Ukrainian Catholic University

Faculty of Applied Sciences

Data Science Master Programme

Visualization of high-dimensional data using t-SNE algorithm

Linear Algebra final project report

Authors:
Serhii Brodiuk
Andrii Yurkiv

24 January 2019

Abstract

In this report we describe our implementation of t-SNE algorithm for visualization of multidimensional data. We compare its performance and visualization capabilities with other classic visualization algorithms and discuss its strengths and weaknesses.

1 Introduction

Data visualization is one of the most important parts of applied data analysis nowadays. Without proper visualization it's hard, or sometimes even impossible to interpret the data in the appropriate way or make reasonable inferences regarding its structure. Dealing with data sets that contain more than 3 attributes start causing problems, because standard visualization techniques does not apply to the data in 4-dimensional space. For humans, it's even hard to perceive graphical information in 3-dimensional space, that's why the majority of classical techniques of data visualization are restricted to two dimensions.

It should be noted that one of the main purposes of visualization of high-dimensional data sets is to determine clusters inside the data which may be an important basement for further data analysis.

2 Motivation

Modern data often contains hundreds or even thousands of features. And classical visualization approaches cannot be used to determine patterns or clusters in it. This is the reason why so many different techniques for visualization high-dimensional data have been proposed. Majority of these techniques merely provide tools to display multidimensional data in two or three-dimensional form and do not give any inferences regarding the inner structure of the data. This severely limits the applicability of those techniques to real-world data, which sometimes contain too much features to reduce them nicely.

Another approach to this problem lies in the field of dimensionality reduction. We can apply classical plotting methods (for example, scatter plots) for the data obtained by decreasing the dimension of the original data set. Dimensionality reduction techniques differ substantially from visualization techniques because they are aimed not to make data more visually appealing or understandable, but to preserve as much of the significant structure of the high-dimensional data as possible in its low-dimensional representation [1].

3 Related work

As we said, there exists two main approaches to the problem of visualization of high-dimentional data. First is to use some strictly visualization technique for high-dimentional data and second is to apply dimentionality reduction and then plot generated attributes

in two or three dimentions using some classical visualization methods [1].

Some famous techniques for visualization of high-dimentional data are: icono- graphic displays such as Chernoff faces [4], pixel-based techniques [5], and those that represent the dimentions in the data as vertices in a graph.

There are much more dimentionality reduction techniques, both linear and non-linear ones. The most important linear dimentionality reduction techniques are PCA (Principal Component Analysis)[6] and classical multidimentional scal- ing [7]. Those two focus on keeping the low-dimentional representation of dis- similar datapoints far apart. Some famous non-linear dimentionality reduction techniques are: Locally Linear Embedding [8], Stochastic Neighbor Embedding [3], Isomap [9], Laplacian Eigenmaps [10] and others.

4 Problem setting

The most important goal of visualization of multidimensional data is to correctly identify clusters of similar data points. This is crucial when we are working on unsupervised learning problem and need to know for sure what number of distinct clusters can our data set be divided in without losing valuable information about inner structure of our data.

As we will see in this report, many classical visualization algorithms correctly identify the relationships between data points and position similar ones together. But at the same time from visualizations those algorithms provide it's hard to tell where one class of points ends and other begins (boundaries between classes are not well-defined). And while this is not a problem for small data sets, for large ones it can cause serious obstacles for correct definition of of the number and location of data clusters.

t-SNE (t-distributed Stochastic Neighbor Embedding) algorithm is aimed to overcome those limitations by identifying the relevant patterns using the approach in which similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability [1].

5 Approach to Solution

Before we start to discuss a solution (implementation) of t-SNE, let's have a little overview about classic SNE [3].
With high-dimensional input space X, we need to transform datapoints in output (usually 2-3 dimensions) space Y. The goal is not to lose much structure or patterns in our data. SNE approach is to start with the transformation of pairwise Euclidean distance between datapoints into conditional probabilities (similarities):

$$p_{j|i} = \frac{exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-||x_i - x_k||^2/2\sigma_i^2)}(1) \tag{1}$$

In other words, this formula shows how close x_j point is to x_i one with the Gaussian (variance of σ) near x_j datapoint. σ will be different for each point with logic, that with high density variance become less. For that, it is used metric of perplexity:

