Visualization of high-dimentional data using t-SNE algorithm

Serhii Brodiuk, Andrii Yurkiv

December 30, 2018

Description of the problem

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 inteprete the data in the appropriate way or make reasonable inferences regarding its structure. Dealing with datasets that contain more than 3 attributes start causing problems, because standard visualization techniques doesn't apply to the data in 4-dimentional space. For humans, it's even hard to perceive graphical information in 3-dimentional space, that's why the majority of classical techniqes of data visualization are restricted to two dimentions. It should be noted that one of the main purposes of visualization of high-dimentional datasets is to determine clusters inside the data which may be an important basement for further data analysis.

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-dimentional data have been proposed. Majority of these techniques merely provide tools to display multidimentional data in two or three-dimentional 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 dimentionality reduction. We can apply classical plotting methods (for example, scatterplots) for the data obtained by decreasing the dimention of the original dataset. Dimentionality 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-dimentional data as possible in its low-dimentional representation [1].

Possible approaches

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: iconographic displays such as Chernoff faces, pixel-based techniques, 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) and classical multidimentional scaling. Those two focus on keeping the low-dimentional representation of dissimilar datapoints far apart. Some famous non-linear dimentionality reduction techniques are: Sammon mapping, curvilinear component analysis, Stochastic Neighbor Embedding, Isomap, Laplacian Eigenmaps and others [1].

The technique we are going to discuss is called t-SNE (t-distributed Stochastic Neighbor Embedding). It is a modification of SNE technique which is capable of capturing both local and global structure of the data by converting a high-dimentional data set into matrix of pariwise similarities of its entries.

Advantages and disadvantages of t-SNE algorithm

Advantages:

- t-SNE creates visualizations that strongly outperform other techniques both in visual appearance and accuracy of clustering.
- This technique is able to preserve local structure of the data.
- Helps to identify relevant patterns by using the approach in which similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability [2].
- Solves the "crowding problem".
- Defines distance between entries in terms of neighbors connections, not input (high dimentional) distance between points.

Disadvantages:

- This method has quadratic time complexity, because it calculates pairwise distances between the entries of the dataset.
- Its cost function is not convex, hence several optimization parameters need to be chosen.
- t-SNE is not deterministic (final solution is stochastic, because of the initial random configuration of the map points).
- While this method maps local structure quite well, it doesn't always preserve the correct global picture.

- With input data dimension increasing, best linear combination in terms of variance explained is farther and farther from the real relation representation.
- Can produce "fake" patterns (due to "most points will repel each other" in some cases [3]);
- Cannot handle incomplete data (But still there are a few ways to deal with it during the preprocessing stage).
- Has complex process of hyperparameters tuning (for example, *perplexity*).
- Native t-SNE (without additional solution), does not map new, unseen samples.

Numerical realization of t-SNE algorithm

Given algorithm can be divided into three logical parts:

1. Calculation of perplexity and Gaussian kernel for the vector of current values of sigma.
2. Calculation of pairwise similarities p_{ij} in input (multidimensional) space using binary search for a given perplexity.
3. Calculation of pairwise similarities for output (display) space, cost function, and gradient.

Perplexity is a hyperparameter and authors of the original paper recommend to use value form 5 to 50.

Similarities are calculated due to Euclidean distance, but it is not necessarily (e. g. use cosine distance instead).

Gradiant in t-SNE (Student-t distribution) is more *constrict* for neighbour data points and it pushes away distant points even farther than the standard SNE algorithm (which uses Gaussian distribution).

References

[1] Laurens van der Maaten, Geoffrey 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] Clustering on the output of t-SNE `https://stats.stackexchange.com/questions/263539/clustering-on-the-output-of-t-sne`