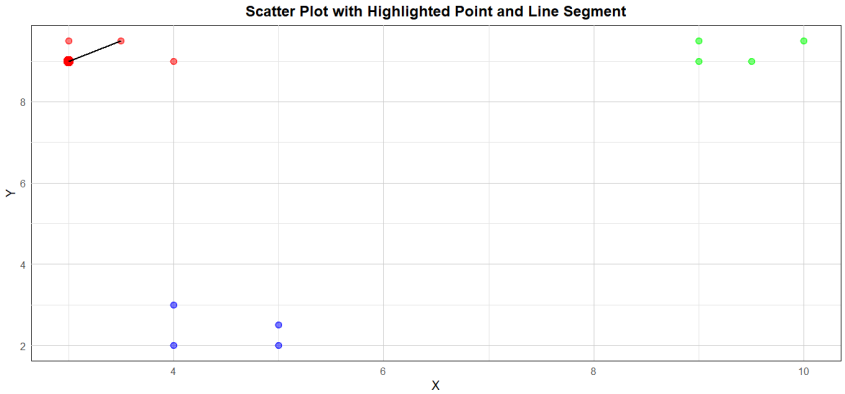
- The similarity of a datapoint $\boldsymbol{x_i}$ with $\boldsymbol{x_j}$ is defined as the conditional probability $p_{j|i}$ that $\boldsymbol{x_i}$ would pick $\boldsymbol{x_j}$ as its neighbour.

Stochastic Neighbour Embedding to t-Stachastic Neighbour embedding: Symmetric SNE



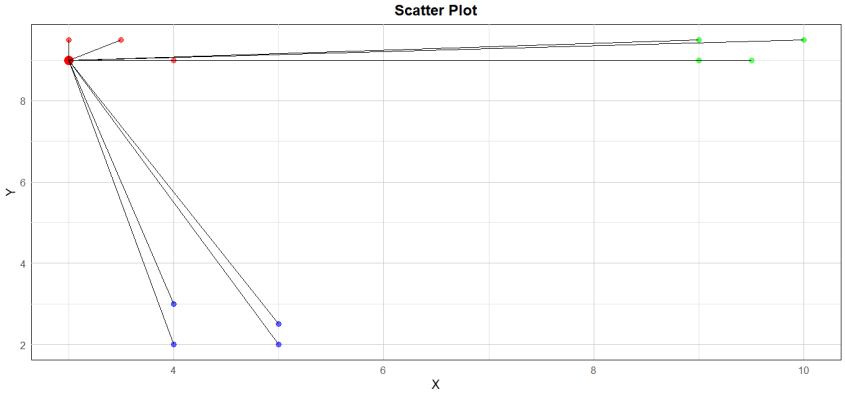
- Consider the highlighted point $\boldsymbol{x_i}$

Stochastic Neighbour Embedding to t-Stachastic Neighbour embedding: Symmetric SNE



- We find the distance(euclidean distance) between x_i and another point, lets call it x_j.

Stochastic Neighbour Embedding to t-Stachastic Neighbour embedding: Symmetric SNE



- We measure the distance(in this case the euclidean distance) from the selected point to all the other points as shown in the plot.

Stochastic Neighbour Embedding to t-Stachastic Neighbour embedding: Symmetric SNE

- Now the similarity measure that is $p_{j|i}$ must be proportional to probability density under a Gaussian centered at $\boldsymbol{x_i}$
- The conditional probability $p_{j|i}$ is given by

$$p_{j|i} = \frac{exp(-\|\boldsymbol{x_i} - \boldsymbol{x_j}\|^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-\|\boldsymbol{x_i} - \boldsymbol{x_j}\|^2/2\sigma_i^2)}$$

  where σ_i^2 is the variance of the Gaussian that is centered on data-point $\boldsymbol{x_i}$.
- We calculate $p_{j|i}$ for all i=1,2,...,N and j=1,2,...,N
- We set the value of $p_{i|i}$ to zero for all i.