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

$H(P_i)$ is the Shannon entropy of P_i in bits:

$$H(P_i) = -\sum_j p_{j|i} log_2 p_{j|i} \tag{2}$$

Perplexity could be interpret as "effective count of neighbours for x_i". It is a hyper-parameter in t-SNE algorithm. σ calculated due to binary search for each x_i, x_j pair (the perplexity increases monotonically with the variance).
For output, target datapoints Y there are similarity with the variance of $\sigma = 1/\sqrt{2}$:

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

Obvious, $p_{j|i}$ and $q_{j|i}$ should be equivalent to each other with perfect converting. The Kullback- Leibler divergence (the cross-entropy up to an additive constant here) - metric to determine cost function (as sum) and then try to minimize it using gradient descent method.

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}} \tag{3}$$

And the gradient (quite simple):

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

Y are initiated randomly from an isotropic Gaussian with small variance that is centered around the origin. It makes SNE/t-SNE algorithms non deterministic (their final solution is stochastic, because of the initial random configuration of the map points). A large momentum term is added to the gradient for faster optimization and to avoid local minima. Final gradient update:

$$Y^{(t)} = Y^{(t-1)} + \eta \frac{\partial C}{\partial Y} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)})$$

t - calculating iteration, η - the learning rate (size of "step"), $\alpha(t)$ - the momentum at iteration t.

Now, we have an overview of SNE algorithm. By using classic SNE, we can obtain good results. But there might be a problem with cost function optimization and crowding problem. Let's see how t-SNE can handle them (or even try to).

6 Solution

First, make symmetry properties from conditional probabilities: p_{ii} and $q_{ii} = 0, p_{ij} = p_{ji}, q_{ij} = q_{ji}$ for any i, j, and p_{ij}:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

n - number of datapoints. Now, new cost function and corresponding gradient:

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

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)$$

Also, if start use Student t-distribution it will be less crowding problem (due to heavy tails). $t = 1$ (a Cauchy distribution):

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l} (1 + ||y_k - y_l||^2)^{-1}} \tag{4}$$

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1} \tag{5}$$

If we speak about embeddings structure: SNE doesn't push enough away datapoints in low-dimensional space which have big distance in high-dimensional, while t-SNE does.

Data: dataset $X = x_1, x_2, \ldots, x_n$
cost function parameters: perplexity $Perp$
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$
Result: low-dimensional data representation $Y(T) = y_1, y_2, \ldots, y_n$
begin
  compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (eq. 1)
  set $p_{ij} = (p_{j|i} + p_{i|j})/2n$
  sample initial solution $Y(0) = y_1, y_2, \ldots, y_n$ from $N(0, 10^{-4}I)$
  for $t \leftarrow 1$ **to** T **do**
    compute low-dimensional affinities q_{ij} (eq. 4)
    compute gradient $\frac{\partial C}{\partial y}$ (eq. 5)
    set $Y(t) = Y(t-1) + \eta \frac{\partial C}{\partial y} + \alpha(t)(Y(t-1) - Y(t-2))$
  end
end

Algorithm 1: Whole simple algorithm [1]

There are also two hints (tricks) to help achieve better results. "early compression" and "early exaggeration" [1].

To have a clear understanding, we have implemented t-SNE with Python and explore how it could be used in practice. Here link to our work repository:

`https://github.com/andreyurkiv/la-tSNE-final-project`

7 Evaluation

Since t-SNE is primarily visualization technique, in order to evaluate its performance and visualizational capacity we need to compare visualization produced by t-SNE with the ones obtained using other non-parametric visualization techniques for multidimensional data, such as Multidimensional scaling [7] using SMACOF (Scaling by Majorizing a Complicated Function) algorithm, Isomap [9], Locally Linear Embedding [8], Principal Components Analysis [6] and Laplacian Eigenmaps [10]. PCA is not usually used for visualization, but we decided that it would be a good idea to show how the visualizations obtained from manifold learning algorithms differ from those obtained from classic dimensionality reduction technique.

7.1 Data sets

In our work we experimented with 2 different data sets: Olivetti faces data set and handwritten digits data set.

Olivetti faces dataset contains 400 64×64 grayscale images of faces (10 images for each of 40 subjects). Those images were taken at different time with different details varying, such as face expression, lighting and facial details (absence or presence of glasses). They are labeled according to the identity of the person depicted.

