Stockholms universitet

A Survey of Stochastic Neighbor Embedding for Dimension Reduction and Data Visualization

Tobias Wängberg

Masteruppsats i matematisk statistik
Master Thesis in Mathematical Statistics

Masteruppsats 2020:12
Matematisk statistik
September 2020

www.math.su.se

Matematisk statistik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm

Matematiska institutionen

Mathematical Statistics
Stockholm University
Master Thesis **2020:12**
http://www.math.su.se

A Survey of Stochastic Neighbor Embedding for Dimension Reduction and Data Visualization

Tobias Wängberg*

September 2020

Abstract

During recent years machine learning and its applications to data science have opened many new opportunities in science, owing to increased computing power and higher availability of data, which has transformed fields such as biology, economics and social science. Along the many great opportunities that data driven science encompasses, new challenges emerge in how to correctly interpret and extract relevant information from complex data. One such challenge is that data today is often high dimensional, where, for example, a cell may be characterised by expression of hundreds of genes or a high resolution image by hundreds of pixels. This makes interpretation of the data difficult, and the need to reduce the dimension without losing critical information becomes an important task. In particular, the t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm has become a popular tool for visualising high dimensional data during recent years due to its capability of creating compelling 2D visualisations. Its inner mathematical workings are however poorly understood, and it can therefore be difficult to interpret the result of the dimension reduction using this algorithm. To this end, this thesis will explore the statistical properties of t-SNE, highlight its possible artifacts and benchmark it with test cases to illustrate its strengths and weaknesses.

Keywords: unsupervised learning, data visualisation, dimension reduction, t-SNE, evaluation, benchmarking

*Postal address: Mathematical Statistics, Stockholm University, SE-106 91, Sweden. E-mail: tobias@math.su.se. Supervisor: Chun-Biu Li.

Contents

Notations

Symbol	Explanation
$\mathbb{R}, \mathbb{N}$	set of real and natural numbers
$i, j \in \mathbb{N}$	matrix indices, i stands for row and j for column
$\mathbf{X}$	high dimensional $D \times N$ data matrix
$\mathbf{Y}$	low dimensional $d \times N$ data matrix
$\boldsymbol{x}, \boldsymbol{y}$	vector
$\mathcal{X}, \mathcal{Y}$	Data set
$\mathbf{A}^T$	Transpose of the matrix $\mathbf{A}$
D	Dimension of $\mathcal{X}$
d	Dimension of $\mathcal{Y}$
N	Number of data points
$H(\cdot)$	Shannon entropy
$\mathrm{KL}(\cdot\|\|\cdot)$	Kullback-Leibler divergence
X, Y	Random variables
P, Q	Probability mass functions
$\sim$	distributed as
S	similarity measure
Δ, δ	distance
$\mathbb{E}$	Expected value operator
$\|\| \cdot \|\|$	Euclidean norm
$\propto$	Proportional to
$\|A\|$	Cardinality of set A
$R(K)$	Measure of neighborhood preservation of size K
$\overline{R}$	Average neighborhood preservation

1 Introduction

Machine learning has seen a rise in popularity in science during recent years due to the rapid increase in computing power. As modern data sets are often increasingly high dimensional, the need for reducing the dimension of the data becomes important since high dimensional lead to problems such as the curse of dimensionality and the difficulty in recognizing patterns in the data by visualisation. For example diagnosis of breast cancer tumours using processed pictures of cell nuclei is based on 30-dimensional data points [24]. The MNIST data set [12] consists of pixel values of pictures of handwritten digits with $28 \times 28 = 784$ pixel resolution. The aim is then, given a high dimensional data set $\mathcal{X} = \{x_1, \ldots, x_N\}$, to reduce $\mathcal{X}$ to a low dimensional set $\mathcal{Y} = \{y_1, \ldots, y_N\}$ such that the important patterns from $\mathcal{X}$ are preserved in $\mathcal{Y}$.

Methods to reduce dimensionality have been suggested as early as 1901 by the famous statistician Karl Pearson [21]. This method works applying a rotation followed by an orthogonal projection down to the desired dimension, or in other words, the algorithm projects the data to a linear hyperplane minimizing the orthogonal distance between the plane and the data points. Hence, this method is restricted to linear data sets, which makes it limited in its applicability as data sets are often have a non-linear structure. Since then, many new methods have been developed that are able to deal with non-linear data, [26, 14] provides a review over some commonly used methods. This thesis will explore a particular variant of Stochastic Neighbor Embedding (SNE) [8], called t-Distributed Stochastic Neighbor Embedding (t-SNE) used for dimensionality reduction [18].

The t-SNE algorithm has been especially popular in cell biology for visualising spatial gene expressions in cells [10, 11, 7, 19], due to its capability of creating two-dimensional visualisations with clearly separated clusters corresponding to various cell functions. The algorithm does however possess artifacts, and its inner mathematical workings are not well understood. The aim of this thesis is therefore to benchmark this algorithm with some common test cases, evaluate its performance and identify possible artifacts, strengths and weaknesses that are important to be aware of when using the algorithm in a scientific setting.

The outline of the thesis is as follows. In Section 2 we provide some background theory. Then, in Section 3 and Section 4 we present the theory behind Principal Component Analysis (PCA) and Stochastic Neighbor Embedding is provided. We proceed by reviewing some evaluation methods for dimensionality reduction in Section 5 and the t-SNE algorithm is then benchmarked with test cases in Section 6. Finally, outlooks and conclusions are contained in Section 7.

2 Preliminaries

In this section some preliminary theory is presented, including some fundamental concepts and results from information theory as well as a brief introduction to the field of machine learning and dimensionality reduction.

2.1 Information Theory

In this section we will provide some of the fundamental concepts and results from Information Theory, that will be used in the remaining part of the thesis. Only a brief introduction of results and definitions will be given here, for a more extensive treatment we refer to [3].

First we establish some notation. Symbols P and Q will denote discrete probability mass functions defined on the indexed space of outcomes $\Omega = \{1, \ldots, N\}$. That is, if a random variable X is distributed according to P, then the probability $\text{Prob}(X = i) = P(i) = p_i$. This is written as $X \sim P$.

We begin by defining Shannon Entropy.

Definition 1 (Shannon Entropy)**.** Let X be a discrete random variable distributed according to the probability mass function P, taking value i with probability $p_i > 0$ for $i = 1, \ldots, N$. The *Shannon Entropy* $H(X)$ of X is defined as

$$H(X) = \mathbb{E}\left[\log \frac{1}{P(X)}\right] = \sum_{i=1}^{N} p_i \log \frac{1}{p_i}.$$

For brevity we will often write $H(P)$ to denote the Shannon Entropy of a random variable distributed according to P. The base of the logarithm is not specified in the definition, since changing the logarithm base will only amount to a change in unit of the entropy:

$$H_a(P) = \sum_{i=1}^{n} p_i \log_a \frac{1}{p_i} = \frac{1}{\log_b a} \sum_{i=1}^{n} p_i \log_b \frac{1}{p_i} = \frac{1}{\log_b a} H_b(P).$$

Usually, the base of the logarithm is 2, and it is then said that the entropy is measured in bits. Another common unit is *nats*, which is the unit referred to when the entropy is computed with the natural logarithm.

Some immediate properties of the entropy is that $H(X) \geq 0$, and that $H(X, Y) = H(X) + H(Y)$ for X and Y independent random variables. The entropy is sometimes referred to as information, as these are properties that a measure of information would intuitively satisfy. Furthermore, it can be shown that the logarithm function is the only function satisfying the axioms for information, which is proved in the original paper introducing Shannon Entropy [23].

The entropy can be interpreted as a measure of the uncertainty of a random variable. The entropy is maximized when all outcomes are equally likely, and the maximum value is equal to $\log N$ where N denotes the number of outcomes. Furthermore it is minimized and equal to 0 when all probability mass is put on a single outcome, that is when there is no uncertainty. Example 1 shows this in the binary outcome case. The general result is shown at the end of the section in Proposition 2. The entropy can also be shown to measure the minimal expected number of yes/no questions to determine the outcome of a random variable, see [3] for a proof. Hence the Shannon Entropy is related to the predictability of a random variable.

Example 1. Consider a two-sided coin with probability of heads equal to $p_H = 1 - p_T$. The Shannon Entropy of the coin as a function of p_H is then given by

$$H(p_H) = p_H \log \frac{1}{p_H} + (1 - p_H) \log \frac{1}{1 - p_H}.$$

The function is illustrated in Fig. 1. The maximum entropy is in this case 1 bit, and is attained when both outcomes are equally likely. The entropy is equal to 0 bits for $p_H = 1$ or $p_H = 0$, i.e. when there is no uncertainty in the outcome of the coin flip. ♢

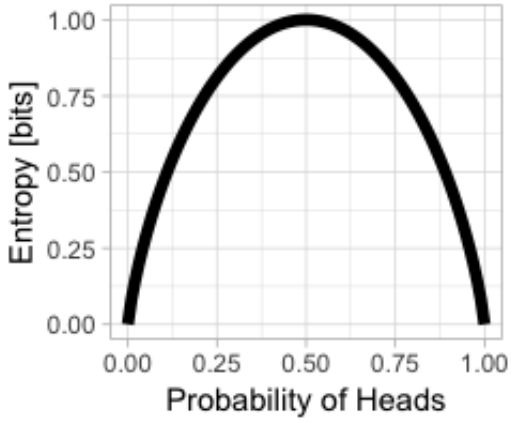


Figure 1: The Shannon Entropy as a function of probability of heads.

It can often be of interest to quantify the extent to which two probability distributions differ from each other. One such measure is the Kullback - Leibler divergence, which is defined in Definition 2 below.

Definition 2 (Kullback - Leibler divergence)**.** Let P and Q be probability mass functions with support on the set of outcomes $\Omega = \{1, \ldots, N\}$. The *Kullback-Leibler divergence* $\mathrm{KL}(P||Q)$ between P and Q is defined as

$$\mathrm{KL}(P||Q) = \sum_{i=1}^{N} p_i \log \frac{p_i}{q_i}.$$

The Kullback- Leibler divergence is a measure of the divergence between two probability distributions. It cannot however be interpreted as a distance in the usual sense, since $KL(P||Q)$ is not equal to $KL(Q||P)$ in general. It does however hold that $KL(P||Q) \geq 0$ and $KL(P||Q) = 0$ if and only if $p_i = q_i$ for all i, which Proposition 1 shows. The Kullback-Leibler divergence can therefore be thought of as a pseudo-distance between probability mass functions.

Proposition 1. Let $X \sim P$ and $Y \sim Q$, both with support on the set of outcomes $\Omega = \{1, \dots N\}$. Then

$$KL(P||Q) \geq 0$$

with equality if and only if $p_i = q_i$ for $i = 1, \dots, N$.

Proof. Since $\log x \leq x - 1$ for all $x > 0$ we have that

$$\begin{aligned} -KL(P||Q) &= \sum_{i=1}^{N} p_i \log \frac{q_i}{p_i} \\ &\leq \sum_{i=1}^{N} p_i \left(\frac{q_i}{p_i} - 1 \right) = 0 \end{aligned}$$

which proves that $KL(P||Q) \geq 0$. Since the logarithm function is concave, equality is attained exactly when $p_i = q_i$ for all $i = 1, \dots, N$. □

Proposition 2. Let P be a probability distribution defined on the set of outcomes $\Omega = \{1, \dots, N\}$. The maxiumum value of the Shannon Entropy is attained when $p_i = \frac{1}{N}$ for all i in Ω and is then equal to $\log N$.

Proof. Let U denote a uniformly distributed random variable with the same support as P. We have that

$$H(U) = \sum_{i=1}^{N} \frac{1}{N} \log \frac{1}{\frac{1}{N}} = \log N.$$

Next,

$$\begin{aligned} H(P) &= \sum_{i=1}^{N} p_i \log \frac{1}{p_i} \\ &= \log N - \log N + \sum_{i=1}^{N} p_i \log \frac{1}{p_i} \\ &= \log N - \sum_{i=1}^{N} p_i \log N + \sum_{i=1}^{N} p_i \log \frac{1}{p_i} \\ &= \log N + \sum_{i=1}^{N} p_i \log \frac{N}{p_i} \\ &= \log N - KL(P||U). \end{aligned}$$

By Proposition 1 it follows that the Kullback-Leibler divergence is maximised if and only if $P = U$ with probability 1. □

2.2 Machine Learning and Dimensionality Reduction

Machine Learning is a field on the border between computer science and mathematical statistics which aims to construct algorithms that make use of patterns in data to solve tasks of interest. Two major sub-fields of machine learning are *supervised learning* and *unsupervised learning*.

Supervised learning deals mainly with the problem of prediction. Put mathematically, given a *labeled* data set $\mathcal{X} = \{(x_1, y_1), \dots (x_n, y_n)\}$ where x_i is a data point, usually an element in $\mathbb{R}^D$, and the corresponding value y_i denotes the, most often discrete valued and 1-dimensional, attached label. For example, data point x_i may be a high dimensional vector containing the pixel values representing an image, and the label y_i indicating whether the image contains a cat or a dog, assuming that the total set of pictures are only pictures of cats and dogs. The aim is then, given a new data point x_{n+1} not present in $\mathcal{X}$, to predict its corresponding label y_{n+1}. More generally and expressed in statistical terms, the goal of supervised learning is given a finite sample $\mathcal{X}$ assumed generated from an unknown joint distribution $P(X, Y)$, to infer properties such as the conditional expectation $\mathbb{E}[Y \mid X = x_{n+1}]$, to be used for predicting the unknown label y_{n+1} from a data point x_{n+1}. A presentation of supervised learning and algorithms used for this purpose can be found in for example [5].

Unsupervised learning, on the other hand, is used in the setting where the data set $\mathcal{X} = \{x_1, \dots, x_N\}$ is *unlabeled*. If the data is assumed to be generated from a probability distribution $P(X)$, the objective is to infer properties about the underlying probability distribution. If it is a multivariate distribution this could be inferring associations between the variables, or other structures such as high density areas of the distribution. Suppose for example that the data $\mathcal{X}$ represents pixel values of pictures containing handwritten digits. Data points corresponding to pictures of a handwritten number seven, say, would be expected to be more similar to each other than pictures of for example the number eight. The pictures containing the number seven would therefore be expected to fall in to a separate *cluster* than data points representing different digits. The particular branch of unsupervised learning aimed at revealing such clusters is called *cluster analysis*. An introduction to such algorithms can be found in [5].

Another important branch of unsupervised learning is *dimensionality reduction*, which is the main focus of this thesis. A more extensive presentation of the field and various dimensionality reduction algorithms can be found in [14].

Images, for example, are usually represented in resolutions of 32×32 pixels, which makes a data point 968 dimensional. High dimensional data causes many problems, due to the difficulty of understanding the association between all the features as well as the structure of the data. It is often the case, however, that the data can be described by a fewer set of coordinates compared to the original observed data set. These coordinates are often called *latent variables* and are assumed to be obtained by transformation of the original variables. In general, given a high dimensional data set which we will denote as $\mathcal{X}$ containing data points of dimension D, the goal is to find a low dimensional representation set of

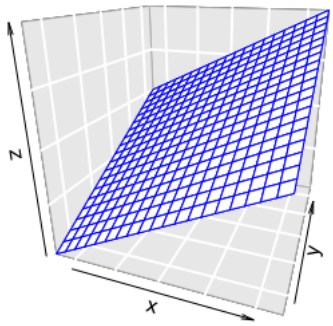


Figure 2: A two dimensional plane embedded in three dimensions. The figure illustrates a simple example of a linear manifold inscribed in a space of higher dimension.

points $\mathcal{Y}$ containing points of dimension d expressed by the latent variables. For example, Fig. 2 shows a two dimensional plane embedded in three dimensions, defined as

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_1 + x_2 \end{pmatrix}.$$

The *intrinsic dimensionality* of the data set is clearly two in this case, and can therefore be accurately represented by the two latent coordinates $(x_1, x_2)^T$, which can be obtained by a linear projection. When the data lies on a hyper plane like in the example above, the intrinsic structure of the data is said to be linear.

In practical applications, the underlying structure is however rarely linear. A common example of a two dimensional non-linear manifold embedded in three dimensions is the Swiss Roll, seen in Fig. 3. The manifold can be defined by

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_1 \cos 2x_1 \\ x_1 \sin 2x_1 \\ x_2 \end{pmatrix}.$$

When the data is non-linear it is more challenging to find the transformation that reduces the data to its intrinsic low dimensional structure. In this case the desired reduction is to unfold the Swiss Roll into a two dimensional rectangle.

In practical scenarios, neither the intrinsic dimensionality nor the intrinsic structure of the data is known before reduction. The data is also not expected to lie completely on the intrinsic structure due to the presence of random noise.

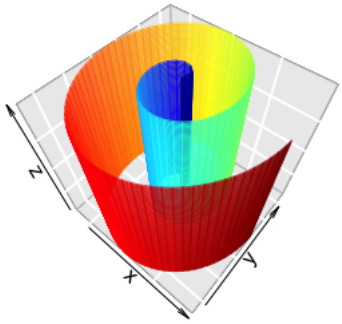


Figure 3: To the left is a non-linear two dimensional manifold embedded in three dimensions. This data set is called the Swiss Roll and is a common benchmarking data set.