Perplexity

- The remaining parameter to be selected is the variance σ_i^2 of the Gaussian that is centered over each high-dimensional data point, $\boldsymbol{x_i}$.
- SNE performs a binary search for the value of σ_i that produces a P_i with a fixed perplexity that is specified by the user.
- The perplexity is defined as

$$Perp(P_i) = 2^{H(P_i)}$$

, where $H(P_i)$ is the Shannon entropy of P_i given as.

$$H(Pi) = \sum_j p_{j|i} \log_2 p_{j|i}$$

- This was a toy data, to explain the initial step. In case of a high dimensional data we shall follow these steps exactly.
- Our goal is to map the high dimensional **x** values to 2 dimensional **y** values.
- The similarity between $\boldsymbol{y_i}$ and $\boldsymbol{y_j}$ is given by,

$$q_{j|i} = \frac{exp(-\|\boldsymbol{y_i} - \boldsymbol{y_j}\|^2)}{\sum_{k \neq i} exp(-\|\boldsymbol{y_i} - \boldsymbol{y_j}\|^2)}$$

- We set the value of $q_{i|i}$ to zero.
- Intuitively, if the mapping from **x** to **y** is efficient, then $p_{j|i}$ and $q_{j|i}$ should be fairly equal for all i and j.
- A natural measure of the faithfulness with which $q_{j|i}$ models $p_{j|i}$ is the Kullback Leibler divergence.

- SNE minimizes the cost function given by,

$$C = \sum_i KL(P_i \| Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

  in which P_i represents the conditional probability distribution over all other data points given data point $\boldsymbol{x_i}$, and Q_i represents the conditional probability distribution over all other map points given map point $\boldsymbol{y_i}$.
- The minimization of the cost function is performed using a gradient descent method. The gradient has a surprisingly simple form

$$\frac{\delta C}{\delta \boldsymbol{y_i}} = 2 \sum_j (p_{i|j} - q_{i|j} + p_{j|i} - q_{j|i})(\boldsymbol{y_i} - \boldsymbol{y_j})$$

- Intuitively it is natural to think that similarity between $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ must be equal to the similarity between $\boldsymbol{x_j}$ and $\boldsymbol{x_i}$.Therefore $p_{j|i}$ should be equal to $p_{i|j}$, but that's not the case here.
- Therefore, we might consider p_{ij} to be the similarity measure between $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ where,
$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2}$$
- If $\boldsymbol{x_i}$ turns out to be an outlier then the values of p_{ij} are extremely small for all j, so the location of its low-dimensional map point $\boldsymbol{y_i}$ has very little effect on the cost function.
- We circumvent this problem by defining the joint probabilities p_{ij} in the high-dimensional space to be
$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$
provided there are N data points.
- This ensures that $\sum_j p_{ij} > \frac{1}{2N}$ for all data points $\boldsymbol{x_i}$.

Crowding Problem

- Consider a sphere in 10 dimensional space. The volume of the sphere centered on data-point i scales as r^{10}, where r is the sphere's radius.
- When we try to model the distances from the *i*th data point to the other data points in the two-dimensional map which are approximately uniformly distributed around the ith point in the 10-dimensional space, we get the following "crowding problem".
- The area of the two-dimensional map that is available to accommodate moderately distant data points will not be nearly large enough compared with the area available to accommodate nearby data points.

Crowding Problem

- Hence, if we want to model the small distances accurately in the map, most of the points at a moderate distance from data point ith will have to be placed much too far away in the two-dimensional map.
- In SNE, the spring connecting datapoint i to each of these too-distant map points will thus exert a very small attractive force. Although these attractive forces are very small, the very large number of such forces crushes together the points in the center of the map, which prevents gaps from forming between the natural clusters.

- To solve the "crowding problem" we can use a probability distribution with much heavier tails than a Gaussian to convert distances into probabilities in the low-dimensional map.
- In t-SNE, we employ a Student t-distribution with one degree of freedom (the same as a standard Cauchy distribution) as the heavy-tailed distribution in the low-dimensional map.
- Using this distribution, the joint probabilities q_{ij} are defined as

$$q_{ij} = \frac{(1 + \|\boldsymbol{y_i} - \boldsymbol{y_j}\|^2)^{-1}}{\sum_{k \neq l}(1 + \|\boldsymbol{y_k} - \boldsymbol{y_l}\|^2)^{-1}} - - - eq^n(1)$$

- The cost function C is given by

$$C = KL(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

  in which P represents the joint probability distribution in the high dimensional space, and Q represents the Student's t based joint distribution in the low dimensional space.