Handwritten digits data set consists of 1797 8×8 grayscale images of handwritten digits of 10 classes. There are nearly 180 instances per digit class in this data set.

7.2 Experimental setup

In the original t-SNE paper [1], the authors use PCA to reduce dimensionality before feeding the data into algorithms that transform it into two-dimensional representation. They are doing it for the purpose of computation efficiency and it absolutely makes sense when we apply those algorithms to large data sets. But since the data sets which we use in our experiments are relatively small, preliminar dimensionality reduction doesn't make much difference in terms of computational costs in our case, so we decided to pass this step up.

For each of these data sets, there is information about the class of each data point. Class information is only used for visualization purposes and is not considered during calculation of the spatial coordinates of the points in two-dimensional space. We decided to visualize the whole images instead of points on scatter plots because this kind of plot is

much more understandable and provides better means of evaluating how well the mapping preserves the similarities within each class and how accurately it separates one class from the other.

We used library utilities in order to demonstrate visualization capacity and performance of algorithms, with which we are comparing our implementation of t-SNE algorithm. We used `Isomap`, `LocallyLinearEmbedding`, `MDS` and `SpectralEmbedding` classes from package `sklearn.manifold` and `PCA` class from package `sklearn.decomposition`.

The cost function parameters we used in our experiments are listed in Table 1. Here *perplexity* represents the perplexity of the conditional probability distribution induced by the Gaussian kernel and *neighbors* represents the number of nearest neighbors employed in a neighborhood graph. Multidimensional scaling was performed using SMACOF algorithm with 4 initializations and 300 iterations and without any other explicit parameters. PCA uses regular library settings.

For each algorithm we visualized all data points present in the data set, that's why some of them look a bit crowded with too many images.

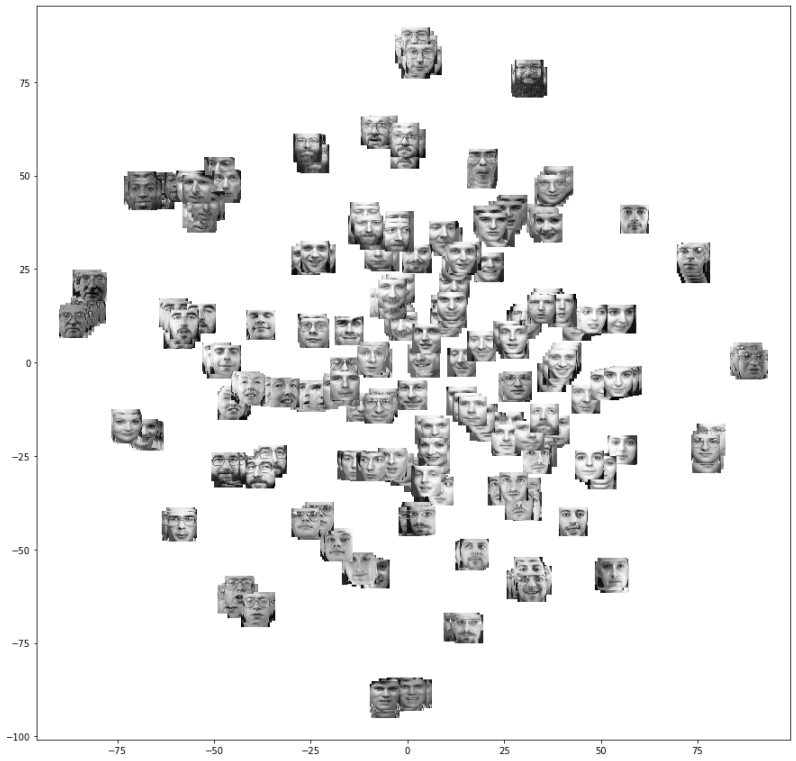
Technique	Parameters
t-SNE	*perplexity* = 5, 1000 iterations
MDS (SMACOF)	-
Isomap	*neighbors* = 5
LLE	*neighbors* = 12
PCA	-
Laplacian Eigenmaps	*neighbors* = 12

Table 1: Cost function parameters for the experiments.