For example if the low dimensional mapping is given by

$$\begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{iD} \end{pmatrix} \rightarrow \begin{pmatrix} y_{i1} + \epsilon_1 \\ y_{i2} + \epsilon_2 \\ \vdots \\ y_{id} + \epsilon_d \\ \epsilon_{d+1} \\ \vdots \\ \epsilon_D \end{pmatrix}.$$

where ϵ_i denotes noise variables, usually assumed to follow a Gaussian distribution, the noise does not count to the intrinsic structure and hence should be ignored.

If the reduction is made to a dimension below the intrinsic dimensionality of the data, then the low dimensional map will misrepresent the data by definition. There are cases when the intrinsic dimensionality is the same as the original dimension, and hence cannot be reduced. For example, consider a data set containing $D + 1$ pairwise equidistant points in D dimensional space. If the dimension were to be reduced to d, pairwise distances between only $d + 1$ of the $D + 1$ points can be preserved. Hence, dimensionality reduction would distort the true structure of the data if it reduces below the intrinsic dimensionality.

Next, a low dimensional data set $\mathcal{Y}$ of the intrinsic dimensionality d needs to be found from the observed D-dimensional data set $\mathcal{X}$. This can be done either by finding a mapping $\mathcal{M} : \mathbb{R}^D \rightarrow \mathbb{R}^d$ that reduces the data to dimension d, or by directly finding coordinates $\mathcal{Y}$ that minimizes some measure of the error of reduction.

It is also important to be able to quantitatively evaluate the quality of the dimensionality reduction, since it is difficult to, in an unbiased manner, qualitatively assess whether the low dimensional representation is a trustworthy

representation of the original data structure. If the method finds a explicit mapping $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$, then the error can be measured by the mean distance between coordinates in $\mathcal{X}$ and their representation in $\mathcal{Y}$. This is referred to as the mean square reconstruction error, and is given by

$$\frac{1}{N}\sum_{i=1}^{N}(x_i - \mathcal{M}^{-1}(\mathcal{M}(x_i)))^2$$

which quantifies the average distortion of the mapping $\mathcal{M}$. Here $\mathcal{M}^{-1}$ denotes the inverse of the mapping $\mathcal{M}$ If the method does not find an explicit mapping, then additional evaluation criteria needs to be developed.

3 Principal Component Analysis

Principal Component Analysis, PCA for short, is a well known method for dimensionality reduction and data visualisation. It was originally proposed in 1901 by one of the founders of modern statistics, Karl Pearson [21]. The algorithm linearly projects the centered data onto a hyperplane spanned by d orthogonal vectors such that the sample variance along each coordinate axis is maximized. This can equivalently be formulated as finding the hyperplane of dimension d that minimizes the orthogonal projection error onto that plane. It is therefore restricted to data that lies on a linear manifold, and may give misleading results if underlying data structure is non-linear. It turns out that the principal components are given by the eigenvectors of the sample covariance matrix and the corresponding eigenvalues of the same matrix is equal to the sample variance along each axis. Hence, the algorithm finds the eigenvalues and eigenvectors of the sample covariance matrix and projects the data onto a subspace whose basis vectors are the eigenvectors with the largest corresponding eigenvalues. In this section we also show that PCA can equivalently be formulated as minimizing the reconstruction error of the projection as well as the difference in inner product between the original and latent space.

3.1 Mathematical Derivation

We now proceed to describe the theory. To establish some notation, let $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$ denote the $D \times N$ data matrix of the observed data set $\mathcal{X}$ consisting of N observation vectors $\boldsymbol{x}_j = (x_{1j}, \ldots, x_{Dj})^T$ of dimension D. That is, the element x_{ij} at row i and column j in $\mathbf{X}$ is the j:th observation of coordinate i. We also denote by $\boldsymbol{x'}_i = (x_{i1}, \ldots, x_{iN})^T$ the i:th row vector of $\mathbf{X}$ containing the N observations of coordinate i. The sample covariance, which we denote as $\hat{\sigma}_{X_i X_{i'}}$, between coordinates i and i' in $\mathbf{X}$ is defined as

$$\hat{\sigma}_{X_i X_{i'}} = \frac{1}{N-1}\sum_{j=1}^{N}(x_{ij} - \overline{x}_i)(x_{ij'} - \overline{x}_{i'})$$

where

$$\overline{x}_i = \frac{1}{N}\sum_{j=1}^{N} x_{ij}$$

denotes the sample mean. From now on we assume $\overline{x}_i = 0$ for all $i = 1, \ldots, D$. This assumption is without loss of generality, since if $\overline{x}_i \neq 0$ for some i we can replace the corresponding row in $\mathbf{X}$ by its centered version. In matrix notation, we get that the sample covariance matrix is given by

$$\hat{\mathbb{C}}_{\mathbf{X}} = \frac{1}{N-1}\mathbf{X}\mathbf{X}^T.$$

Now, let $\mathbf{P} = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_d]$ be a $D \times d$ orthonormal matrix, that is the column vectors $\boldsymbol{p}_i$ of $\mathbf{P}$ are of unit length and orthogonal to each other. Hence, $\mathbf{P}^T\mathbf{P} = \mathbf{I}_d$ where $\mathbf{I}_d$ is the identity matrix of size $d \times d$, and $\mathbf{P}\mathbf{P}^T$ is the projection matrix projecting to the subspace spanned by the column vectors of $\mathbf{P}$. We have that

$$\boldsymbol{y} = \mathbf{P}^T\boldsymbol{x} = \begin{pmatrix} \boldsymbol{p}_1^T\boldsymbol{x} \\ \vdots \\ \boldsymbol{p}_d^T\boldsymbol{x} \end{pmatrix}$$

are the coordinates of the vector $\boldsymbol{x}$ expressed in the basis spanned by the column vectors of $\mathbf{P}$. Hence, the low dimensional $d \times N$ transformed data matrix $\mathbf{Y}$ is given by

$$\mathbf{Y} = \mathbf{P}^T\mathbf{X}$$

consisting of the coordinates in the low dimensional space. Furthermore, note that since $\overline{x}_i$ is equal to 0 for all i, $\overline{y}_i$ is necessarily also equal to 0, as

$$\overline{y}_i = \frac{1}{N}\sum_{j=1}^{N} y_{ij} = \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{p}_i^T\boldsymbol{x}_j = \frac{1}{N}\sum_{j=1}^{N}\sum_{k=1}^{D} p_{ik}x_{kj} = \sum_{k=1}^{D} p_{ik}\frac{1}{N}\sum_{j=1}^{N} x_{kj} = \sum_{k=1}^{D} p_{ik}\overline{x}_k = 0.$$

The projected sample variance along each coordinate axis in the subspace $\mathcal{Y}$ is therefore given by

$$\hat{\sigma}_{Y_iY_i} = \frac{1}{N-1}\sum_{j=1}^{N} y_{ij}^2 = \frac{1}{N-1}\sum_{j=1}^{N} (\boldsymbol{p}_i^T\boldsymbol{x}_j)^2.$$

We want to find the matrix $\mathbf{P}$ so that σ_{ii}^2 is maximised for all $i = 1, \ldots, d$. This is equivalent to finding $\mathbf{P}$ that maximises the *total variance*

$$\sum_{i=1}^{d} \hat{\sigma}_{Y_iY_i} = \frac{1}{N-1}\sum_{i=1}^{d}\sum_{j=1}^{N} (\boldsymbol{p}_i^T\boldsymbol{x}_j)^2.$$

If we denote by $\hat{\mathbb{C}}_{\mathbf{Y}}$ the sample covariance matrix of the low dimensional data matrix $\mathbf{Y} = \mathbf{P}^T\mathbf{X}$, we get that the sample covariance of $\mathbf{Y}$ is given by

$$\hat{\mathbb{C}}_{\mathbf{Y}} = \frac{1}{N-1}\mathbf{Y}\mathbf{Y}^T = \frac{1}{N-1}(\mathbf{P}^T\mathbf{X})(\mathbf{P}^T\mathbf{X})^T = \mathbf{P}^T\frac{\mathbf{X}\mathbf{X}^T}{N-1}\mathbf{P} = \mathbf{P}^T\hat{\mathbb{C}}_{\mathbf{X}}\mathbf{P}.$$

Denote by trace($\mathbf{M}$) the sum of the diagonal elements of the matrix $\mathbf{M}$. We want to find

$$\mathbf{P} = \arg\max_{\mathbf{P}} \text{trace}(\hat{\mathbb{C}}_{\mathbf{Y}}) = \arg\max_{\mathbf{P}} \text{trace}(\mathbf{P}^T \hat{\mathbb{C}}_{\mathbf{X}} \mathbf{P}).$$

First, since $\hat{\mathbb{C}}_{\mathbf{X}}$ is positive definite and symmetric, it follows that $\hat{\mathbb{C}}_{\mathbf{X}}$ is diagonalizable with non-negative eigenvalues. Furthermore, the number of non-negative eigenvalues is equal to the rank of $\mathbf{X}$, and are therefore related to the intrinsic dimensionality of $\mathcal{X}$. Hence, we can decompose $\hat{\mathbb{C}}_{\mathbf{X}}$ as

$$\hat{\mathbb{C}}_{\mathbf{X}} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$$

where $\mathbf{\Lambda}$ is a $D \times D$ diagonal matrix containing the eigenvalues λ_i on the diagonal, and $\mathbf{V}$ is a $D \times D$ orthonormal matrix whose columns are the eigenvectors $\boldsymbol{v}_i$ of $\hat{\mathbb{C}}_{\mathbf{X}}$. Therefore we can write

$$\text{trace}(\hat{\mathbb{C}}_{\mathbf{Y}}) = \text{trace}(\mathbf{P}^T \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T \mathbf{P}).$$

Writing this again as a sum, we get

$$\text{trace}(\hat{\mathbb{C}}_{\mathbf{Y}}) = \sum_{i=1}^{d} \sum_{j=1}^{D} (\boldsymbol{p}_i^T \boldsymbol{v}_j)^2 \lambda_j$$

Assume that the eigenvalues are ordered in descending order, so that

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D.$$

We get that

$$\sum_{j=1}^{D} (\boldsymbol{p}_1^T \boldsymbol{v}_j)^2 \lambda_j \leq \lambda_1 \sum_{j=1}^{D} (\boldsymbol{p}_1^T \boldsymbol{v}_j)^2 = \lambda_1 ||\mathbf{V}\mathbf{V}^T \boldsymbol{p}_1||_2^2 \leq ||\boldsymbol{p}_1||_2^2 \lambda_1 = \lambda_1.$$

The last inequality follows from that the length of an orthonormal projection is less than the length of the projected vector. Since $\boldsymbol{p_1}, \ldots, \boldsymbol{p}_d$ are required to be orthogonal to each other, we get that

$$\text{trace}(\hat{\mathbb{C}}_{\mathbf{Y}}) \leq \sum_{i=1}^{d} \lambda_i.$$

Equality is attained by setting $\boldsymbol{p_i} = \boldsymbol{v_i}$ for $i = 1, \ldots, d$.

Hence, the projection matrix $\mathbf{P}$ that maximises the sample variance is obtained by selecting the first d eigenvectors of $\hat{\mathbb{C}}_{\mathbf{X}}$, assuming the eigenvectors are sorted in descending order according to eigenvalue. The sample variance $\hat{\sigma}^2_{Y_i Y_i}$ is given by the eigenvalue λ_i, since

$$\hat{\sigma}^2_{Y_i Y_i} = \frac{1}{N-1} (\boldsymbol{v}_i^T \mathbf{X})(\boldsymbol{v}_i^T \mathbf{X})^T = \boldsymbol{v}_i^T \hat{\mathbb{C}}_{\mathbf{X}} \boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i^T \boldsymbol{v}_i = \lambda_i.$$

Moreover, the low dimensional covariance matrix is given by

$$\hat{\mathbb{C}}_{\mathbf{Y}} = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & \lambda_d \end{pmatrix}$$

which shows that the latent variables are uncorrelated.

3.1.1 Additional Properties

Assuming the data lies on a linear manifold, the PCA algorithm can immediately be used to determine the intrinsic dimensionaity d of the hyperplane. Since the eigenvalues will be zero for the remaining corresponding eigenvectors not spanning the latent hyperplane, the intrinsic dimensionality can be estimated by the number of non-zero eigenvalues of the sample covariance matrix. In practise, often no eigenvalues are found to be exactly zero. Then the coordinates with eigenvalues close to zero can be discarded. Alternatively, eigenvalues making up a large fraction of the total variance, 95% say, can be kept.

Since PCA finds a linear transformation $\mathbf{P}$, it is possible to quantify the error of the projection by the *mean square reconstruction error*

$$\text{MSRE}(\mathbf{P}) = \frac{1}{N}\sum_{j=1}^{N}(\boldsymbol{x}_j - \mathbf{P}\mathbf{P}^T\boldsymbol{x}_j)^2.$$

That is, the error is measured as the mean squared Euclidean distance between the observed data points and the projections onto the low dimensional hyper plane.

Furthermore, finding a transformation matrix $\mathbf{P}$, with orthogonal and unit length column vectors, that minimizes the reconstruction error, is equivalent to finding the transformation that maximizes variance. This follows from the following derivation.

$$\begin{aligned}
\text{MSRE}(\mathbf{P}) &\propto \text{trace}\left[(\mathbf{X} - \mathbf{P}\mathbf{P}^T\mathbf{X})^T(\mathbf{X} - \mathbf{P}\mathbf{P}^T\mathbf{X})\right] \\
&= \text{trace}\left[\mathbf{X}^T\mathbf{X} - \mathbf{X}^T\mathbf{P}\mathbf{P}^T\mathbf{X} - \mathbf{X}^T\mathbf{P}\mathbf{P}^T\mathbf{X} + \mathbf{X}^T\mathbf{P}\mathbf{P}^T\mathbf{P}\mathbf{P}^T\mathbf{X})\right] \\
&= \text{trace}\left[\mathbf{X}^T\mathbf{X} - \mathbf{X}^T\mathbf{P}\mathbf{P}^T\mathbf{X} - \mathbf{X}^T\mathbf{P}\mathbf{P}^T\mathbf{X} + \mathbf{X}^T\mathbf{P}\mathbf{P}^T\mathbf{X})\right] \\
&= \text{trace}\left[\mathbf{X}^T\mathbf{X}\right] - \text{trace}\left[\mathbf{X}^T\mathbf{P}\mathbf{P}^T\mathbf{X}\right] \\
&\propto -\text{trace}\left[\mathbf{X}^T\mathbf{P}\mathbf{P}^T\mathbf{X}\right] \\
&= -\text{trace}\left[\mathbf{P}^T\mathbf{X}\mathbf{X}^T\mathbf{P}\right] \\
&\propto -\text{trace}(\hat{\mathbb{C}}_{\mathbf{Y}}).
\end{aligned}$$

The equivalence follows from that

$$\arg\min_{\mathbf{P}} -\text{trace}(\hat{\mathbb{C}}_{\mathbf{Y}}) = \arg\max_{\mathbf{P}} \text{trace}(\hat{\mathbb{C}}_{\mathbf{Y}}).$$

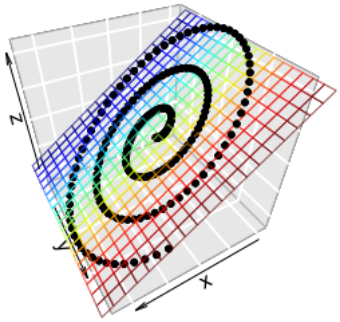


Figure 4: The figure shows a one-dimensional spiral lying on a two-dimensional plane embedded in three dimensions, thus illustrating a case when a one-dimensional non-linear manifold is inscribed in a two-dimensional linear plane.

Thereby PCA can also be thought of as finding the orthogonal projection $\mathbf{P}$ that minimizes the reconstruction error.

The PCA algorithm can also be combined with non-linear method. If the data has a hierarchical structure with a non-linear manifold lying on a low dimensional linear hyperplane, then PCA can be used before applying a non-linear dimension reduction method. A simple example is shown in Fig. 4.