- The gradient of the Kullback-Leibler divergence between P and the Student-t based joint probability distribution Q is given by

$$\frac{\delta C}{\delta \boldsymbol{y_i}} = 4 \sum_j (p_{ij} - q_{ij})(\boldsymbol{y_i} - \boldsymbol{y_j})(1 + \|\boldsymbol{y_i} - \boldsymbol{y_j}\|^2)^{-1} - - - eq^n(2)$$

t-SNE Algorithm at a glance

- Data set X=($\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}$)
- Compute $\sigma_i s$ for a pre-chosen perplexity value
- Compute pairwise affinities $p_{j|i}$ and set $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$
- Sample initial solution $Y^{(0)} = (\boldsymbol{y_1}, \boldsymbol{y_2}, ..., \boldsymbol{y_n})$, from $N_2(\mathbf{0}, 10^{-4} I)$
- Compute low dimensional affinities q_{ij} from $eq^n(1)$
- Compute gradient $\frac{\delta C}{\delta Y}$ from $eq^n(2)$
- Set $Y^{(t)} = Y^{(t-1)} + \eta \frac{\delta C}{\delta Y} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)})$
- Return to step 5 and repeat the consecutive steps until convergence

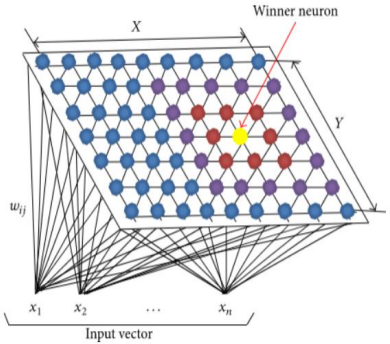
Weaknesses

- It is unclear how t-SNE performs on general dimensionality reduction tasks.
- The relatively local nature of t-SNE makes it sensitive to the curse of the intrinsic dimensionality of the data.
- t-SNE is not guaranteed to converge to a global optimum of its cost function.

Origins of Self-Organizing Maps (SOM)

- The Self-Organising Map (SOM) is an unsupervised machine learning algorithm introduced by Teuvo Kohonen in the 1980s.
- SOMs are inspired by the brain's structure, specifically the organization of sensory neurons in the cerebral cortex. Brain maps organize information spatially, with nearby neurons responding to similar sensory stimuli.
- This inspired the creation of a network that can map inputs in a structured, spatially meaningful way.

SOM Algorithm



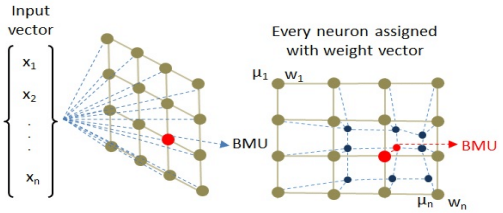
- The neural network of the Self-Organising Map has one input layer and one output layer.
- The second layer typically consists of a two-dimensional lattice of m x n nodes(neurons).

SOM Algorithm

- **Neural Structure:** Each node i in a SOM is represented by a weight vector $\boldsymbol{w}_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$, with the same dimension as input $\boldsymbol{x}$. This weight vector determines what data each node represents.
- Before recursive processing the $\boldsymbol{w}_i$ must be initialized. We select random numbers for the components of the $\boldsymbol{w}_i$.
- Let us define the initials values of w_{ij} by the following:

$$\boldsymbol{w_i(0)} = [w_{i1}(0), w_{i2}(0), \ldots, w_{in}(0)]^T$$

SOM Algorithm



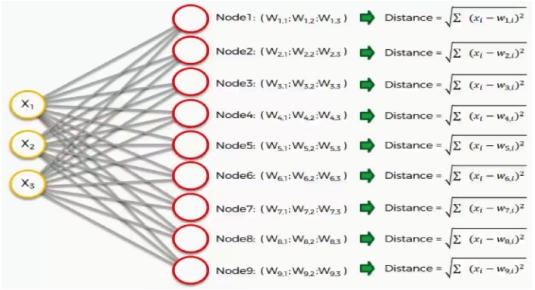
- In competitive learning, nodes in the output layer compete among themselves to be activated.
- In SOM, the competitive process is to search for the most similar node with the input pattern; the winner is called Best Matching Unit (BMU).

SOM Algorithm

- At the beginning we randomly choose a data point from the input space, let us call it $\boldsymbol{x}(0)$ corresponding to the first iteration.
- we calculate the distance of $\boldsymbol{x}(0)$ from all the nodes and define the *best matching* node $\boldsymbol{c}$ corresponding to $\boldsymbol{x}(0)$ as,

$$c = \arg\min_i \|\boldsymbol{x}(0) - \boldsymbol{w_i(0)}\|$$

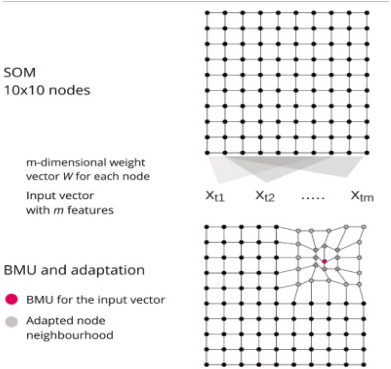
SOM Algorithm



- Similarity as the winning criterion can be measured in several ways. The most common measure employed is Euclidean distance.
- The node with the shortest distance from the input signal becomes the BMU.