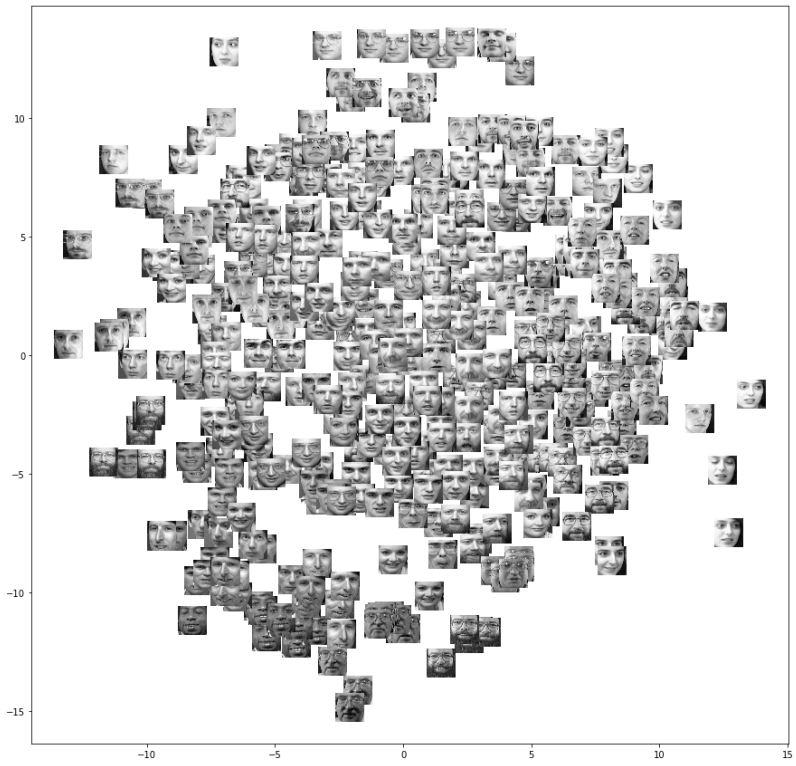
7.3 Results

In figures 1-3 we displayed the visualization produced by t-SNE, MDS, Isomap, LLE, PCA and Laplacian Eigenmaps algorithms applied to the Olivetti faces data set. Those result support our intuition that t-SNE strongly outperforms other techniques in visualization of multidimensional data set. For example, MDS generates ball-like figure, in which only few face classes are visually separated from the other images. Almost all algorithms have few face classes that are correctly clustered and located so that their group is distinct from the others. But at the same they have the majority of images overlapping, which makes it hard to distinguish the borders of the clusters. With such extreme overlaps as in plots produced by LLE or Laplacian Eigenmaps those graphs are almost useless, because if we hadn't class information we would fail to both number of distinct classes and their location.

On the contrary, t-SNE creates a map with relatively accurate separation of clusters. It locates images of the same person very close to each other, even if some distractions