3.1.2 Multidimensional Scaling

Another method for dimensionality reduction is Multidimensional Scaling (MDS) [4], that approaches the dimension reduction problem in a different way than PCA. Rather than finding a transformation $\mathbf{P}$ that minimizes some cost function, MDS instead directly finds low dimensional coordinates $\mathbf{Y}$ that minimises difference in some measure of pairwise similarity between the high dimensional and low dimensional spaces. There are many different versions that are based on this idea depending on what similarity measure is used. A well know MDS algorithm is classical MDS that minimizes the difference in centered inner products between the high dimensional and low dimensional spaces. The cost function minimized is given in Eq. (1) and is called the Strain cost function.

$$\text{Strain}(\mathbf{Y}) = \sum_{j=1}^{N}\sum_{j'=1}^{N}((\boldsymbol{x}_j - \overline{\boldsymbol{x}})^T(\boldsymbol{x}_{j'} - \overline{\boldsymbol{x}}) - (\boldsymbol{y}_j - \overline{\boldsymbol{y}})^T(\boldsymbol{y}_{j'} - \overline{\boldsymbol{y}}))^2 \qquad (1)$$

Here $\overline{\boldsymbol{x}} = (\overline{x}_1, \ldots, \overline{x}_D)^T$ denotes the sample mean vector. The cMDS method is often referred to as Principal Coordinate Analysis, PCoA, due to its equivalence to PCA when pairwise distances are measured with Euclidean distance which will be proved below. Instead of formulating the problem so as to directly find a transformation that maps the high dimensional data matrix $\mathbf{X}$ to a low dimensional data matrix $\mathbf{Y}$, cMDS directly finds coordinates $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N$ in $\mathbb{R}^d$

to minimize the cost. Due to the equivalence between PCA and cMDS, the mapping can however be retrieved.

The optimal set of coordinates $\mathbf{Y}$ can be found analytically, and is given by the eigenvectors and eigenvalues of the Gram matrix $\mathbf{X}^T\mathbf{X}$, given that the distances are taken to be Euclidean. To derive this, we start by rewriting Strain function in matrix notation using the trace operator,

$$\begin{aligned}
\text{Strain}(\mathbf{Y}) &= \text{trace}\left[(\mathbf{X}^T\mathbf{X} - \mathbf{Y}^T\mathbf{Y})^T(\mathbf{X}^T\mathbf{X} - \mathbf{Y}^T\mathbf{Y})\right] \\
&= \text{trace}\left[(\mathbf{X}^T\mathbf{X}\mathbf{X}^T\mathbf{X} - \mathbf{Y}^T\mathbf{Y}\mathbf{X}^T\mathbf{X} - \mathbf{X}^T\mathbf{X}\mathbf{Y}^T\mathbf{Y} + \mathbf{Y}^T\mathbf{Y}\mathbf{Y}^T\mathbf{Y}\right] \\
&= \text{trace}\left[\mathbf{X}^T\mathbf{X}\mathbf{X}^T\mathbf{X}\right] - \text{trace}\left[\mathbf{Y}^T\mathbf{Y}\mathbf{X}^T\mathbf{X}\right] - \text{trace}\left[\mathbf{X}^T\mathbf{X}\mathbf{Y}^T\mathbf{Y}\right] + \text{trace}\left[\mathbf{Y}^T\mathbf{Y}\mathbf{Y}^T\mathbf{Y}\right] \\
&= \text{trace}\left[\mathbf{X}^T\mathbf{X}\mathbf{X}^T\mathbf{X}\right] - 2\text{trace}\left[\mathbf{Y}^T\mathbf{Y}\mathbf{X}^T\mathbf{X}\right] + \text{trace}\left[\mathbf{Y}^T\mathbf{Y}\mathbf{Y}^T\mathbf{Y}\right].
\end{aligned}$$

Next, since $\mathbf{X}^T\mathbf{X}$ is positive definite and symmetric, it follows that it can be diagonalised by eigendecomposition as

$$\mathbf{U}_x\mathbf{\Lambda}_x\mathbf{U}_x^T$$

Furthermore we can decompose the Gram matrix $\mathbf{Y}^T\mathbf{Y}$ of the sought low dimensional data matrix $\mathbf{Y}$ in the same fashion,

$$\mathbf{Y}^T\mathbf{Y} = \mathbf{U}_y\mathbf{\Lambda}_y\mathbf{U}_y^T.$$

Let $\mathbf{1}_{m\times n}$ denote the $m \times n$ matrix obtained by discarding the first m columns of the $n \times n$ identity matrix. We can write

$$\mathbf{Y}^T\mathbf{Y} = \mathbf{U}_y\mathbf{\Lambda}_y\mathbf{U}_y^T = \mathbf{U}_y\mathbf{\Lambda}_y^{\frac{1}{2}}\mathbf{1}_{N\times d}\mathbf{1}_{d\times N}\mathbf{\Lambda}_y^{\frac{1}{2}}\mathbf{U}_y^T = (\mathbf{1}_{d\times N}\mathbf{\Lambda}_y^{\frac{1}{2}}\mathbf{U}_y^T)^T\mathbf{1}_{d\times N}\mathbf{\Lambda}_y^{\frac{1}{2}}\mathbf{U}_y^T.$$

We can therefore express the coordinate matrix $\mathbf{Y}$ as

$$\mathbf{Y} = \mathbf{1}_{d\times N}\mathbf{\Lambda}_y^{\frac{1}{2}}\mathbf{U}_y^T \tag{2}$$

and reformulate the problem as finding the components $\mathbf{\Lambda}_y$ and $\mathbf{U}_y^T$ that minimizes the Strain. Furthermore,

$$\begin{aligned}
\text{trace}\left[\mathbf{X}^T\mathbf{X}\mathbf{X}^T\mathbf{X}\right] &= \text{trace}\left[\mathbf{U}_x\mathbf{\Lambda}_x\mathbf{U}_x^T\mathbf{U}_x\mathbf{\Lambda}_x\mathbf{U}_x^T\right] \\
&= \text{trace}\left[\mathbf{U}_x\mathbf{\Lambda}_x^2\mathbf{U}_x^T\right] \\
&= \text{trace}\left[\mathbf{\Lambda}_x^2\right].
\end{aligned}$$

and

$$\text{trace}\left[\mathbf{Y}^T\mathbf{Y}\mathbf{X}^T\mathbf{X}\right]$$

We get that

$$\begin{aligned}
\text{Strain}(\mathbf{Y}) &= \text{trace}\left[\mathbf{\Lambda}_x^2 + \mathbf{U}_y\mathbf{\Lambda}_y^2\mathbf{U}_y^T - 2\mathbf{\Lambda}_y\mathbf{U}_y^T\mathbf{U}_x\mathbf{\Lambda}_x\mathbf{U}_x^T\mathbf{U}_y\right] \\
&= \text{trace}\left[(\mathbf{\Lambda}_x - \mathbf{\Lambda}_y\mathbf{U}_x^T\mathbf{U}_y)^2\right] \\
&= \sum_{i=1}^{N}(\lambda_i^x - \lambda_i^y\boldsymbol{u}_i^y\boldsymbol{u}_i^x)^2
\end{aligned}$$

Assume that the eigenvalues are ordered in descending order. We have that only $\lambda_1^x, \ldots, \lambda_D^x$, and $\lambda_1^y, \ldots, \lambda_d^y$ are non-zero. It follows that Strain is minimized by letting $\lambda_i^y = \lambda_i^x$ and $\boldsymbol{u}_i^y = \boldsymbol{u}_i^x$ for $i = 1, \ldots, d$. Equivalently, the Strain is minimized by setting the coordinate matrix $\mathbf{Y}$ as

$$\mathbf{Y} = \mathbf{1}_{d\times N}\mathbf{\Lambda}_x^{\frac{1}{2}}\mathbf{U}_x^T = \mathbf{1}_{d\times N}\mathbf{V}_x^T\mathbf{V}_x\mathbf{\Lambda}_x^{\frac{1}{2}}\mathbf{U}_x^T = \mathbf{1}_{d\times N}\mathbf{V}_x^T\mathbf{X}.$$

The second equality follows from singular value decomposition of $\mathbf{X}$. Here the equivalence with PCA becomes apparent, since the matrix $\mathbf{1}_{d\times N}\mathbf{V}_x^T$ contains the d eigenvectors of the matrix $\mathbf{X}\mathbf{X}^T \propto \hat{\mathbb{C}}_{\mathbf{X}}$, with largest corresponding eigenvalue.

Depending on what similarity measure, as well as what distance metric is used, different versions of Multidimensional Scaling is obtained. In cMDS, the similarity is measured by inner products, and if the distances are Euclidean it was shown to be equivalent to PCA. If the distances are no longer Euclidean, MDS is no longer equivalent to PCA since the similarity matrix is no longer given by the matrix product $\mathbf{X}^T\mathbf{X}$.

A similar method, called metric MDS, finds a low dimensional representation $\mathcal{Y}$ of the observed data set $\mathcal{X}$ by minimizing the difference in pairwise distances. The resulting cost function is found in Eq. (3) and is commonly called Stress.

$$\text{Stress}(\mathcal{Y}) = \sum_{i<j}(\Delta_{ij} - \delta_{ij})^2. \tag{3}$$

Here Δ_{ij} and δ_{ij} denotes the pairwise distances in the high dimensional and low dimensional spaces respectively. The Stress function lacks a general analytical solution, and hence it has to be minimized with a numerical optimization method, such as gradient descent. Unlike PCA, this method therefore does not provide an explicit mapping or a natural measure of the intrinsic dimension of the data, which is a clear drawback of this method. A schematic illustration of the Stress cost function can be seen in Fig. 5. The shape reveals that the cost will be higher for error in modelling large pairwise distances, rather than small pairwise distances. Hence, preservation of global structure is prioritised over local structure when minimizing with respect to the Stress cost function. Metric MDS is however not a linear method like PCA, and therefore is not restricted to data sets where the intrinsic structure is linear, which is an advantage compared to PCA and cMDS.

Different distance measures can be used when the Euclidean is not suitable for the intrinsic geometry of the data set. For example the ISOMAP algorithm [25] by using the *Geodesic* distance measure [2]. In this case MDS is not equivalent to PCA.

One can also modify the similarity measure, or dissimilarity measure in the case of metric MDS, to create new MDS algorithms. For example, Sammon Mapping [22] modifies the Stress cost function in Eq. (3) by normalising it as in Eq. (4), putting more emphasis on correctly modelling local structure.

$$\text{Sammon}(\mathcal{Y}) = \sum_{i<j}\frac{1}{\Delta_{ij}}\sum_{i<j}\frac{(\Delta_{ij} - \delta_{ij})^2}{\Delta_{ij}}. \tag{4}$$

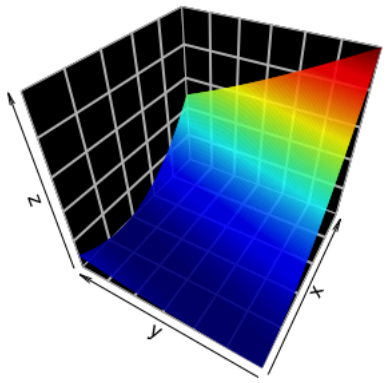


Figure 5: Schematic illustration of the MDS Stress cost function that illustrates the larger error resulted by incorrectly modelling large pairwise distances. Here the x-axis corresponds to high dimensional pariwise distance and the y-axis corresponds to low dimensional pairwise distance. The z-axis corresponds to the value of the Stress cost function, with higher values meaning higher cost.

A schematic illustration of the Sammon cost function can be seen in Fig. 6, which shows the larger cost for errors in modelling local structure.

4 Stochastic Neighbor Embedding

In this section we present the theory of Stochastic Neighbor Embedding (SNE), which is a method for dimensionality reduction and visualisation. The technique was originally introduced in [8] but gained popularity by a modified variant called t-distributed SNE, or t-SNE for short, which was proposed in [18].

The SNE method is an iterative algorithm that works by converting pairwise similarities into a probability distribution P over pairs of points in the high dimensional space $\mathcal{X} = \{x_1, \ldots, x_N\} \subset \mathbb{R}^D$, such that points that are similar have high probability. Then the algorithm finds points $\mathcal{Y} = \{y_1, \ldots, y_N\} \subset \mathbb{R}^d$, where $d < D$ is the assumed intrinsic dimensionality of the data, such that its corresponding probability distribution Q over pairwise similarities in $\mathcal{Y}$ matches P as closely as possible.

Different variants of SNE can be constructed depending on (i) what similarity measure $S_X : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ is used, how distances are measured and how the corresponding probability mass function P over the similarities is constructed in the high dimensional space (ii) what similarity measure $S_Y : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is used and how the corresponding probability mass function Q over pairwise similarities is defined in the low dimensional space and (iii) how the cost $C(P, Q)$, which is to be minimized, is defined based on the divergence between the probability distributions P and Q. In the following section we will focus on the particular SNE algorithm named t-distributed SNE.

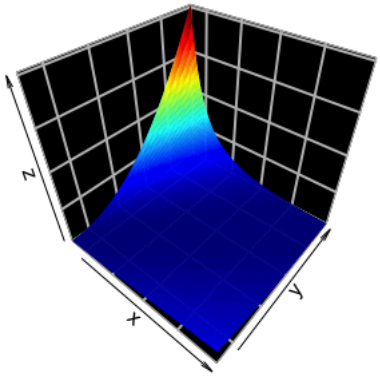


Figure 6: Schematic illustration of the Sammon mapping cost function that illustrates the larger error resulted by incorrectly modelling small pairwise distances. Here the x-axis corresponds to high dimensional pariwise distance and the y-axis corresponds to low dimensional pairwise distance. The z-axis corresponds to the value of the Sammon cost function, with higher values meaning higher cost.

4.1 Original SNE algorithm

In this section the theory behind the original SNE algorithm is presented.

Let $\mathcal{X} = \{x_1, \ldots, x_N\} \subset \mathbb{R}^D$ denote the high dimensional data set of dimension D containing N data points. Denote by Δ_{ij} the distance between data point x_i and x_j, which is assumed to be Euclidean in this context. Other distance measures can however be used. Next, define the high dimensional similarity measure $\mathcal{S}_\mathcal{X}$ as

$$\mathcal{S}_\mathcal{X}(x_i, x_j; \sigma_i) = e^{-\frac{\Delta_{ij}^2}{2\sigma^2}}.$$

This can be recognized as the kernel of a Gaussian distribution with variance parameter σ_i^2 which remains to be determined. This kernel smoothly weighs points that are further away from each other lower. The similarity decreases from 1, when $\Delta_{ij} = 0$, down towards 0 as Δ_{ij} tends toward infinity, the rate of which is determined by the parameter σ_i. Fig. 7 illustrates the kernel for different values of the variance parameter. Note that varying the parameter σ_i amounts to an isotropic re-scaling of the input data, since

$$e^{-\frac{||x_i - x_j||^2}{2\sigma^2}} = e^{-||\frac{x_i}{\sqrt{2}\sigma} - \frac{x_j}{\sqrt{2}\sigma}||^2}.$$

Next, for each point x_j, a probability mass function (p.m.f.) is defined over

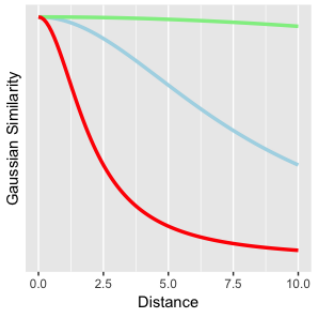


Figure 7: Figure illustrating Gaussian kernel for increasing values of the σ parameter.

all points x_i as

$$P(x_i \mid x_j, \sigma_j) = \begin{cases} \frac{e^{-\frac{||x_i - x_j||^2}{2\sigma_j}}}{\sum_{k \neq j} e^{-\frac{||x_k - x_j||^2}{2\sigma_j^2}}} = \frac{S_X(x_i, x_j; \sigma_j)}{\sum_{k \neq j} S_X(x_k, x_j; \sigma_j^2)} & i \neq j \\ 0 & i = j \end{cases}.$$

We will often denote $P(x_i \mid x_j, \sigma_j)$ as $p_{i|j}$ for short.

This can be interpreted as a normalised similarity measure, since most probability mass will be placed on the most similar neighbor, decreasing as the similarity decreases. This probability function will represent the high dimensional data set.

The probability mass function requires an appropriate choice of σ_j, which in turn determines how the high dimensional data set is represented. Note that as $\sigma_j \to 0$, all mass will be placed on the nearest neighbor. On the other hand, as σ_j increases from zero, mass is continuously shifted to increase for all neighbors, reaching a uniform distribution as $\sigma_j \to \infty$. Choosing the parameter σ_j therefore has a large effect on what is counted as similar in the high dimensional space. As noted above, changing σ_j amounts to an isotropic re-scaling of the data, hence the p.m.f. does depend on the scale of the data points.

An artifact resulting from this definition is that if the data has a large isotropic scaling, then the probability mass will simply be put on the very local neighbor to a given point, and hence result in a poor representation of the global structure of the data set. In other words, the probability representation would be sensitive to the scaling of the input data. In addition, points located in a sparse region would be similar to fewer number of points compared to points located in dense regions.

To circumvent this problem, the *perplexity* parameter is introduced. The perplexity, denoted by $\text{Perp}(P)$, of a probability mass function P, is defined as

$$\text{Perp}(P) = 2^{H(P)}.$$

Here $H(P)$ is the Shannon Entropy of a random variable X distributed according to P, defined in Section 2.1 as

$$H(X) = \mathbb{E}\left[\log \frac{1}{P(X)}\right] = \sum_{i=1}^{N} p_i \log \frac{1}{p_i}.$$

It was also shown in Section 2.1 that

$$0 \leq H(P) \leq \log N$$

with $H(P) = 0$ when P places all probability mass on a single outcome and $H(P) = \log N$ when P is uniform. Therefore, it follows that

$$1 \leq \text{Perp}(P) \leq N.$$

So $\text{Perp}(P)$ is equal to 1 when $\sigma_j = 0$ and equal to N when $\sigma = \infty$. Perplexity could therefore loosely be interpreted as the expected number of neighbors x_i of x_j that has a non-zero probability, or in other words what is counted as local. An illustration is found in Fig. 8 showing how the p.m.f. changes for varying perplexity values.

