Title: T-distributed stochastic neighbor embedding
URL: https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding
PageID: 39758474
Categories: Category:Dimension reduction, Category:Machine learning algorithms

---
Exploratory data analysis
Information design
Descriptive statistics
Inferential statistics
Statistical graphics
Plot
Data analysis
Infographic
Data science
Tamara Munzner
Ben Shneiderman
John Tukey
Edward Tufte
Simon Wardley
Hans Rosling
David McCandless
Kim Albrecht
Alexander Osterwalder
Ed Hawkins
Hadley Wickham
Leland Wilkinson
Mike Bostock
Jeffrey Heer
Ihab Ilyas
Line chart
Bar chart
Histogram
Scatter plot
Box plot
Pareto chart
Pie chart

Area chart

Tree map

Bubble chart

Stripe graphic

Control chart

Run chart

Stem-and-leaf display

Cartogram

Small multiple

Sparkline

Table

Marimekko chart

Data

Information

Big data

Database

Chartjunk

Visual perception

Regression analysis

Statistical model

Misleading graph

Topological data analysis

v

t

e

t-distributed stochastic neighbor embedding (t-SNE) is a statistical method for visualizing high-dimensional data by giving each datapoint a location in a two or three-dimensional map. It is based on Stochastic Neighbor Embedding originally developed by Geoffrey Hinton and Sam Roweis, [1] where Laurens van der Maaten and Hinton proposed the t -distributed variant. [2] It is a nonlinear dimensionality reduction technique for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions. Specifically, it models each high-dimensional object by a two- or three-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability.

The t-SNE algorithm comprises two main stages. First, t-SNE constructs a probability distribution over pairs of high-dimensional objects in such a way that similar objects are assigned a higher probability while dissimilar points are assigned a lower probability. Second, t-SNE defines a similar probability distribution over the points in the low-dimensional map, and it minimizes the Kullback–Leibler divergence (KL divergence) between the two distributions with respect to the locations of the points in the map. While the original algorithm uses the Euclidean distance between objects as the base of its similarity metric, this can be changed as appropriate. A Riemannian variant is UMAP .

t-SNE has been used for visualization in a wide range of applications, including genomics , computer security research, [3] natural language processing , music analysis , [4] cancer

research , [5] bioinformatics , [6] geological domain interpretation, [7] [8] [9] and biomedical signal processing. [10]

For a data set with n elements, t-SNE runs in O(n 2) time and requires O(n 2) space. [11]

Details

Given a set of N {\displaystyle N} high-dimensional objects x 1 , … , x N {\displaystyle \mathbf {x} _{1},\dots ,\mathbf {x} _{N}} , t-SNE first computes probabilities p i j {\displaystyle p_{ij}} that are proportional to the similarity of objects x i {\displaystyle \mathbf {x} _{i}} and x j {\displaystyle \mathbf {x} _{j}} , as follows.

For i ≠ j {\displaystyle i\neq j} , define

and set p i ■ i = 0 {\displaystyle p_{i\mid i}=0} .

Note the above denominator ensures ∑ j p j ■ i = 1 {\displaystyle \sum _{j}p_{j\mid i}=1} for all i {\displaystyle i} .

As van der Maaten and Hinton explained: "The similarity of datapoint x j {\displaystyle x_{j}} to datapoint x i {\displaystyle x_{i}} is the conditional probability, p j | i {\displaystyle p_{j|i}} , that x i {\displaystyle x_{i}} would pick x j {\displaystyle x_{j}} as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at x i {\displaystyle x_{i}} ." [2]

Now define

This is motivated because p i {\displaystyle p_{i}} and p j {\displaystyle p_{j}} from the N samples are estimated as 1/N, so the conditional probability can be written as p i ■ j = N p i j {\displaystyle p_{i\mid j}=Np_{ij}} and p j ■ i = N p j i {\displaystyle p_{j\mid i}=Np_{ji}} . Since p i j = p j i {\displaystyle p_{ij}=p_{ji}} , you can obtain previous formula.

Also note that p i i = 0 {\displaystyle p_{ii}=0} and ∑ i , j p i j = 1 {\displaystyle \sum _{i,j}p_{ij}=1} .