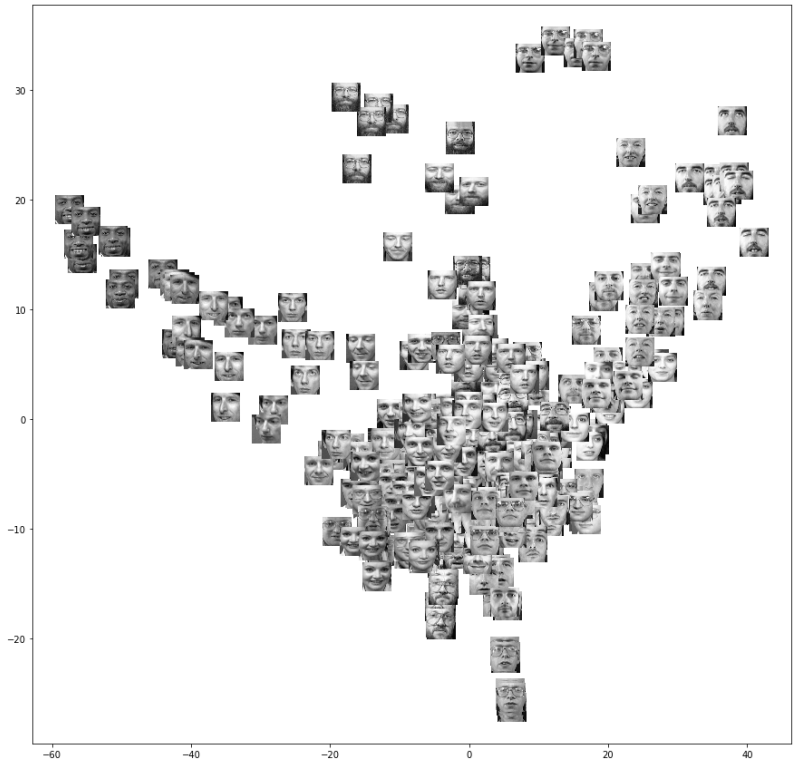


(a) using t-SNE

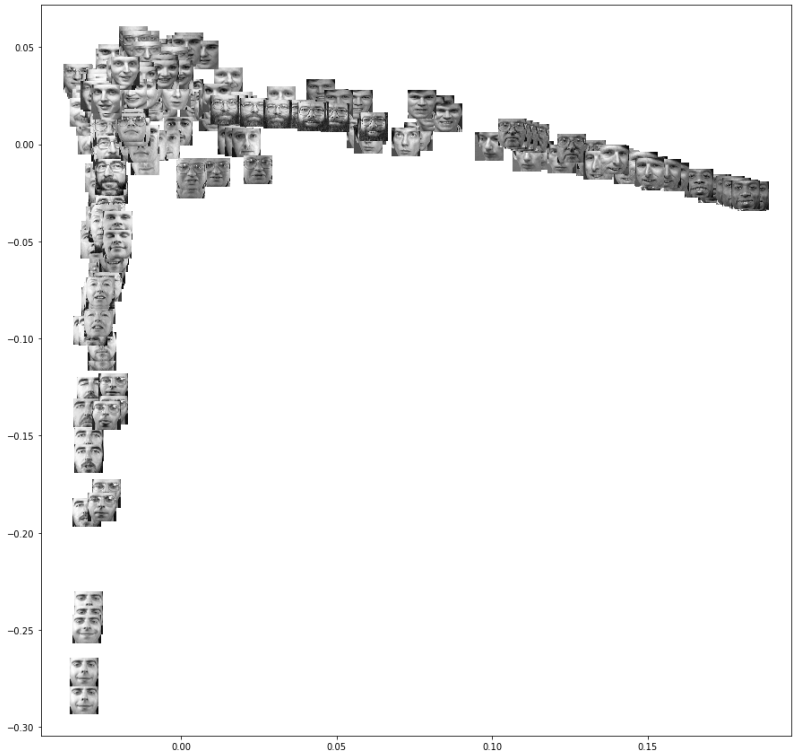


(b) using MDS

Figure 1: Visualization of Olivetti faces data set

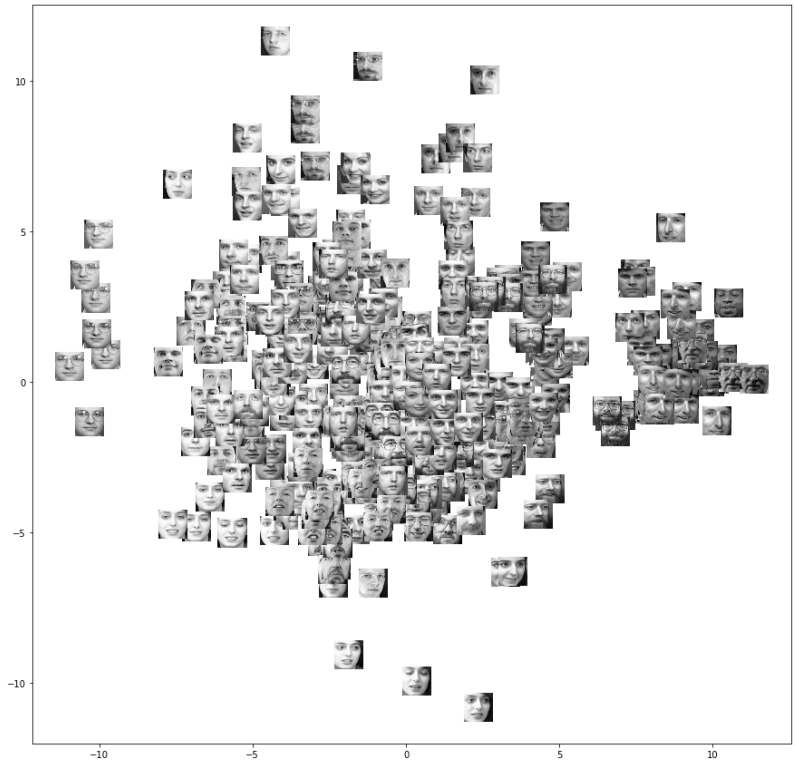


(a) using Isomap

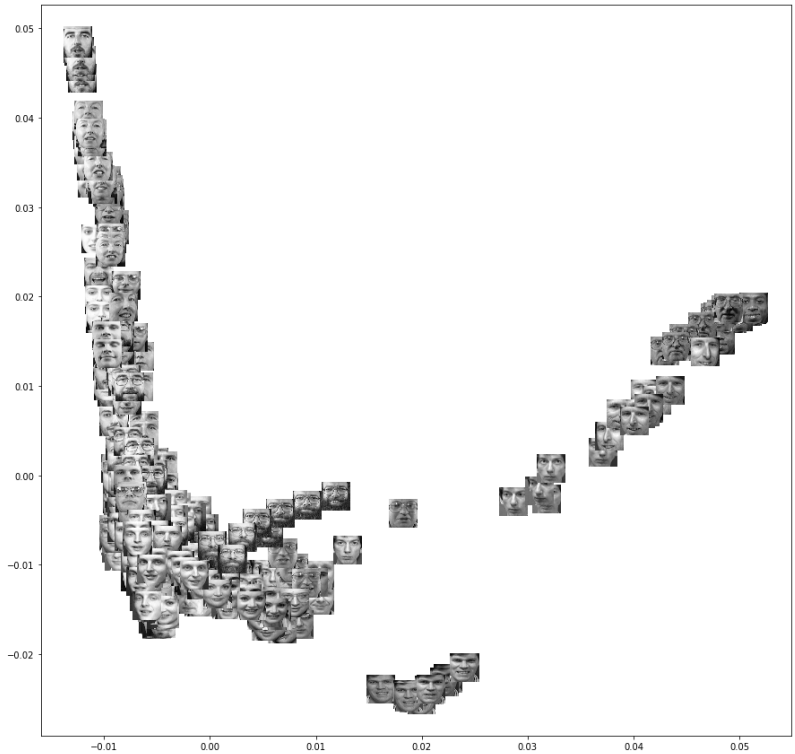


(b) using LLE