The σ_j parameter is now chosen so that the perplexity is equal to a fixed value u between 1 and N for each point. The scaling of the input data now has no effect on the representation, since the data is locally rescaled according to the perplexity chosen. Since $\text{Perp}(P)$ is monotonically increasing in σ, the correct value σ resulting in perplexity equal to u can be found approximately and efficiently by a binary bisection search. However, the problem now arises that the perplexity value needs to be chosen that correctly models locality in the high dimensional space.

Next, the problem remains to find a low dimension representation $\mathcal{Y} \subset \mathbb{R}^d$ of $\mathcal{X}$.

Similar to the high dimensional representation, a low dimensional similarity measure $\mathcal{S}_\mathcal{Y}$ between points y_i and y_j is defined as

$$\mathcal{S}_\mathcal{Y}(y_i, y_j; \sigma) = e^{-\frac{\delta_{ij}^2}{2\sigma^2}}.$$

Here δ_{ij} denotes the low dimensional distance measure between the points y_i and y_j and taken to be Euclidean. The variance parameter can be taken to be equal for all points in this case, since it simply results in a re-scaling of the resulting map. This can be seen as

$$\mathcal{S}_\mathcal{Y}(y_i, y_j; \sigma) = e^{-\frac{\delta_{ij}^2}{2\sigma^2}} = e^{-\frac{||y_i - y_j||^2}{2\sigma^2}} = e^{-||\frac{y_i}{\sqrt{2}\sigma} - \frac{y_j}{\sqrt{2}\sigma}||^2}.$$

Then, the probability distribution Q is defined as in the high dimensional representation

$$Q(y_i \mid y_j, \sigma) = \begin{cases} \frac{e^{-\frac{\delta_{ij}^2}{2\sigma}}}{\sum_{k \neq j} e^{-\frac{\delta_{jk}^2}{2\sigma}}} = \frac{\mathcal{S}_\mathcal{Y}(y_i, y_j; \sigma)}{\sum_{k \neq j} \mathcal{S}_\mathcal{Y}(y_k, y_j; \sigma)} & i \neq j \\ 0 & i = j \end{cases}.$$

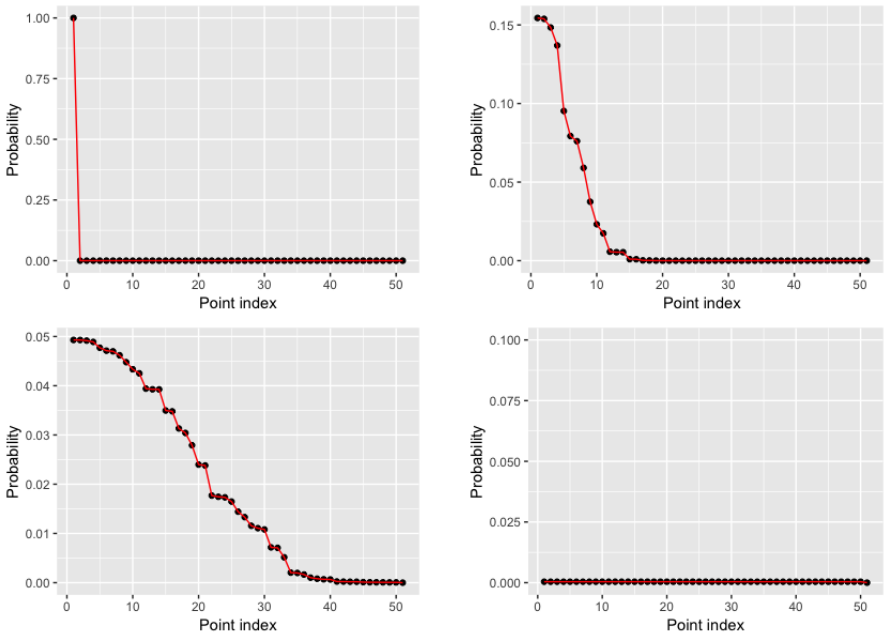


Figure 8: The plots illustrates the probability distribution $P(x_i \mid x_j)$ over pairwise similarities of 50 points, for increasing values of the perplexity parameter. From top left to bottom right the perplexity values are set to 1, 10, 30 and 50 respectively.

We will write $Q(y_i \mid y_j, \sigma)$ as $q_{i|j}$ for brevity.

The low dimensional set is found by minimizing the Kullback-Leibler divergence, defined in Definition 2, found in Section 2.1 on page 7, with respect to the low dimensional data points y_i. Hence, the cost function can be expressed as

$$C(P, Q) = \sum_{i=1}^{N} \sum_{j=1}^{N} p_{i|j} \log \frac{p_{i|j}}{q_{i|j}}.$$

For brevity we have written $p_{i|j} = P(x_i \mid x_j, \sigma_j)$ and likewise for $q_{i|j}$. The cost function is not convex with respect to y_i, hence no analytical solution can be obtained. The cost is therefore minimized with gradient descent. The gradient can be calculated analytically, and is of a surprisingly simple form:

$$\frac{\partial C}{\partial y_j} = 2 \sum_{i=1, i \neq j}^{N} (p_{i|j} - q_{i|j} + p_{j|i} - q_{j|i})(y_i - y_j).$$

Thus, given the initial set of points $\mathcal{Y}^{(0)}$, which is usually obtained by sampling points from a d-dimensional Gaussian distribution with mean zero and standard deviation 10^{-4}. A reason for chosing the standard deviation small is that when the pairwise distances δ_{ij} are all small,

$$q_{i|j} = \frac{e^{-\frac{\delta_{ij}^2}{2\sigma}}}{\sum_{k \neq j} e^{-\frac{\delta_{jk}^2}{2\sigma}}} \approx \frac{1}{N}.$$

The initial distribution Q is therefore approximately uniform. The algorithm then proceeds in an iterative manner. By gradient descent, each iteration t updates the low dimensional points as

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} - \eta \frac{\partial C}{\partial \mathcal{Y}^{(t-1)}}$$

until the algorithm has converged at a local minimum. Note again that the cost function is not convex, and hence it is not guaranteed to converge to a global minimum.

4.2 t-Distributed SNE

In this section the theory of the t-distributed Stochastic Neighbor Embedding (t-SNE) algorithm is presented, where its name is due to the low dimensional similarity $\mathcal{S}_\mathcal{Y}$ being defined as the kernel of a t-distribution. The main reasons for using this similarity measure are (i) computational as it does not involve any exponential, (ii) a mismatch in tails compared to the Gaussian kernel results in more clearly separated clusters and compensation for increased distances in high dimension and (iii) approximate scale invariance for large distances. The algorithm, originally proposed in [18] has gained a lot of popularity as a data

visualization tool, for example in visualisation of cellular data in molecular biology [19], and is often considered as the state of the art of data visualisation. The idea behind the algorithm is to prioritise local structure, that is the algorithm puts emphasis on preserving small pairwise distances, but tolerates errors in modelling large distances as long as they are placed far apart in the low dimensional map.

Let $\mathcal{X} = \{x_1, \ldots, x_N\}$ denote the data set containing N points x_i of dimension D. The similarity $S_\mathcal{X}$ between data points x_i and x_j is defined in the same way as in the original SNE algorithm,

$$\mathcal{S}_\mathcal{X}(x_i, x_j; \sigma) = e^{-\frac{\Delta_{ij}^2}{2\sigma^2}}.$$

As in the original algorithm, for each point x_j, a probability mass function P'_j is defined over the points as

$$P'_j(x_i \mid x_j, \sigma_j) = \begin{cases} \frac{e^{-\frac{\Delta_{ij}^2}{2\sigma_j}}}{\sum_{k \neq j} e^{-\frac{\Delta_{kj}^2}{2\sigma_j}}} = \frac{S_X(x_i, x_j; \sigma_j)}{\sum_{k \neq j} S_X(x_k, x_j; \sigma_j)} & i \neq j \\ 0 & i = j \end{cases}. \tag{5}$$

The variance parameter is chosen so as to maintain a constant perplexity for each point.

One key difference between SNE and t-SNE is that the probability distribution P is made symmetric, and instead defined as a joint probability mass function over pairs of points. The probability distribution P over the $\frac{N(N-1)}{2}$ pairwise similarities is defined as

$$P(x_i, x_j) = \frac{P'_j(x_i|x_j) + P'_i(x_j|x_i)}{2N}. \tag{6}$$

We will denote $P(x_i, x_j)$ as p_{ij}. Hence, a consequence of this definition is that that it is symmetric, that is $p_{ij} = p_{ji}$ and $p_{ii} = 0$. The advantage of symmetrisation is mainly compoutational as fewer values of the low dimensional p.m.f needs to updated in each iteration of the gradient descent algorithm. Furthermore, it has the propery that

$$\sum_{i=1}^{N} p_{ij} > \frac{1}{2N}$$

which means a significant portion of probability mass will always be placed on points. Without symmetrising the placement of an outlier x_i, for example, would otherwise have no impact on the cost:

$$p_{i|j} \approx 0$$

for every other point x_j. For an outlier x_j, the symmetrised probability of the outlier will be

$$\sum_{i=1}^{N} p_{ij} = \sum_{i=1}^{N} \frac{p_{i|j} + p_{j|i}}{2N} \approx \sum_{i=1}^{N} \frac{p_{i|j}}{2N} = \frac{1}{2N}.$$

Dividing by the denominator $2N$ makes the joint probability distribution sum to 1. If x_i and x_j are located in regions of similar density, then

$$P(x_i, x_j) \approx \frac{P_i'(x_j|x_i)}{N}$$

and hence the construction could motivated by Bayes formula: $P(x_i, x_j) = P(x_j \mid x_i)P(x_i)$ and $P(x_i) = \frac{1}{N}$.

The low dimensional similarity measure $\mathcal{S}_\mathcal{Y}$, defined over pairwise distances δ_{ij}, usually defined as the Euclidean distance between y_i and y_j, is measured using the kernel of the t-distribution with one degrees of freedom, instead of the Gaussian kernel used in the high dimensional space. This is the second key difference between t-SNE and SNE.

$$\mathcal{S}_\mathcal{Y}(y_i, y_j) = \frac{1}{1 + \delta_{ij}^2}.$$

The mismatch in kernels thus leads to similarity being measured in different ways in the high dimensional and low dimensional spaces. An illustration of the differences between the Gaussian kernel and the t-distribution kernel can be seen in Fig. 9. As can be seen in the figure, the kernels are approximately equal for small distances. This also follows by a second order two Taylor approximation:

$$e^{-\delta^2} \approx \frac{1}{1 + \delta^2}.$$

There is a mismatch for larger distances, where the Gaussian kernel decreases faster to zero as distances gets larger compared to the t-distribution kernel.

Thus, even if the dimension is chosen equal for the high dimensional and low dimensional spaces, the result of the algorithm will be different than the input data, mainly due to a large pairwise distance in high dimensional space having to be mapped by an even larger distance in low dimensional space to compensate the for the mismatch.

An example can be seen in Fig. 10. Here the two points that belong to different clusters, and thus are relatively far apart, are put even further apart in the t-SNE embedding due to the mismatch in tails of the similarities. This mismatch therefore aids in more clearly separating clusters. The authors also motivate the choice of kernel by that if there is a large mismatch in dimensionality between the high and low dimensional spaces, distances in the high dimensional space will tend to be much larger. This is related to the classical problem of the *curse of dimensionality* in machine learning. The mismatch in tails can therefore compensate for the mismatch in dimensionality.

Other technical advantages with the t-distribution kernel is that it is computationally faster to evaluate since it does not involve any exponential, which would otherwise need to be approximated in each iteration, as in the original SNE algorithm.

Subsequently, the low dimensional probability mass function Q is defined as

$$Q(y_i, y_j) = \frac{S_Y(y_i, y_j)}{\sum_{k \neq l} S_Y(y_k, y_l)} \tag{7}$$

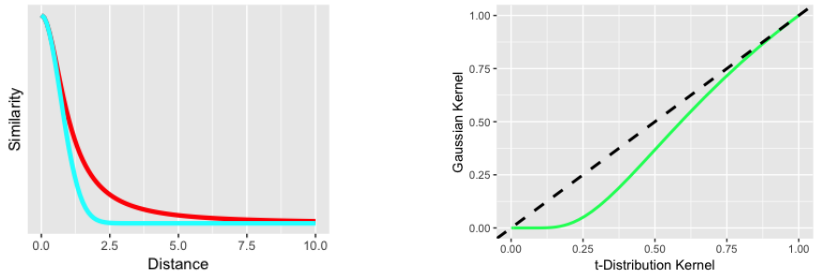


Figure 9: The figures illustrates the difference between the t-distribution and Gaussian kernels. The left figure shows similarity as a function of distance with t-distribution kernel in red and Gaussian kernel in blue. The figure on the right visualises the difference by plotting t-distribution (x-value) against Gaussian similarity (y-value).

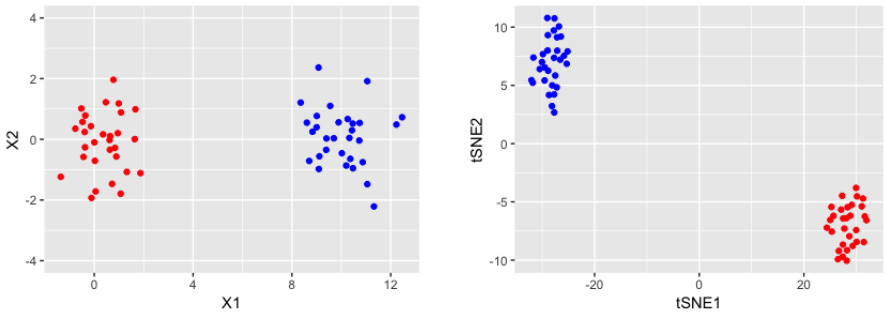


Figure 10: The figure on the right shows a data set containing two 2-dimensional clusters. The right figure shows the t-SNE embedding into 2 dimensions with perplexity set to 15.

for $i \neq j$ and $Q(y_i, y_i) = 0$. We will denote $Q(y_i, y_j)$ as q_{ij}. As mentioned above, a consequence of the mismatch in tails is that points far away in $\mathcal{X}$ will be mapped further away in $\mathcal{Y}$. This can be seen from Fig. 9 which shows the heavier tails of the t-distribution compared to the Gaussian kernel. Hence, to attain the same similarity value of the kernels the low dimensional distance has to be larger than the high dimensional distance. The p.m.f. Q also has the property of being approximately scale invariant for large distances, as

$$q_{ij} = \frac{\frac{1}{1+c\delta_{ij}^2}}{\sum_{k\neq l} \frac{1}{1+c\delta_{kl}^2}} \approx \frac{\frac{1}{c\delta_{ij}^2}}{\sum_{k\neq l} \frac{1}{c\delta_{kl}^2}} = \frac{\frac{1}{\delta_{ij}^2}}{\sum_{k\neq l} \frac{1}{\delta_{kl}^2}}$$

where c is some scaling factor. Hence, when distances are large, the pairwise probability will be almost unaffected by changes in scale. For large distances, the relative distances will determine the probability and not exact distances.