SOM Algorithm

- In SOM, learning is not only for the winner but also for the nodes in physical proximity to it on the map.
- The BMU shares the privilege with its neighbours to learn together.



SOM Algorithm

- Consider the neighbourhood N_c around the node c. Here N_c consists of all nodes upto a certain radius in the grid from node c.
- The $\boldsymbol{w_i}$'s are updated in the following way:

$$\boldsymbol{w_i}(1) = \boldsymbol{w_i}(0) + h_{ci}(0)(\boldsymbol{x}(0) - \boldsymbol{w_i}(0))$$

- We repeat this process untill convergence where at the tth iteration the weights are updates as:

$$\boldsymbol{w_i}(t+1) = \boldsymbol{w_i}(t) + h_{ci}(t)(\boldsymbol{x}(t) - \boldsymbol{w_i}(t))$$

where $t = 0, 1, 2, \ldots$.

Features of the Neighborhood Function and Learning Rate

- The function $h_{ci}(t)$ acts as the *neighbourhood function*, a smoothing kernel defined over the grid points.
- For convergence we need:

$$h_{ci}(t) \to 0 \quad \text{as} \quad t \to \infty$$

- One simple choice of $h_{ci}(t)$ is:

$$h_{ci}(t) = \begin{cases} \alpha(t), & \text{if} \quad i \in N_c (= N_c(t)) \\ 0, & \text{if} \quad i \notin N_c \end{cases}$$

- Here $\alpha(t)$ is the *learning rate* factor and $0 < \alpha(t) < 1$.
- Both $\alpha(t)$ and the radius of $N_c(t)$ are monotonically decreasing with the number of iterations.

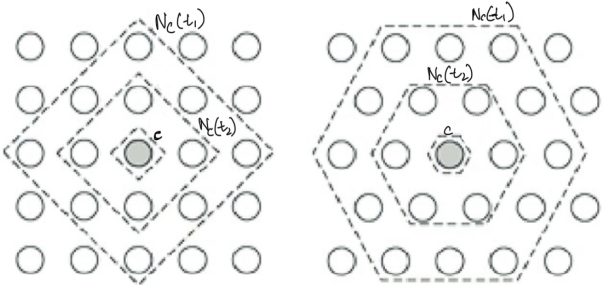
Another Exmaple of neighborhood function

- Another widely used neighborhood function is:

$$h_{ci}(t) = \alpha(t).\exp\left(\frac{-\|\boldsymbol{r_c} - \boldsymbol{r_i}\|^2}{2\sigma^2(t)}\right)$$

  where $\alpha(t)$ is another scalar valued learning rate factor and $\sigma(t)$ defines the width of the kernel. Both $\alpha(t)$ and $\sigma(t)$ are monotonically decreasing with the number of iterations.
- $\boldsymbol{r_c}$ and $\boldsymbol{r_i}$ are the position vectors of the cth and ith nodes respectively.

Rectangular vs. Hexagonal Grids in SOM



- **Rectangular SOM:** Neurons arranged in a grid with four neighbors (top, bottom, left, right). Easier to implement but may limit topology.
- **Hexagonal SOM:** Each neuron has six neighbors, providing smoother data representation and often better topology preservation.

Word of Caution

- If the neighbourhood range is too small, the trained model will suffer from overfitting, and there is a risk of having some dead neurons that never have an opportunity to change.

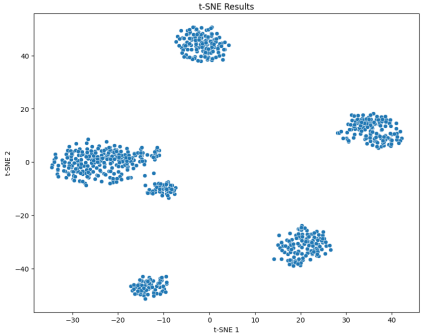
SOM as a Feedback Network

- Unlike feedforward networks that process information only in one direction (from input to output).
- SOMs rely on feedback, constantly adjusting neurons' weights until the network stabilizes.
- This iterative feedback allows the network to self-organize based on the data structure.