Figure 2: Visualization of Olivetti faces data set

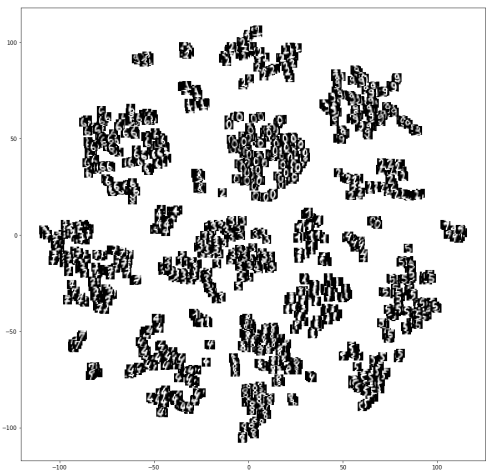


(a) using PCA

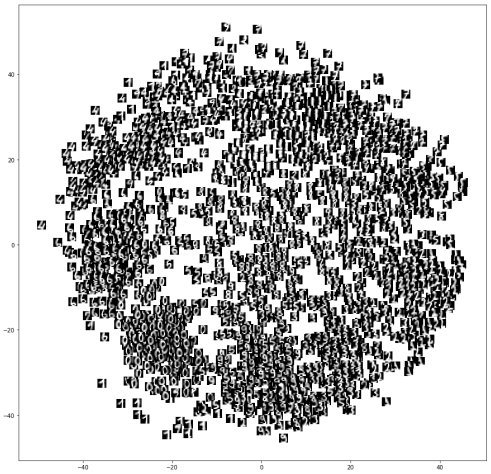


(b) using Laplacian Eigenmaps

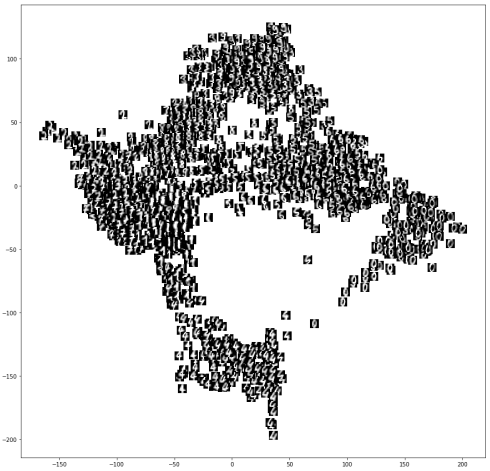
Figure 3: Visualization of Olivetti faces data set