The divergence between P and Q is measured with the Kullback-Leibler divergence, as in the SNE algorithm, defined as

$$C(P, Q) = KL(P||Q) = \sum_{i\neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

Here p_{ij} and q_{ij} are the pairwise probabilities defined in Eq. (6) and Eq. (7) respectively. A property of the cost function, just as in the original SNE algorithm, is that it is asymmetric in P and Q. This asymmetry leads to higher cost when modelling a small high dimensional distance by a large distance in the low dimensional space, compared to the error of modelling a large high dimensional distance by a small low dimensional distance. A schematic visualisation is found in Fig. 11, illustrating this asymmetry.[1] This property of the cost function therefore prioritizes keeping points that are close in the high dimensional space being close in the low dimensional space as well. It is however less accurate in keeping distant points in the high dimensional space distant in the low dimensional space. As can be seen in Fig. 11, the cost for modelling large distances incorrectly is almost negligible. Hence, error induced by modelling local structure incorrectly has a larger effect on the cost function than error in modelling global structure. Comparing the t-SNE cost function to the cost of the SNE algorithm, the cost is even higher for errors in modelling small distances in t-SNE compared to SNE. The cost is also lower for error in modelling large distances in t-SNE. Hence t-SNE puts even more emphasis on correctly modelling local structure and tolerates higher error in modelling global structure. Since the algorithm preserves normalised distances, see Eq. (5) and Eq. (7), it

[1] The visualisation was created by simplifying the term in the sums as follows.

$$p_{ij} \approx e^{-\Delta_{ij}^2} \text{ and } q_{ij} \approx \frac{1}{1+\delta_{ij}^2}$$

and

$$p_{ij} \log \frac{p_{ij}}{q_{ij}} \propto -p_{ij} \log q_{ij} \approx e^{-\Delta_{ij}^2} \log(1+\delta_{ij}^2).$$

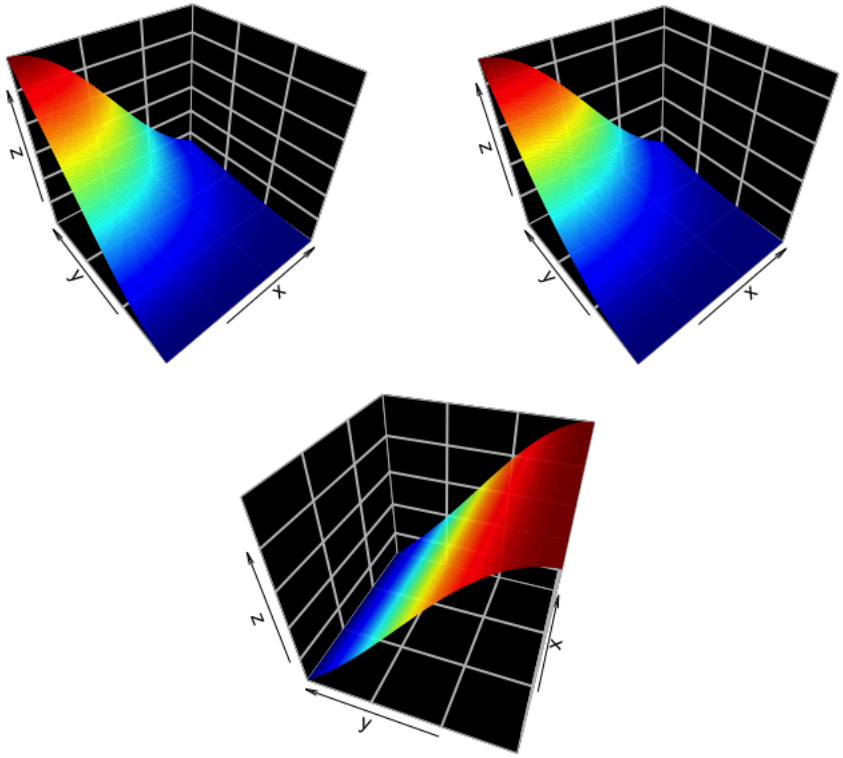


Figure 11: A schematic illustration of the Kullback-Leibler cost function, illustrating its asymmetry. Here the x-axis corresponds to high dimensional pairwise distance and the y-axis corresponds to low dimensional pairwise distance. The z-axis corresponds to the value of a $p_{ij} \log \frac{p_{ij}}{q_{ij}}$ term in the Kullback-Leibler cost function, with higher values meaning higher cost. The top figures corresponds to the cost of the t-SNE (left) and SNE (right) algorithms. The bottom figure plots the cost of t-SNE divided by the cost of SNE, illustrating their relationship.

still places points that are far apart in the high dimensional map far apart in the low dimensional map, however, but exact relations may not be preserved. To elaborate, correctly modelling small pairwise distance means placing high probability on those small distances, since the probability must sum to one, this means that less probability mass is placed on distant points.

The cost function is finally minimized with gradient descent. The gradient is given by

$$\frac{\partial C}{\partial y_i} = \sum_j (p_{ij} - q_{ij}) \frac{1}{1 + \delta_{ij}^2} (y_i - y_j).$$

In order to find $\mathcal{Y}$, starting with an initial set of points $\mathcal{Y}^{(0)}$, the algorithm iteratively moves the points around until the algorithm converges or a maximum number of iterations has been reached. Important to note is that the cost function is non-convex in $\mathcal{Y}$. Hence, the algorithm runs the risk of getting stuck in a local minimum.

5 Evaluation

As noted in Section 2, the result of dimensionality reduction needs to be evaluated. If the data is reduced to two dimensions, then the evaluation can be carried out qualitatively by simply looking at the result. This is however highly subjective, and not possible if the reduced dimension is higher than 3. There is therefore a need for a more precise quantitative notion of how trustworthy the reduction is. The PCA algorithm presented in Section 3 has a built in evaluation tool in the reconstruction error. However methods such as t-SNE and SNE does not find an explicit mapping from the high dimensional space to the low dimensional space, so there is no means to calculate the reconstruction error. Therefore other evaluation criteria needs to be developed. With a precise mathematical definition, it is transparent what is meant by a successful dimensionality reduction depending on what quantitative notion of quality is used. One class of methods for evaluation is *rank based criteria*, that quantifies the preservation of the rank order of pairwise distances. A multitude of different measures have been suggested in the literature, we refer to [6, 20] for a summary and more suggestions.

5.1 Rank Based Criteria

A method for evaluating dimensionality reduction is to consider the preservation of the rank order of pairwise distances between the high and low dimensional spaces [16, 13]. Often, one restricts for each point x_i in the high dimensional space $\mathcal{X}$ to only consider how many of the K nearest points to x_i remains among the K nearest to the corresponding low dimensional point y_i in $\mathcal{Y}$. Here K is a whole number in the interval $[1, N-1]$. So for different values of neighborhood size K, the evaluation is more or less restricted to preservation of local structure.

To establish some notation, define

$$N_X^K(i) = \{j : \Delta_{ij} < \Delta_{iK}\}$$

the index set of the K nearest neighbors to the point indexed i in $\mathcal{X}$. Similarly define N_Y^K in the low dimensional space. A measure of the preservation of K nearest neighbor neighborhoods can then be defined as

$$N(K) = \frac{1}{NK} \sum_{i=1}^{N} |N_X^K(i) \cap N_Y^K(i)|.$$

Here $|A|$ denotes the cardinality of a set A and $A \cap B$ the intersection between sets A and B. The measure $N(K)$ lies in the interval $[0, 1]$, and is equal to 0 with complete distortion, and equal to 1 with perfect preservation with respect to the K-ary neighborhoods of each of the points. Since complete distortion is unlikely to occur, a similar measure instead compares the result with a completely random embedding. Assuming that the positioning of points in the low dimensional space is completely random, so that for each point i, it is equally likely that a point j in $N_X^K(i)$ will have a corresponding point in $N_Y^K(i)$. Then the number of points in the intersection will follow a hypergeometric distribution $\text{Hyp}(N, K, K)$. Thus the expected number in each intersection will thereby be $\frac{K^2}{N-1}$. So the preservation will be $\frac{1}{NK} N \frac{K^2}{N-1} = \frac{K}{N-1}$ on average. We get that the improvement from a random representation can be quantified as

$$R(K) = \frac{(N-1)N(K) - K}{N - 1 - K} \tag{8}$$

so that it equals 0 on average when the embedding is completely random, and equals 1 if the embedding perfectly preserves the rank ordering. Note that this measure will can be negative if the algorithm performs worse than random. In case of complete distortion, when $N(K) = 0$, the value of $R(K)$ will be equal to $-\frac{K}{N-1-K}$.

Finally, the quality measure, which we will denote as $\overline{R}$, is defined by averaging the preservation over all neighborhood sizes.

$$\overline{R} = \sum_{K=1}^{N-2} \frac{R(K)}{K} / \sum_{K=1}^{N-2} \frac{1}{K} \tag{9}$$

Hence, the $\overline{R}$ criterion can be interpreted as the average neighborhood preservation over all neighborhood sizes, and is given by the are under the curve of the R values, normalised by dividing by the maximum area. The area under the curve defined by the $R(K)$ values is thus given by $\sum_{K=1}^{N-2} \frac{R(K)}{K}$ and the maximum area is given by $\sum_{K=1}^{N-2} \frac{1}{K}$. The reason for normalising is that otherwise the R value may depend on the number of data points. A higher value of $\overline{R}$ corresponds to a better preservation of the data. The neighborhood size of $K = N - 1$ is excluded here since it always gives the same value of $R = 1$.

6 Benchmarking

In this section the t-SNE algorithm will be benchmarked and compared with the SNE and PCA algorithms presented in previous sections. The purpose is to illustrate the properties and potential artifacts of t-SNE with various test cases. The test cases were mainly inspired by [14] and [27]. In Section 6.1 the t-SNE algorithm is applied to various test data sets and compared to SNE and PCA. Then, in Section 6.2, potential artifacts of the t-SNE algorithm is explored to highlight possible pitfalls when using the algorithm.

6.1 Test Cases

For all test cases, the low dimensional points $\mathcal{Y}^{(0)}$ used in the SNE and t-SNE algorithms were initialised by sampling N points from a Gaussian distribution with zero mean and variance 10^{-4}, which are the default initialisation in the original implementations. The number of iterations is set to 5000 throughout, since both SNE and t-SNE were found to have converged at this number. It is also the default setting of similar experiments conducted by the authors of the algorithm. In order to avoid getting stuck in a local optimum in the gradient descent optimization, since the cost functions of both t-SNE and SNE are not convex, multiple maps were produced with different random initialization for each test case. The reductions are evaluated by the $R(K)$ and $\overline{R}$ quality

	t-SNE	SNE	PCA
MNIST	0.41163	-0.00092	0.14852
Swiss Roll	0.73052	0.73440	0.62236
S-Curve	0.83929	0.70484	0.67466
Curved Clusters	0.43626	0.33777	0.36265

Table 1: The table contains $\overline{R}$ scores for the algorithms analysed for each data set. A higher score means higher quality of the embedding in terms of rank order neighborhood preservation averaged over all neighborhood sizes. A score of 0 means the quality is equal to that of a random embedding, and a negative score that the quality is worse than a random embedding.

measures defined in Eq. (8) and Eq. (9) in Section 5 respectively. The quality measure $\overline{R}$, see Eq. (9) in Section 5, results are summarised for each data set in Table 1. A higher value corresponds to better preservation, and a negative score means the reduction preserved that structure worse than a random embedding. The t-SNE algorithm outperforms both SNE and PCA according to this measure on all data sets except on the Swiss Roll data set, where SNE performs slightly better although with a small margin. The PCA algorithm was found to perform consistently worse than the other algorithms on test cases analysed here.

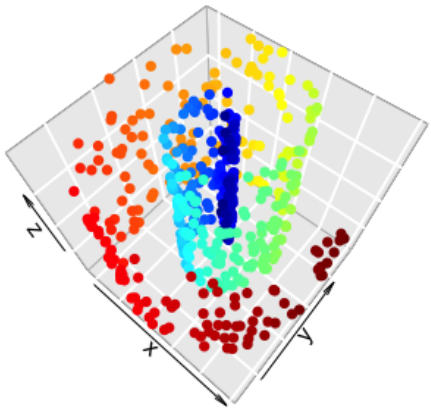


Figure 12: Top left shows 500 points drawn from the Swiss Roll data set.

6.1.1 Swiss Roll

Here we apply the t-SNE, SNE and PCA algorithms to the Swiss Roll data set, commonly used in benchmarking of dimension reduction techniques. The data set is plotted in Fig. 12. The data was generated by sampling 500 points u_1 from the uniform distribution $U_1 \sim U(0, 2\pi)$ and 500 points u_2 from the uniform distribution $U_2 \sim U(-1, 1)$. The three-dimensional coordinates are then given by

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} u_1 \cos 2u_1 \\ u_1 \sin 2u_1 \\ u_2 \end{pmatrix}.$$

Two t-SNE mappings were created, with perplexity values of 20 and 50 respectively. The t-SNE embedding is shown in Fig. 13. The top figure shows the result with perplexity equal to 50. This value is too large, since two points lying in different layers in the Swiss Roll are counted as close, which breaks the connectedness between the red and orange points. A lower perplexity value of 20, which is shown in the bottom figure, creates a good low dimensional representation. The t-SNE algorithm preserves the local connectedness of the Swiss Roll, but the global structure is not perfectly preserved.

The result of dimensionality reduction of the Swiss Roll data set using the PCA algorithm can be seen in Fig. 14. PCA is clearly not able to preserve all the structure of the data, since the result appears to be a one dimensional spiral. Hence in this case the t-SNE algorithm performs better.

The result of the SNE algorithm can be seen if Fig. 15. The plot on the right is the result with perplexity parameter set to 50, and the right plot with perplexity set to 20. The results are similar for both perplexity values and highly resembles the PCA projection. The algorithm is does not manage to unfold the Swiss Roll as well as t-SNE.

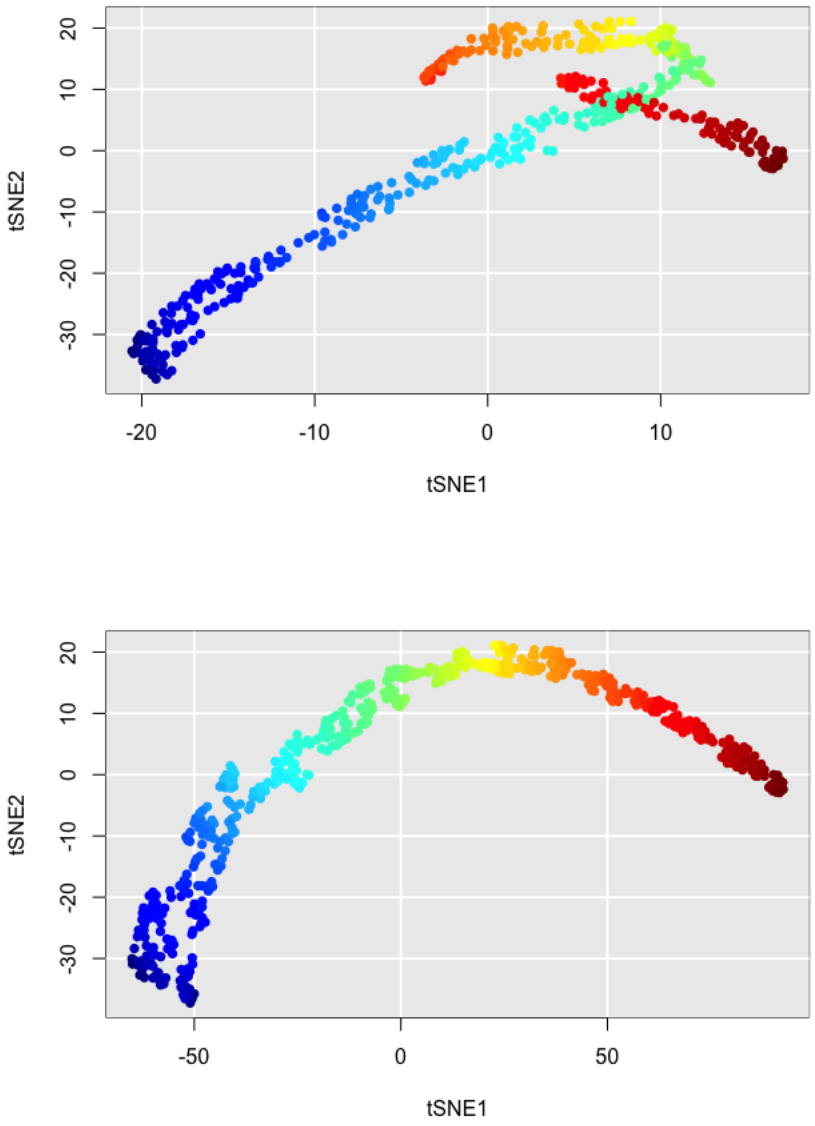


Figure 13: The top figure shows the t-SNE projection with perplexity 50. The bottom picture shows the t-SNE projection with the perplexity parameter set to 20.

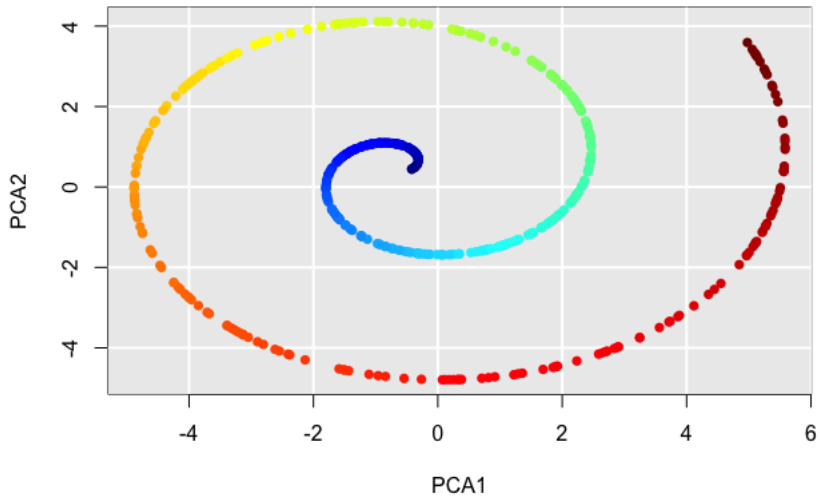


Figure 14: Reduction of the Swiss Roll data set using the PCA-algorithm. This illustrates PCA's shortcoming in reducing non-linear data sets.