The bandwidth of the Gaussian kernels σ i {\displaystyle \sigma _{i}} is set in such a way that the entropy of the conditional distribution equals a predefined entropy using the bisection method . As a result, the bandwidth is adapted to the density of the data: smaller values of σ i {\displaystyle \sigma _{i}} are used in denser parts of the data space. The entropy increases with the perplexity of this distribution P i {\displaystyle P_{i}} ; this relation is seen as

where H (P i) {\displaystyle H(P_{i})} is the Shannon entropy H (P i) = − ∑ j p j | i log 2 ■ p j | i . {\displaystyle H(P_{i})=-\sum _{j}p_{j|i}\log _{2}p_{j|i}.}

The perplexity is a hand-chosen parameter of t-SNE, and as the authors state, "perplexity can be interpreted as a smooth measure of the effective number of neighbors. The performance of SNE is fairly robust to changes in the perplexity, and typical values are between 5 and 50.". [2]

Since the Gaussian kernel uses the Euclidean distance ■ x i − x j ■ {\displaystyle \lVert x_{i}-x_{j}\rVert } , it is affected by the curse of dimensionality , and in high dimensional data when distances lose the ability to discriminate, the p i j {\displaystyle p_{ij}} become too similar (asymptotically, they would converge to a constant). It has been proposed to adjust the distances with a power transform, based on the intrinsic dimension of each point, to alleviate this. [12]

t-SNE aims to learn a d {\displaystyle d} -dimensional map y 1 , … , y N {\displaystyle \mathbf {y} _{1},\dots ,\mathbf {y} _{N}} (with y i ∈ R d {\displaystyle \mathbf {y} _{i}\in \mathbb {R} ^{d}} and d {\displaystyle d} typically chosen as 2 or 3) that reflects the similarities p i j {\displaystyle p_{ij}} as well as possible. To this end, it measures similarities q i j {\displaystyle q_{ij}} between two points in the map y i {\displaystyle \mathbf {y} _{i}} and y j {\displaystyle \mathbf {y} _{j}} , using a very similar approach.

Specifically, for i ≠ j {\displaystyle i\neq j} , define q i j {\displaystyle q_{ij}} as

and set q i i = 0 {\displaystyle q_{ii}=0} .

Herein a heavy-tailed Student t-distribution (with one-degree of freedom, which is the same as a Cauchy distribution) is used to measure similarities between low-dimensional points in order to allow dissimilar objects to be modeled far apart in the map.

The locations of the points y i {\displaystyle \mathbf {y} _{i}} in the map are determined by minimizing the (non-symmetric) Kullback–Leibler divergence of the distribution P {\displaystyle P} from the distribution Q {\displaystyle Q} , that is:

The minimization of the Kullback–Leibler divergence with respect to the points y i {\displaystyle \mathbf {y} _{i}} is performed using gradient descent .

The result of this optimization is a map that reflects the similarities between the high-dimensional inputs.

Output

While t-SNE plots often seem to display clusters , the visual clusters can be strongly influenced by the chosen parameterization (especially the perplexity) and so a good understanding of the parameters for t-SNE is needed. Such "clusters" can be shown to even appear in structured data with no clear clustering, [13] and so may be false findings. Similarly, the size of clusters produced by t-SNE is not informative, and neither is the distance between clusters. [14] Thus, interactive exploration may be needed to choose parameters and validate results. [15] [16] It has been shown that t-SNE can often recover well-separated clusters, and with special parameter choices, approximates a simple form of spectral clustering . [17]

Software

A C++ implementation of Barnes-Hut is available on the github account of one of the original authors.

The R package Rtsne implements t-SNE in R .

ELKI contains tSNE, also with Barnes-Hut approximation

scikit-learn , a popular machine learning library in Python implements t-SNE with both exact solutions and the Barnes-Hut approximation.

Tensorboard, the visualization kit associated with TensorFlow , also implements t-SNE (online version)

The Julia package TSne implements t-SNE

References

External links

Wattenberg, Martin; Viégas, Fernanda; Johnson, Ian (2016-10-13). "How to Use t-SNE Effectively" . Distill . 1 (10): e2. doi : 10.23915/distill.00002 . ISSN 2476-0757 . . Interactive demonstration and tutorial.

Visualizing Data Using t-SNE , Google Tech Talk about t-SNE

Implementations of t-SNE in various languages , A link collection maintained by Laurens van der Maaten