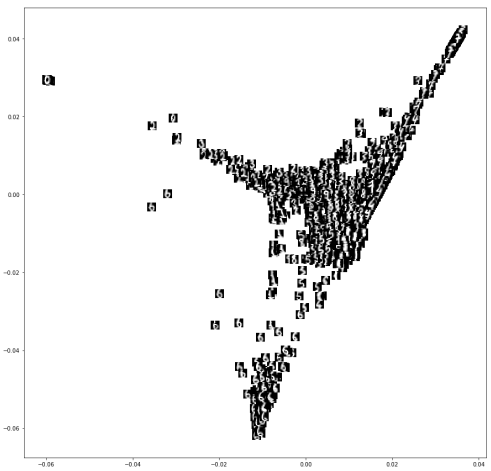
(a) using t-SNE



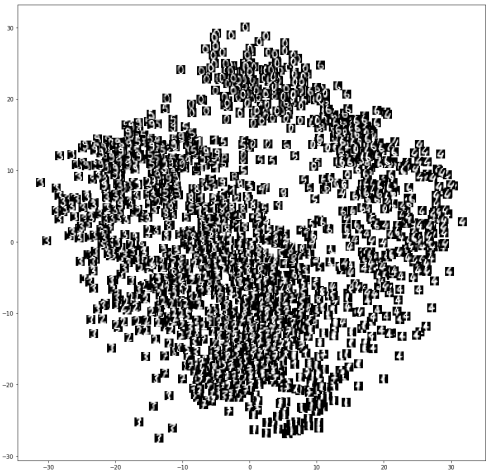
(b) using MDS



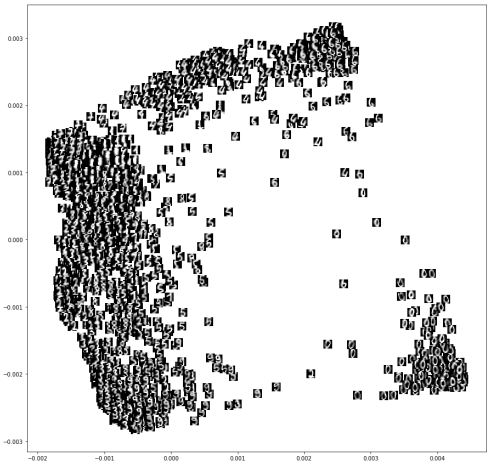
(c) using Isomap



(d) using LLE



(e) using PCA



(f) using Laplacian Eigenmaps

Figure 4: Visualizations of handwritten digits data set

are introduced (like glasses or face rotation). We can observe a number of sole points that don't belong to any cluster. Those are the faces with some specific facial expressions, which t-SNE is not able to capture. Besides it, there are some misclassified data points, but most of them correspond to distorted images which is hard to identify.

Figure 4 depicts the visualizations obtained after applyting t-SNE, MDS, Isomap, LLE, PCA and Laplacian Eigenmaps algorithms to handwritten digits data set. And again, t-SNE provides much better visualization capabilities comparing to other visualization techniques. LLE, Laplacian Eigenmaps and Isomap generate some angular structures in which the majority of points are overlapping and borders between classes are not distinguishable. MDS and PCA result in ball-like graphs, in which points of the same class lay close to each other, but in which there is no clear separation between clusters.

There is a number points, misclassified by t-SNE algorithm, or such that lay apart from clusters they truly belong to. This can be explained by the relatively small number of features (64), which make instances of different classes look extremely similar.

An important quality of any algorithm for visualization of multidimensional data is its speed. Since data analysis in general, and data visualization in particular, is highly iterative process, it is crucial to be able to perform it as fast as possible. The comparison of the visualization techniques in terms of its execution time is given in Table 2.

	t-SNE	MDS	Isomap	LLE	PCA	LE
Olivetti faces	**21.1858**	3.1424	1.2939	1.4976	0.1158	1.2543
Handwritten digits	**242.0297**	73.9359	1.7312	0.6649	0.02638	1.5366

Table 2: Execution time (in seconds) of discussed visualization algorithms.