In Table 2 the $\overline{R}$ scores are summarised for each of the tests. According to this measure, the SNE algorithm works slightly better for unfolding the Swiss Roll than t-SNE. This is surprising as from a qualitative evaluation t-SNE performs better. This shows that the rank based critera based on the rank order of Euclidean distances may not directly imply whether a non-linear manifold is unfolded or not. Unsurprisingly, PCA reports the lowest score. The $R(K)$ scores, see Eq. (8) in Section 5, for each test and neighborhood size K is shown in Fig. 16. The t-SNE algorithm outperforms the others for small neighborhood sizes, but for larger values of K, PCA and SNE preserves the rank ordering better. This confirms that t-SNE tends to prioritise local structure, but may distort larger distances.

6.1.2 S-Curve

In this section we look at the S-Curve data set. The data set was generated by sampling 800 points u_1 from the uniformly distributed random variable $U_1 \sim U(-\frac{2\pi}{3}, \frac{2\pi}{3})$ and an additional 800 points u_2 from the uniformly distributed random variable $U_2 \sim U(0, 5)$. Then, for the first 400 points the three

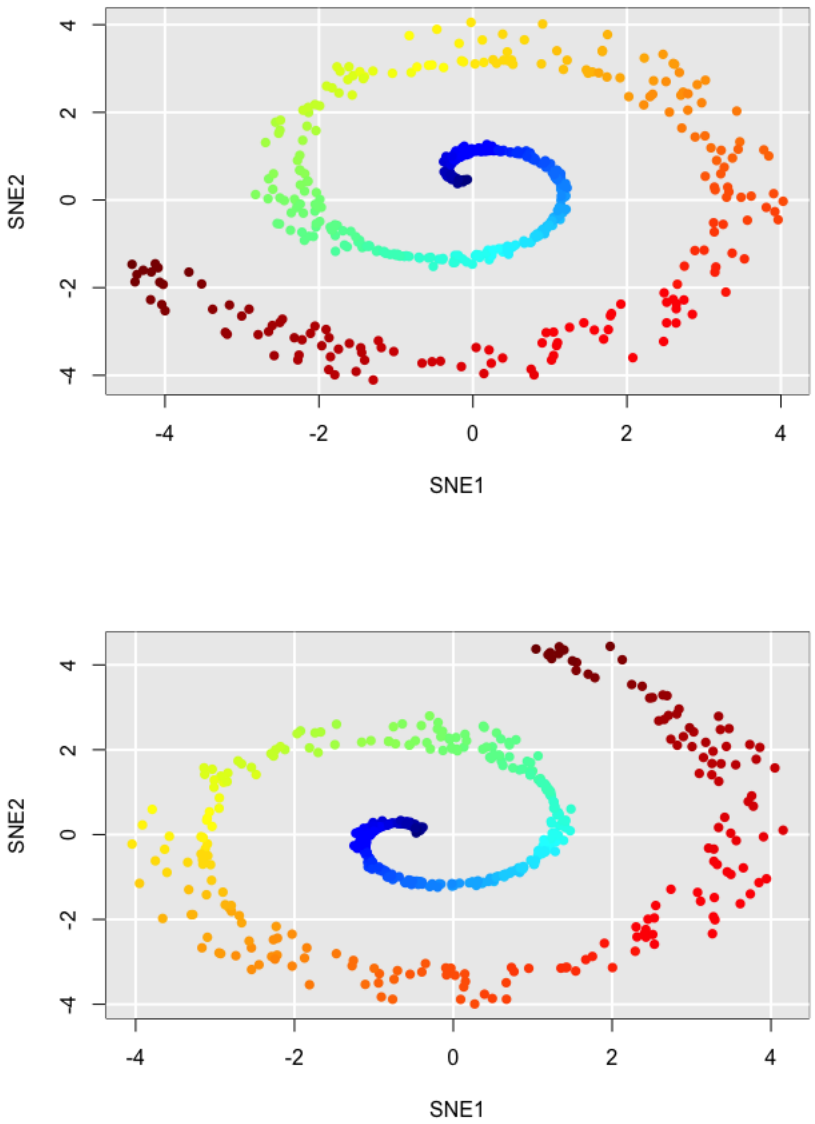


Figure 15: The figure displays the reduction of the Swiss Roll data set to dimension 2 using the SNE algorithm. The top left plot and top right plot are the results using perplexity 50 and 20 respectively.

	$\overline{R}$
t-SNE (Perp. 20)	0.72089
t-SNE (Perp. 50)	0.73052
SNE (Perp. 20)	0.73440
SNE (Perp. 50)	0.73143
PCA	0.62236

Table 2: The table contains $\overline{R}$ scores for each test case on the Swiss Roll data set. A higher score means better preservation of rank ordering.

dimensional points are given by

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos u_1 + \epsilon_1 \\ u_2 + \epsilon_2 \\ \sin u_1 + \epsilon_3 \end{pmatrix}.$$

The remaining 400 points were transformed using the mapping

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -\cos u_1 + \epsilon_1 \\ u_2 + \epsilon_2 \\ 2 - \sin u_1 + \epsilon_3 \end{pmatrix}.$$

Here ϵ_i, $i = 1, 2, 3$, denotes Normally distributed independent random variables from the $\mathcal{N}(0, 1)$ distribution modelling random noise. The data is plotted in Fig. 17. The desired reduction is thus to a two-dimensional rectangular manifold.

The result of the t-SNE algorithm can be found in Fig. 18. The top plot shows the result when perplexity was set to 50. In this case the perplexity is too low and the algorithm clusters points within the two-dimensional rectangle. The right plot is the result when perplexity was set to 200. With this parameter setting the algorithm successfully unfolds the S-Curve.

Fig. 19 shows the reduction using the PCA algorithm. Since PCA finds the best fit plane, it is naturally not able to preserve the non-linear structure of the data set.

In Fig. 20 the result of the SNE algorithm is plotted. The result surprisingly resembles the reduction by the PCA algorithm. As with the t-SNE algorithm the result is better with higher perplexity. The SNE algorithm is however not as successful at untangeling the blue and red points.

The $\overline{R}$ scores are summarised in Table 3. The t-SNE algorithm outperforms SNE and PCA by a large margin. In Fig. 21 the $R(K)$ values are plotted, for every neighborhood size K, for each of the test cases. The scores of PCA and SNE are surprisingly similar. The t-SNE algorithm achieves the highest score for smaller neighborhood sizes, as expected since it prioritises preservation of small distances.

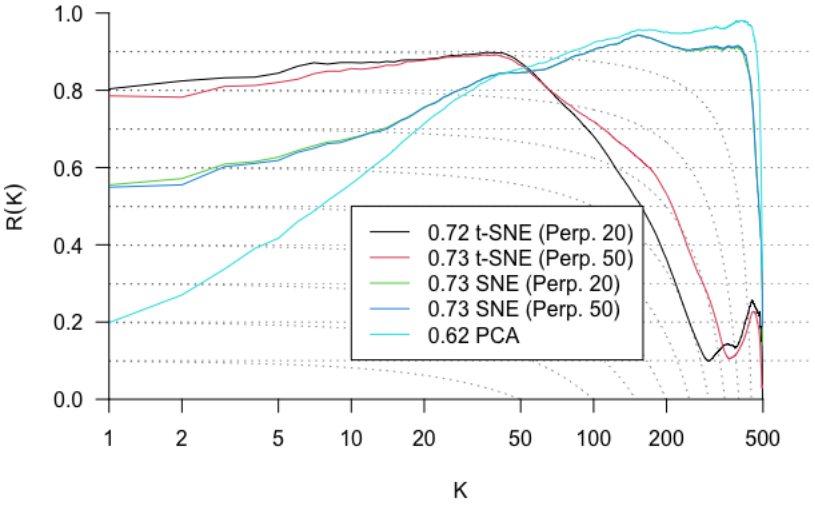


Figure 16: The figure shows R-curves for all of the tests conducted on the Swiss Roll data set. The higher the curve, the higher the preservation.

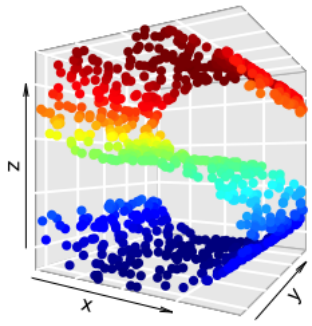


Figure 17: The plot illustrates 800 points from the S-Curve data set. The figure is best viewed in color.

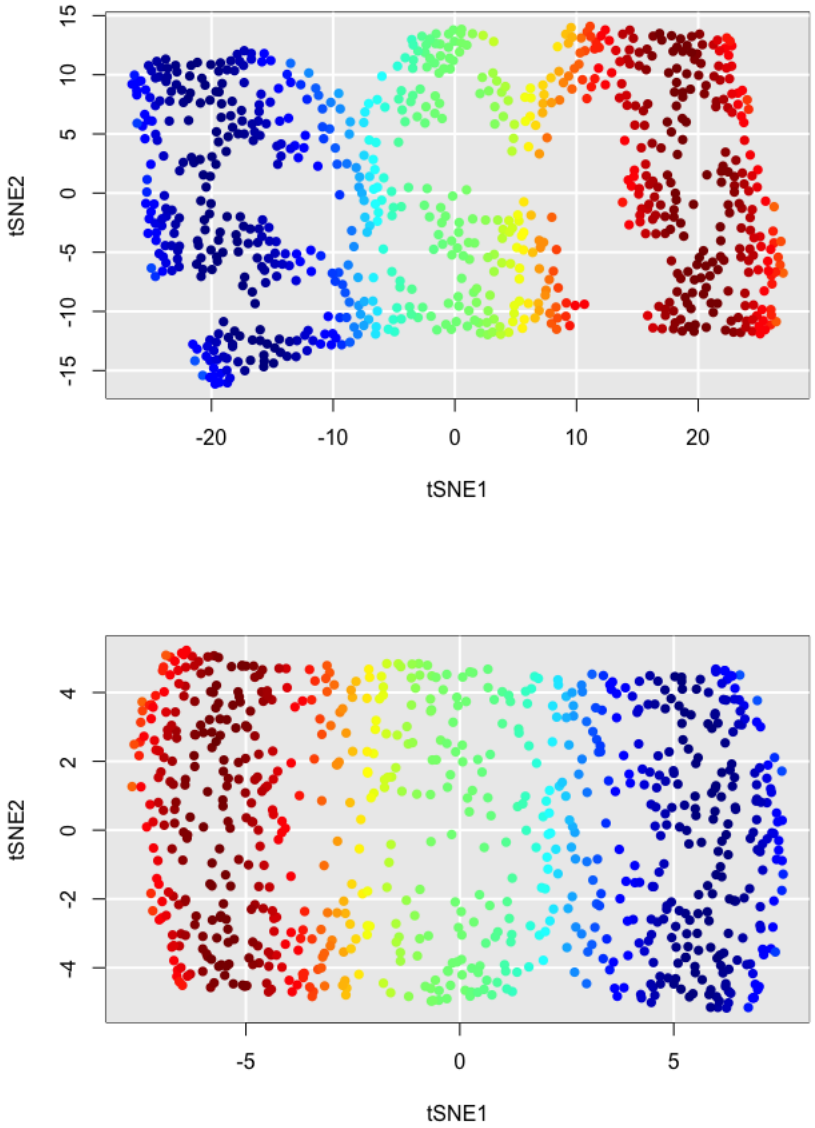


Figure 18: The figure shows the two dimensional reduction of the S-Curve data set using the t-SNE algorithm. The top figure is the result with the perplexity parameter set to 50, and the bottom with perplexity parameter set to 200.

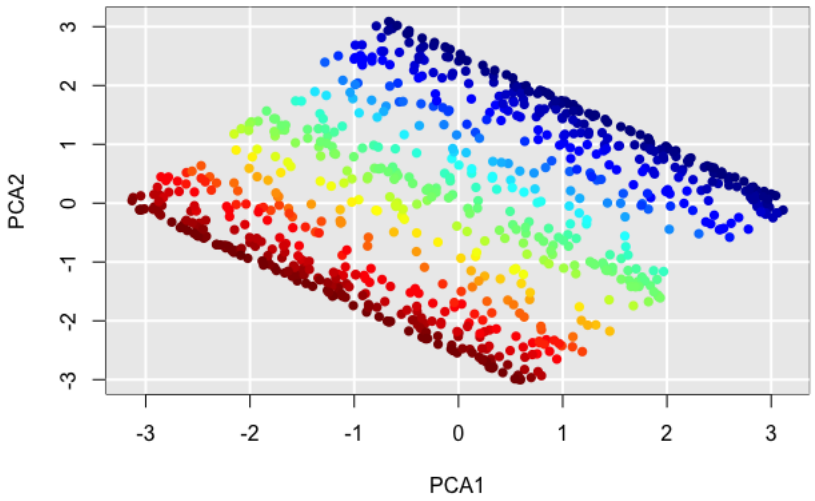


Figure 19: This figure shows the first two principal components of the S-Curve data set.

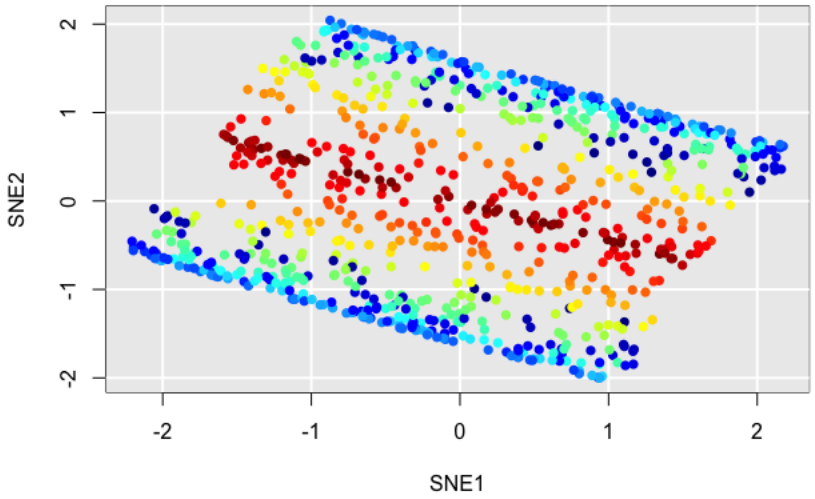
6.1.3 Curved Clusters

Here we investigate a simulated data set containing five three dimensional clusters sampled from Gaussian distributions which are situated on a one dimensional c-shaped structure, seen in Fig. 23. The aim is to project to data down to dimension one while preserving the cluster structure.

Results are shown in Fig. 24. The top left figure shows the t-SNE result with perplexity set equal to 50. The t-SNE algorithm provides the best reduction out of the algorithms compared, preserving and clearly distinguishing all clusters. The SNE result with perplexity 50, in the top right figure, shows that the algorithm fails to preserve the clusters by introducing overlap between them. This shows one of the advantages of the t-distribution kernel, which helps to more clearly separate clusters. The PCA result in the bottom figure is also not able to preserve the clusters, due to being limited to linear reductions.

The $R(K)$ values are shown in Fig. 22. The PCA and SNE algorithms perform similarly for all neighborhood sizes. The t-SNE algorithm outperforms PCA and SNE for all values of K except for large neighborhood sizes.

The $\overline{R}$ values are found in Table 4. The t-SNE algorithm resulted in the highest value, whereas SNE performed reported the lowest score.



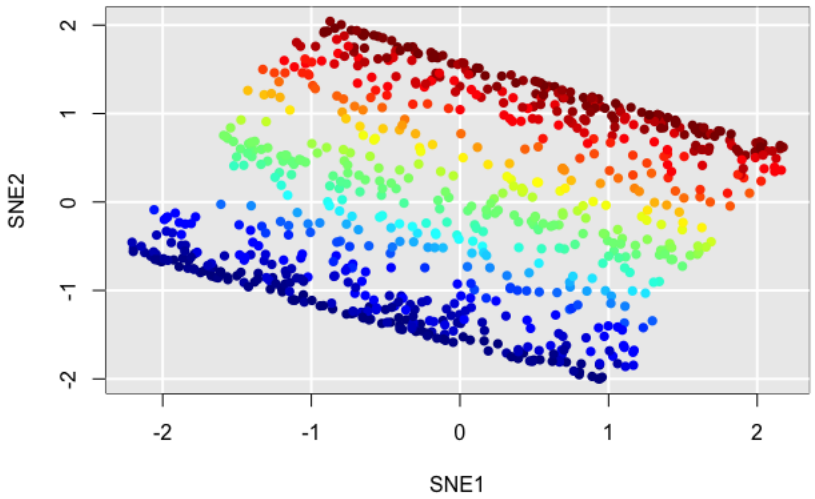


Figure 20: The figure shows the reduced S-Curve data set into two dimensions using the SNE algorithm. The left plot shows the result with perplexity 30 and the right plot shows the result with perplexity 50.