Data set 1 information:
gene expression cancer RNA-Seq

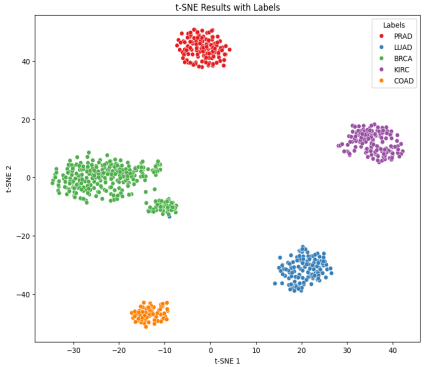
- We have data of a part of the RNA-Seq (HiSeq) PANCAN data set. It is a random extraction of gene expressions of patients having different types of tumors: BRCA, KIRC, COAD, LUAD, and PRAD.
- Samples are stored row-wise. There are a total of 801 samples.
- We have 20531 variables for each sample. They are RNA-Sequence gene expression levels measured by the Illumina HiSeq platform.
- Corresponding to each sample we have a 20531 dimensional feature vector. Considering the labels of each sample to be unknown, we shall map the high-dimensional data to 2-dimensional feature space for visualization purpose

Implementing t-SNE on Data Set-1



- We see that the mapping of the high-dimensional data to the 2-D space has given rise to 5 clusters. We guess that each cluster corresponds to samples belonging to a particular tumor group.

- Now we have re-performed t-SNE estimation but we have colour-coded the mapped data points for each sample according to their tumor class.

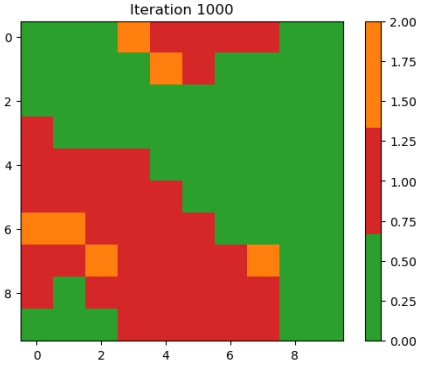


- Therefore, visually it seems that that t-SNE has mapped the data successfully to a lower dimension space.

Data set 2 information:
Banknote Authentication

- Data were extracted from images that were taken for the evaluation of an authentication procedure for banknotes.
- For digitization, an industrial camera usually used for print inspection was used.
- The final images have 400x 400 pixels. Due to the object lens and distance to the investigated object gray-scale pictures with a resolution of about 660 dpi were gained. Wavelet Transform tool were used to extract features from images.
- Samples are stored row-wise. There are a total of 1372 samples.
- We have 4 variables for each sample.

Implementing SOM on Data Set-2



- We see that at the iteration 1000 SOM gives more or less two main clusters. Therefore, visually it seem that SOM has mapped the data successfully to a lower dimension space.

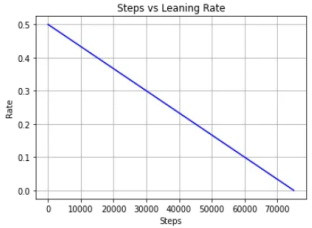
Implementing SOM on Data Set-2

```
# test data

# using the trained som, search the winning node of corresponding to the test data
# get the label of the winning node

data = minmax_scaler(test_x) # normalisation

winner_labels = []

for t in range(data.shape[0]):
 winner = winning_neuron(data, t, som, num_rows, num_cols)
 row = winner[0]
 col = winner[1]
 predicted = label_map[row][col]
 winner_labels.append(predicted)

print("Accuracy: ",accuracy_score(test_y, np.array(winner_labels)))
```

Python

```
Accuracy:  0.9636363636363636
```

Plot of leraning rate vs iteration



- Here the following learning rate function is used:

$$\alpha(t) = \left(1 - \frac{t}{\max(t)}\right) * \alpha(0) \quad , t = 0, 1, 2, \ldots$$

,which seems to be a reasonable choice and where $\alpha(0) = 0.5$ which is the initially chosen learning rate. And $\max(t) = 75000$ which is the total number of iteration.

References

- Teuvo Kohonen,Self-Organizing Maps, Springer Series in Information Sciences, Springer Berlin, Heidelberg
- Ken Moriwaki, Understanding Self-Organising Map Neural Network with Python Code, towardsdatascience.com
- Alexander N. Gorban, Balzs Kgl, Principal Manifolds for Data Visualization and Dimension Reduction, Springer Publishing Company
- Laurens van der Maaten, Geoffrey Hinton, Visualizing Data using t-SNE, Journal of Machine Learning Research 9 (2008) 2579-2605

Thank You