Table 2 demonstrates important weakness of t-SNE algorithm - its quadratic computational and memory complexity. This quality makes it impossible to use standard version of this algorithm for data sets that contain more than, for instance, 10000 points. One obvious solution for this problem is to select a random subset of points and display them, but this approach has one serious drawback - we are no longer able to use the information that missing points provide about the underlying manifolds. The way t-SNE can be modified in order to use information about the entire data set is well described in the original t-SNE paper [1].

Another possible solution to this problem is reduce dimensionality of the data set before feeding it into t-SNE. This can be performed using PCA, or any other fast dimensionality reduction algorithm.

7.4 Discussion

Our experiments showed that t-SNE algorithm strongly outperforms other existing techniques for visualization of multidimensional data sets. But still, it has a number of

weaknesses, which we would like to discuss deeper.

First of all, it's not clear whether t-SNE can be used for general dimensionality reduction purposes when the data is reduced to more than 3 dimensions. The case is, that the heavy tails of Student t-distribution do not allow to extrapolate the behavior of this algorithm in 2 or 3 dimensions to higher ones. In high dimensions area under the heavy tails of Student t-distribution amounts to a large portion of the probability mass, which might lead to the situation when the local structure of the data is not preserved well.

Secondly, the cost function of t-SNE algorithm is non-convex, which means that several optimization parameters need to be chosen. The solution obtained depend on the optimization parameters choice and may be different each time it is run from the initial random configuration. But in general, it was shown, that even if the optimization process ends up in the local optima, it still provides a visualization good enough to prefer it instead of other visualization techniques. Besides it, the convexity of the cost function can be misleading, because it is often impossible to optimize the function for large data sets, so some appropriate approximation techniques are used instead.

The third weakness of t-SNE algorithm is its sensitivity to the intrinsic dimensionality of the data. This can cause the violation of local linearity assumptions on the manifold that t-SNE makes [1].

One more thing to note regarding t-SNE is that it cannot handle new, unseen data points and add it to the existing graph. While this limitation is common to many visualization algorithms, it still detains t-soSNE from being used in real-time systems.

8 Conclusions

The main aim of the visualization of multidimensional data is to represent properly the inner structure of the data. And it seems that t-SNE is an excellent choice for this kind of task.

In this work we have successfully implemented t-SNE algorithm and proved it superiority over other algorithms for visualization of multidimensional data by experimenting with two data sets and comparing visualizations produced during those experiments. It has been shown that besides its visualization power and the ability to reproduce both local and global patterns in the data and provide excellent clustering capabilities, it has a number of limitations and potential weaknesses, which in some sense limit its applicability in certain areas of data analysis.

References

[1] L. van der Maaten, G. Hinton. *Visualizing Data using t-SNE.* Journal of Machine Learning Research, 9 (2008), 2579-2605.

[2] Wikipedia contributors *t-distributed stochastic neighbor embedding.* `https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding`

[3] G. Hinton and S. Roweis *Stochastic Neighbor Embedding.* `https://papers.nips.cc/paper/2276-stochastic-neighbor-embedding.pdf`

[4] H. Chernoff *The Use of Faces to Represent Points in K-Dimensional Space Graphically* Journal of the American Statistical Association. American Statistical Association. 68 (342): 361–368.

[5] D.A. Keim *Designing pixel-oriented visualization techniques: Theory and applications* IEEE Transactions on Visualization and Computer Graphics, 6(1):59–78, 2000.

[6] H. Hotelling. *Analysis of a complex of statistical variables into principal components* Journal of Educational Psychology, 24:417–441, 1933.

[7] W.S. Torgerson *Multidimensional scaling I: Theory and method* Psychometrika, 17:401–419, 1952.

[8] S.T. Roweis, L.K. Saul *Nonlinear dimensionality reduction by Locally Linear Embedding* Science, 290(5500):2323–2326, 2000.

[9] J.B. Tenenbaum, V. de Silva, J.C. Langford *A global geometric framework for nonlinear dimensionality reduction* Science, 290(5500):2319–2323, 2000.

[10] M. Belkin, P. Niyogi *Laplacian Eigenmaps and spectral techniques for embedding and clustering* Advances in Neural Information Processing Systems, volume 14, 585–591, 2002.