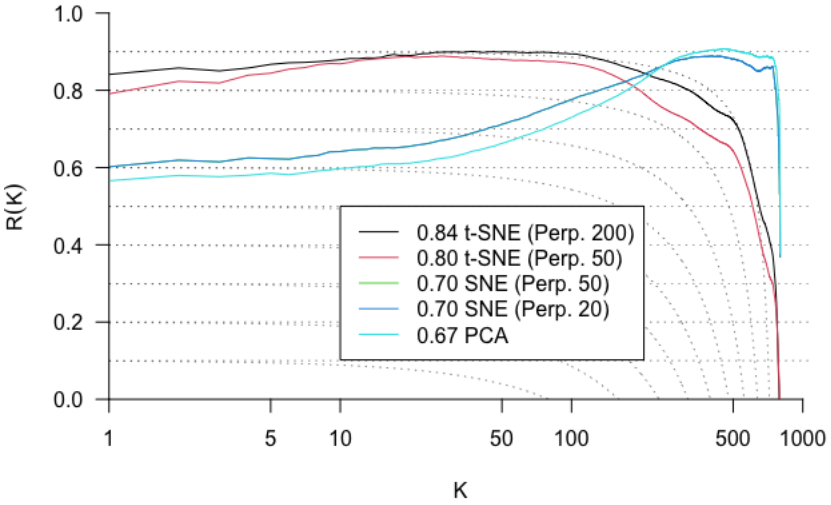


Figure 21: The figure shows R-curves for all of the tests conducted on the S-Curve data set. The higher the curve, the higher the preservation.

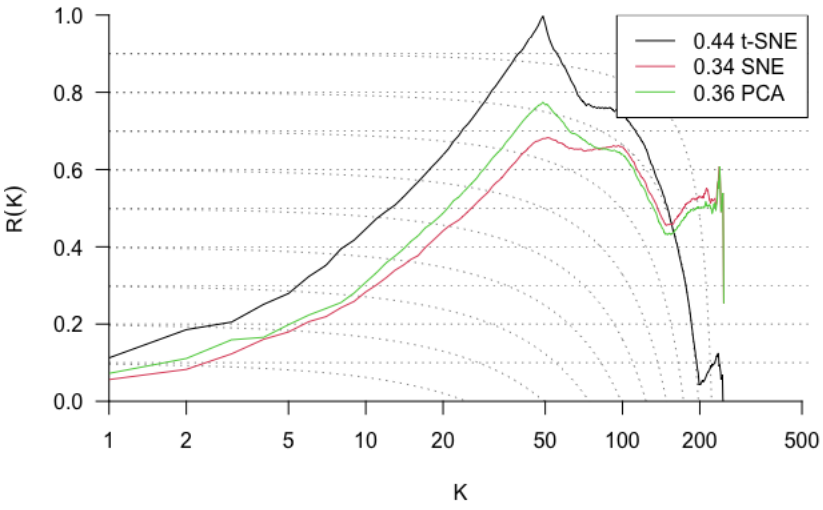


Figure 22: The figure shows R-curves for all of the tests conducted on the Curved Clusters data set. The higher the curve, the higher the preservation.

	$\overline{R}$
t-SNE (Perp. 50)	0.80494
t-SNE (Perp. 200)	0.83929
SNE (Perp. 20)	0.70484
SNE (Perp. 50)	0.70484
PCA	0.67466

Table 3: The table contains $\overline{R}$ values for each test case on the S-Curve data set. A higher score means better preservation of rank ordering.

	t-SNE	SNE	PCA
$\overline{R}$	0.43626	0.33777	0.36265

Table 4: The table contains $\overline{R}$ scores for each test case on the Curved Clusters data set. A higher score means better preservation of rank ordering.

6.1.4 MNIST Data Set

In this section the MNIST data set is tested. The data set was originally introduced by [12] and consists of 28×28 pixels depicting handwritten digits between 0 and 9. An example of an image from this data set is shown in Fig. 25.

Since the dimension of the vector representing each image is 784 it is advantageous to reduce the dimension as much as possible in order to see the cluster structure of the data, where images of the same digit is expected to fall in the same cluster. Before applying the t-SNE and SNE algorithms PCA algorithm was used and the first 422 principal components were kept, as these preserved more than 99% of the variation in the data, in order reduce complexity. Fig. 26 shows the result of the t-SNE algorithm with perplexity set to 40. The algorithm manages to preserve most of the clusters with pictures containing the same digit belonging to the same cluster. The SNE algorithm with perplexity 40, results shown in Fig. 27, and the PCA algorithm, with results shown in Fig. 28, were unable to preserve the cluster structure.

	t-SNE	SNE	PCA
$\overline{R}$	0.41163	-0.00092	0.14852

Table 5: The table contains $\overline{R}$ scores for each test case on the MNIST data set. A higher score means better preservation of rank ordering.

The $\overline{R}$ scores are summarised in Table 5. According to this measure t-SNE clearly outperforms SNE and PCA. SNE interestingly performs the worst on this data set with a negative value of $\overline{R}$ which means that the quality of the embedding is worse than a random embedding according to this measure. Fig. 29 contains the $R(K)$ values for each neighborhood size K for each test.

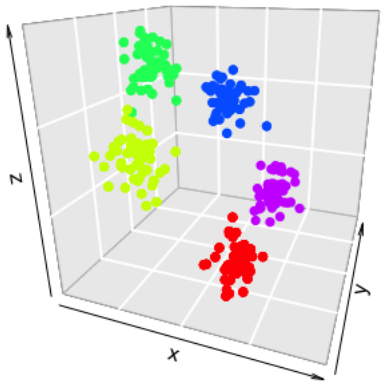


Figure 23: Figure on the left shows five clusters in three dimensions sampled from Gaussian distributions, arranged on a one dimensional c-shape.

The t-SNE algorithm resulted higher R values for most values of neighborhood sizes K. PCA scores slightly higher for large values of K.

6.2 Artifacts

In this section some of the potential artifacts of t-SNE are illustrated by simulated examples.

6.2.1 t-SNE will overfit the data if perplexity is set too low

A possible artifact of the t-SNE algorithm is that it may find structure in random noise when the perplexity parameter is set too low. To illustrate such an example 500 points where sampled from a 50 dimensional standard Gaussian distribution with uncorrelated components. The data was then reduced to 2 dimensions with the t-SNE algorithm with perplexity set equal to 5. The result is shown in Fig. 30. It is not possible to see the Gaussian structure with denser points towards the center. Also, sub-clusters are formed, which may mislead the user to interpret random noise as structure. Therefore the perplexity should not be set too low, as one might be misled to see structure where there is none.

In contrast, when perplexity is too high the distribution of points will appear uniform and no structure will be preserved. This can be seen in the same figure, where perplexity was set equal to 499 in the bottom right plot.

6.2.2 Size of clusters may be misinterpreted

The size of a cluster cannot be interpreted as representing the original size of the cluster. Rather, it corresponds to the original density of the cluster.

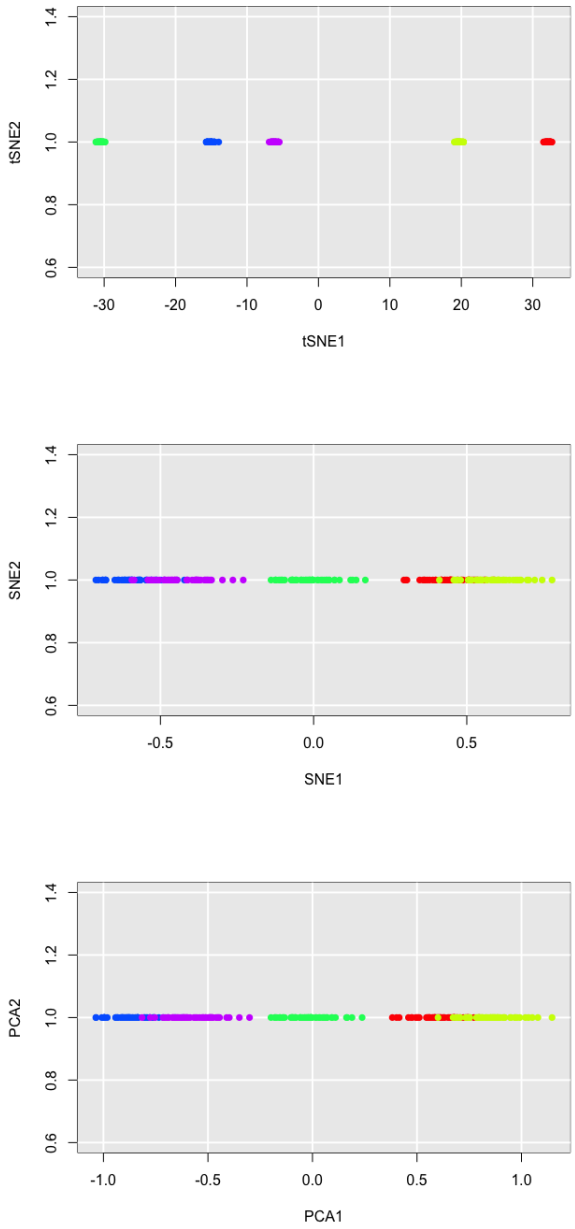


Figure 24: The figure shows the reduced data set of five Gaussian clusters on a one dimensional curved c-shape structure in three dimensions. The data is reduced to dimension 1. The top plot shows the t-SNE result with perplexity set to 50. The middle figure shows the SNE result with perplexity equal to 50. The bottom plot shows the data projected to the first principal component of the data set.

Figure 25: The figure shows an example of a handwritten digit image of the number 5 from the MNIST data set.

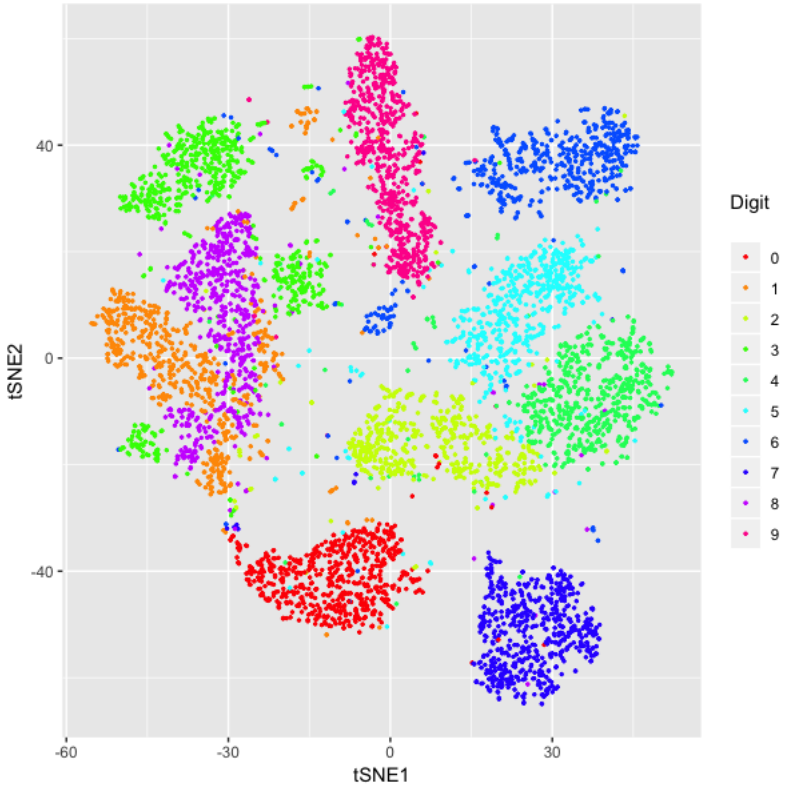


Figure 26: The figure shows the reduction to dimension 2 of the MNIST data set consisting of pixel values corresponding to pictures of handwritten digits. The figure shows the reduction using the t-SNE algorithm with perplexity set to 40. The algorithm was applied to the first 422 principal components of the data to reduce complexity.

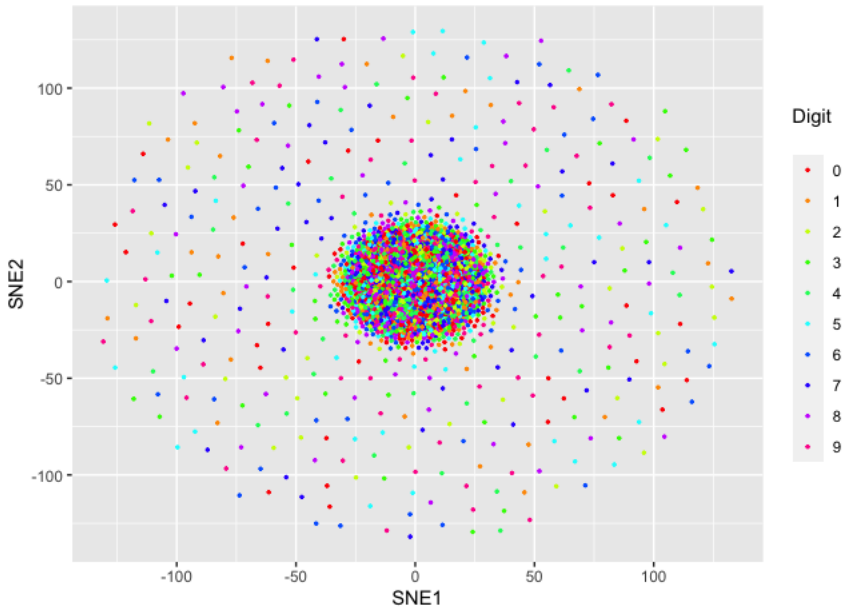


Figure 27: The figure shows the reduction to dimension 2 of the MNIST data set consisting of pixel values corresponding to pictures of handwritten digits. The figure shows the reduction using the SNE algorithm with perplexity set to 40. The algorithm was applied to the first 422 principal components of the data to reduce complexity.

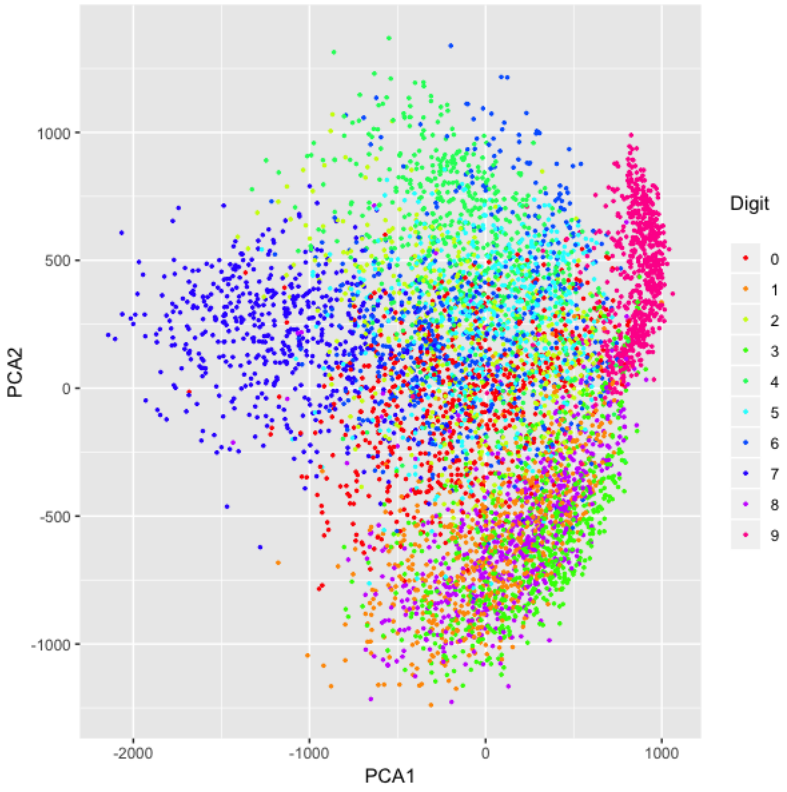


Figure 28: The figure shows the reduction to dimension 2 of the MNIST data set consisting of pixel values corresponding to pictures of handwritten digits. The picture shows the first 2 principal components of 6000 images.

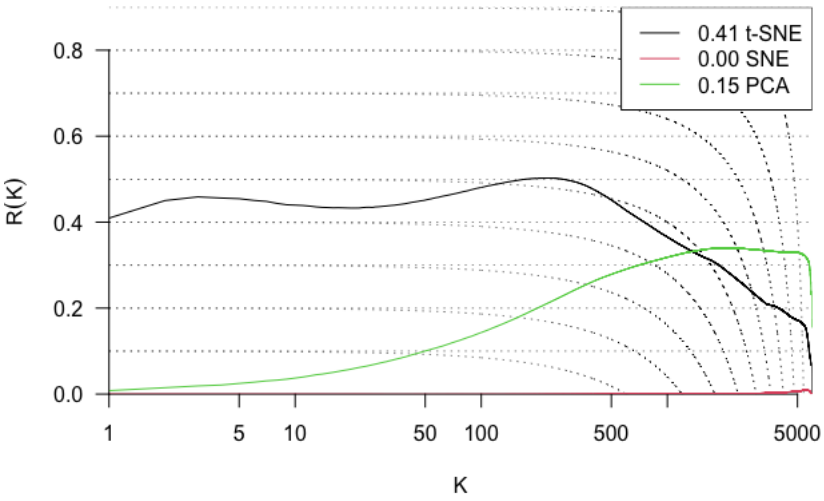


Figure 29: The figure shows R-curves for all of the tests conducted on the MNIST data set. The higher the curve, the higher the preservation.

A dense cluster will be modelled by a larger cluster and sparse cluster by a smaller cluster, in order to make them equally dense. Fig. 31 shows an example with data simulated from two Gaussian distributions with same variance but different means. Thus the data consists of two clusters with equal size and unequal number of points. A similar example is found in Fig. 32 which shows the reduction of a data set consisting of two clusters with equal number of points but different sizes. The size of the larger sparse cluster gets reduced, and the dense smaller cluster gets increased, so that they appear equally sized in the t-SNE map. This is an implication of introducing the perplexity parameter in order to select the variances in the Gaussian similarities.

6.2.3 Outliers will not remain outliers

An artifact of the t-SNE algorithm is that it will not preserve outliers. Since the variances are chosen so as to make the perplexity equal for the probability distributions of all points, the variance parameter will be chosen very large for an outlier. Therefore, the points otherwise distant to the outlier would be counted as close. This fact, combined with the low cost of error in modelling large distances, results in that the outlier is placed closer other clusters of points, making it difficult to recognize it as an outlier in the embedding. If x_i is an outlier and x_j its nearest neighbor, then if the same σ were to be used for all

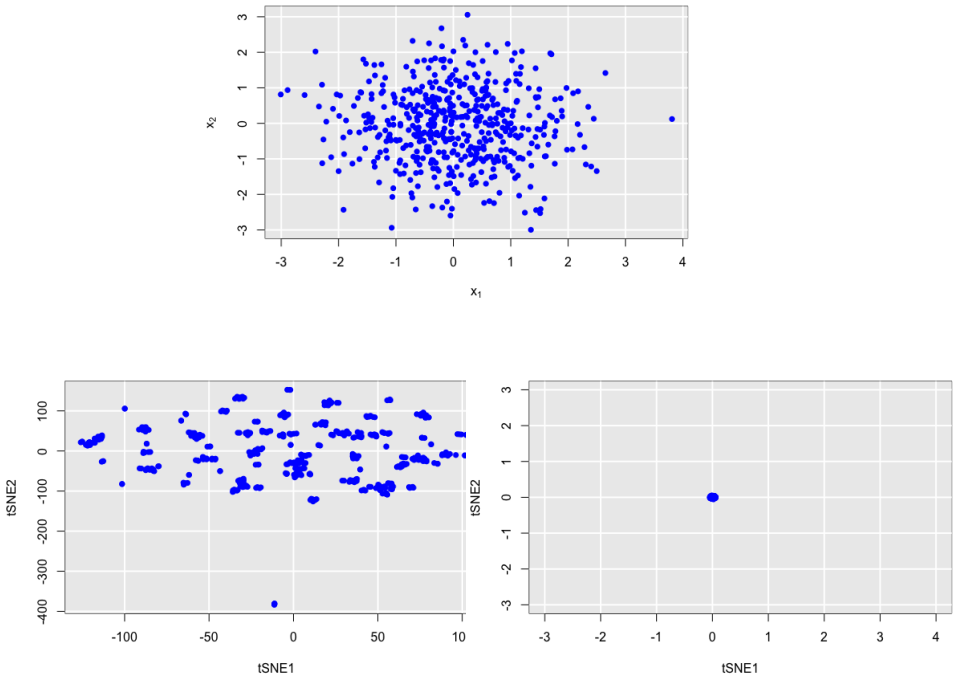


Figure 30: To the right the first two coordinates of 500 data points sampled from a 50 dimensional Gaussian distribution with independent zero mean and unit variance components. The bottom left is the result of the t-SNE algorithm when the dimension is reduced to 2 and perplexity is set equal to 5. The bottom right picture shows the result when perplexity was set to 499.

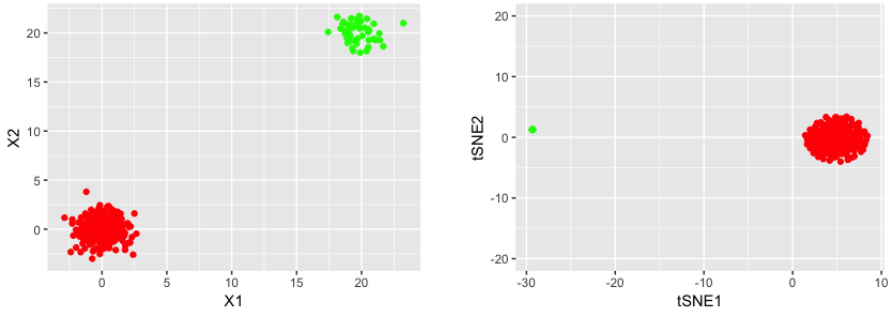


Figure 31: The figure on the left shows the first two coordinates of the original 50 dimensional data set. The figure on the right shows the t-SNE projection.

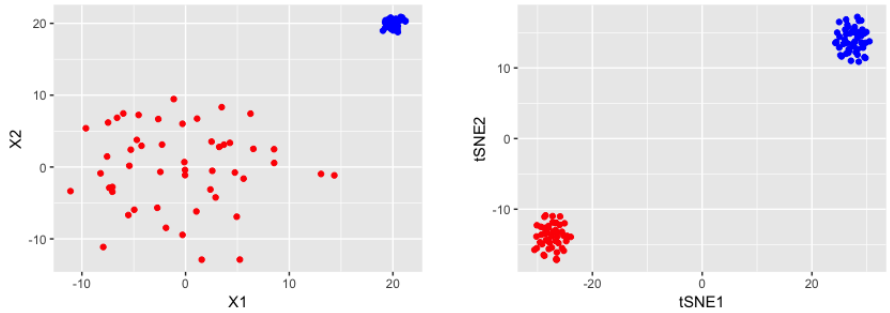


Figure 32: The left figure depicts the first coordinates of a data set with two clusters of points, with different densities. The right figure shows the tSNE projection into two dimensions.

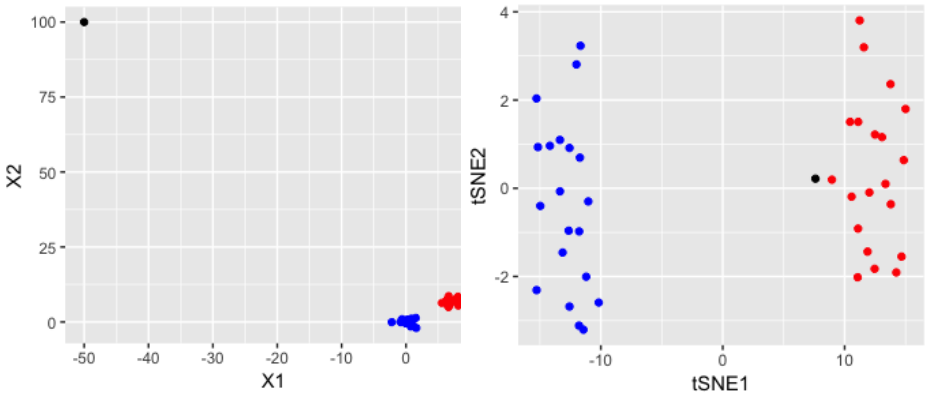


Figure 33: To the right is the original 50 dimensional data, displayed with the first two coordinates. To the left is the result of the t-SNE reduction to two dimensions. Perplexity was set equal to 10.

points

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N} \approx \frac{0+1}{2N} = \frac{1}{2N}.$$

When σ is increased, this probability will increase. If a data point has high distance to every other data point, the variance parameter corresponding to that point will increase to match the number of points located in more dense regions. Hence the outlier will be modelled as being much closer to the remaining points than it actually is. An example of this phenomenon is shown in Fig. 33.

6.2.4 Global distances may not be preserved

Since the t-SNE algorithm is designed to prioritize modelling local structure, it may result in artifacts in modelling of the global structure. This is partly due to

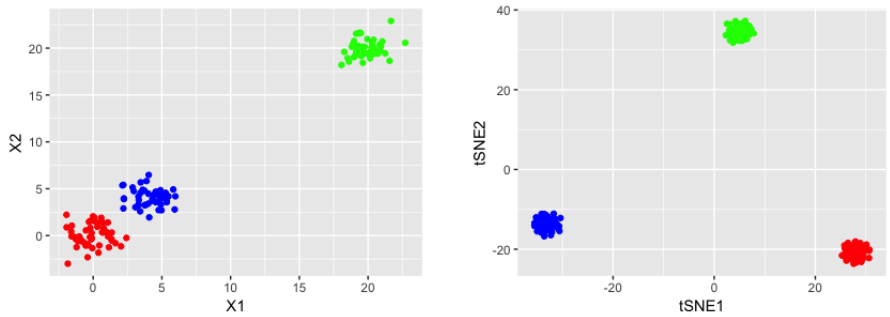


Figure 34: The plot on the right shows the first two coordinates of a 50 dimensional data set containing three clusters. The plot on the left shows the t-SNE projection.

the choice of using a Gaussian kernel with squared Euclidean distances, since the similarity will decrease quickly to zero for large distances, and mainly because the Kullback-Leibler cost function is almost unaffected by errors in modelling large pairwise distances. Fig. 34 shows an example with three clusters of points, where two of them are close, whereas the third is far away. The resulting map models the clusters as being almost equidistant. Hence care needs to be taken when drawing conclusion about global structure from the map.

7 Discussion

In this section the main findings and conclusion of this thesis are summarised and discussed. We also provide some outlooks on possible future research.

7.1 Conclusions

In this thesis the topic of dimension reduction, and in particular the t-SNE, PCA and SNE algorithms have been explored. In Section 4 the theory behind the SNE and t-SNE algorithms were covered. Then, in Section 5 some evaluation criteria were introduced. These algorithms were then applied to test data sets in Section 6.1. The t-SNE algorithm outperformes both SNE and PCA on most of the test cases. A weakness with t-SNE however is its failure of preserving global structure, due to the asymmetric cost function used as well as the mismatch between the similarity kernels used.

Other artifacts that were found was that distance between clusters may be distorted, the size of clusters can mislead, outliers may not remain outliers and the algorithm may find structures in random noise. The results often highly depend on choosing the right perplexity value, and therefore there is a need to develop a method for optimally choosing this parameter.

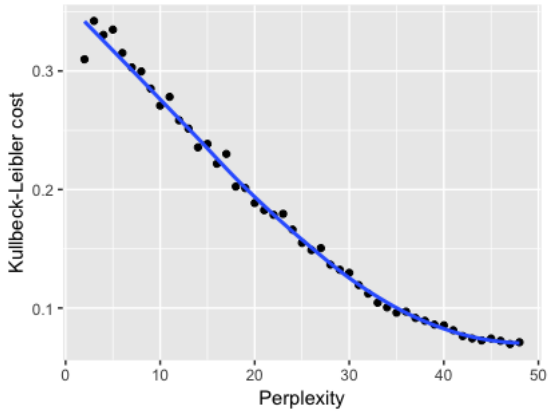


Figure 35: The figure illustrates the Kullback-Leibler divergence cost as a function of perplexity after 5000 iterations on the IRIS data set.

7.2 Outlooks and Open Questions

Many variations of SNE have been suggested in the literature to combat with the limitations of t-SNE. Different cost functions, which does not have the asymmetric property of the Kullback-Leibler divergence have been suggested, that improves the preservation of global structure [9, 15, 7, 1].

The problem of choosing an optimal value for perplexity parameter remains. In this thesis the perplexity was mainly chosen subjectively in the experiments of Section 6, and no standardised method was used. The Gaussian kernel is approximately equal to the t-distribution kernel for large perplexity values, as for σ large

$$e^{-\frac{\Delta^2}{2\sigma^2}} \approx \frac{1}{1+\frac{\Delta^2}{2\sigma^2}} \approx 1.$$

Hence the similarities in the high dimensional space will resemble the similarities in the low dimensional space more closely, and the distribution will be modelled by a uniform distribution. The optimal Kullback-Leibler cost function value will therefore decrease monotonically as perplexity decreases as can be seen in Fig. 35, which shows the Kullback-Leibler divergence after 5000 iterations of the IRIS data set. Evaluating in this way would therefore result in representing the data by a uniform distribution and hence no structure would be preserved, as the optimal perplexity would be equal to the number of points. This corresponds to representing the high dimensional data by a uniform distribution, and thus no structure is preserved by the probability distribution representing it. A low dimensional map can easily be found by setting all map points equal to each other, which would lead to a Kullback-Leibler cost of 0. Therefore, a method for choosing an optimal perplexity value is needed. We do not find much literature dealing with this problem, and is often chosen subjectively, as the default value in applications or by heuristical arguments [10]. A possible approach for deciding

on the right perplexity value is to look at the $\overline{R}$ score described in Section 5, and look for a minimum value.

The mathematical foundation of Stochastic Neighbor Embedding and its variants is still not fully understood. Therefore a possible future research direction is to further explore this. For example [17] attempts this.

References

[1] Andrés Marino Álvarez-Meza et al. "Kernel-based dimensionality reduction using Renyi's α-entropy measures of similarity". In: *Neurocomputing* 222 (2017), pp. 36–46.

[2] Jérémie Bouttier, Philippe Di Francesco, and Emmanuel Guitter. "Geodesic distance in planar graphs". In: *Nuclear physics B* 663.3 (2003), pp. 535–567.

[3] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[4] Michael AA Cox and Trevor F Cox. "Multidimensional scaling". In: *Handbook of data visualization*. Springer, 2008, pp. 315–347.

[5] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York, 2001.

[6] Antonio Gracia et al. "A methodology to compare dimensionality reduction algorithms in terms of loss of quality". In: *Information Sciences* 270 (2014), pp. 1–27.

[7] Kenneth D Harris et al. "Classes and continua of hippocampal CA1 inhibitory neurons revealed by single-cell transcriptomics". In: *PLoS biology* 16.6 (2018), e2006387.

[8] Geoffrey E Hinton and Sam T Roweis. "Stochastic neighbor embedding". In: *Advances in neural information processing systems*. 2003, pp. 857–864.

[9] Anna Klimovskaia et al. "Poincaré maps for analyzing complex hierarchies in single-cell data". In: *Nature Communications* 11.1 (2020), pp. 1–9.

[10] Dmitry Kobak and Philipp Berens. "The art of using t-SNE for single-cell transcriptomics". In: *Nature communications* 10.1 (2019), pp. 1–14.

[11] Ashwinikumar Kulkarni et al. "Beyond bulk: a review of single cell transcriptomics methodologies and applications". In: *Current opinion in biotechnology* 58 (2019), pp. 129–136.

[12] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[13] John A Lee, Diego H Peluffo-Ordóñez, and Michel Verleysen. "Multi-scale similarities in stochastic neighbour embedding: Reducing dimensionality while preserving both local and global structure". In: *Neurocomputing* 169 (2015), pp. 246–261.

[14] John A Lee and Michel Verleysen. *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

[15] John A Lee et al. "Type 1 and 2 mixtures of Kullback–Leibler divergences as cost functions in dimensionality reduction based on similarity preservation". In: *Neurocomputing* 112 (2013), pp. 92–108.

[16] John Lee and Michel Verleysen. "Quality assessment of nonlinear dimensionality reduction based on K-ary neighborhoods". In: *New Challenges for Feature Selection in Data Mining and Knowledge Discovery*. 2008, pp. 21–35.

[17] George C Linderman and Stefan Steinerberger. "Clustering with t-SNE, provably". In: *SIAM Journal on Mathematics of Data Science* 1.2 (2019), pp. 313–332.

[18] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[19] Ahmed Mahfouz et al. "Visualizing the spatial gene expression organization in the brain through non-linear similarity embeddings". In: *Methods* 73 (2015), pp. 79–89.

[20] Rahul Paul and Stephan K Chalup. "A study on validating non-linear dimensionality reduction using persistent homology". In: *Pattern Recognition Letters* 100 (2017), pp. 160–166.

[21] Karl Pearson. "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572.

[22] John W Sammon. "A nonlinear mapping for data structure analysis". In: *IEEE Transactions on computers* 100.5 (1969), pp. 401–409.

[23] Claude E Shannon. "A mathematical theory of communication". In: *Bell system technical journal* 27.3 (1948), pp. 379–423.

[24] W Nick Street, William H Wolberg, and Olvi L Mangasarian. "Nuclear feature extraction for breast tumor diagnosis". In: *Biomedical image processing and biomedical visualization*. Vol. 1905. International Society for Optics and Photonics. 1993, pp. 861–870.

[25] Joshua B Tenenbaum, Vin De Silva, and John C Langford. "A global geometric framework for nonlinear dimensionality reduction". In: *science* 290.5500 (2000), pp. 2319–2323.

[26] Laurens Van Der Maaten, Eric Postma, and Jaap Van den Herik. "Dimensionality reduction: a comparative". In: *J Mach Learn Res* 10.66-71 (2009), p. 13.

[27] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. "How to Use t-SNE Effectively". In: *Distill* (2016). DOI: 10.23915/distill.00002. URL: http://distill.pub/2016/misread-tsne.