Comparison between t-SNE and UMAP

Fan Cheng

2021-08-05

1 t-SNE

t-Distributed Stochastic Neighbor Embedding [t-SNE; Van der Maaten and Hinton (2008)] is a dimension reduction technique well suited for the visualization of high-dimensional data in scatterplots. It works by minimizing the Kullback-Leibler divergence $KL(P||Q)$ between two joint distributions, p_{ij} and q_{ij}, measuring the pairwise similarities of the high-dimensional input data x_i and the low-dimensional embedded points y_i respectively. To better capture the local data structure and solve the crowding problem, a normalized Student-t kernel is chosen for q_{ij} since it is heavy-tailed, while a Gaussian kernel is used for p_{ij}. The bandwidth of the Gaussian kernel, σ_i is set to control the *perplexity* of the conditional distribution, relating to the density of the input data. Considering the asymmetry of the Kullback-Leibler divergence, the object function aims at modelling similar points in the input space by nearby points in the embedding space. Analytical form of the gradient is used in the Gradient Descent optimization of Kullback-Leibler divergence. Details can be found in Algorithm 1. Variants of the Barnes-Hut algorithm can be used to approximate the gradient of the object function (Van Der Maaten 2014).

Perplexity: to balance attention between local and global aspects of your data.

- The parameter is a guess about the number of close neighbors each point has. The original paper says, *"The performance of SNE is fairly robust to changes in the perplexity, and typical values are between 5 and 50."*

Other hyperparameters: these choices interact with the amount of momentum and the step size are employed in the gradient descent. It is therefore common to run the optimization several times on a data set to find appropriate values for the parameters.

1.1 Illustration

Below are some figures to illustrate the steps in t-SNE from a book named *Machine Learning with R, the tidyverse, and mlr* by Hefin I. Rhys.

```
Algorithm 1: t-SNE
Input                  : high-dimensional data $x_i$ for all $i = 1, \ldots, N$
Output                 : low-dimensional embedding $y_i$ for all $i = 1, \ldots, N$
parameter              : perplexity
optimization parameter : number of iterations $T$, learning rate $\eta$, momentum $\alpha(t)$
1: Compute the Gaussian conditional probability of distances for $x_i$ and $x_j$ as $p_{j|i}$ using
   $$p_{j|i} = \frac{\exp\left(-\left\|x_i - x_j\right\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\left\|x_i - x_k\right\|^2 / 2\sigma_i^2\right)}$$
   where the variance $\sigma_i$ for each $x_i$ optimized with the perplexity parameter by $2^{-\sum_j p_{j|i} \log_2 p_{j|i}} = perplexity$, and set $p_{i|i} = 0$;
2: Set the probability as $p_{ij} = (p_{j|i} + p_{i|j})/(2N)$;
3: Initialize the optimization solution as $\mathcal{Y}^{(0)} = \{y_1, \ldots, y_N\}$ and $\mathcal{Y}^{(1)}$ sampled from normal distribution $\mathcal{N}(0, 10^{-4}\boldsymbol{I})$;
4: for $t = 1, \ldots, T$ do
5:     Compute the Student t-distribution of distances between $y_i$ and $y_j$ as
       $$q_{ij} = \frac{\left(1 + \left\|y_i - y_j\right\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \left\|y_k - y_l\right\|^2\right)^{-1}};$$
6:     Compute the gradient of the Kullback-Leibler divergence loss function for all $i = 1, \ldots, N$ as
       $$\frac{\delta KL}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)\left(1 + \left\|y_i - y_j\right\|^2\right)^{-1};$$
7:     Update $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta KL}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$.
8: end for
```

Figure 14.1. t-SNE measures the distance from each case to every other case, converted into a probability by fitting a normal distribution over the current case. These probabilities are scaled by dividing them by their sum, so that they add to 1.

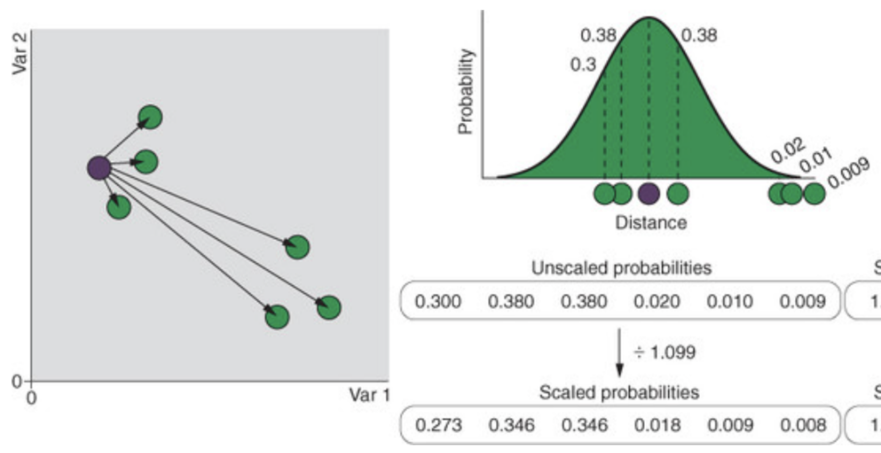


Figure 14.2. The scaled probabilities for each case are stored as a matrix of values. This is visualized here as a heatmap: the closer two cases are, the darker the box is that represents their distance in the heatmap.

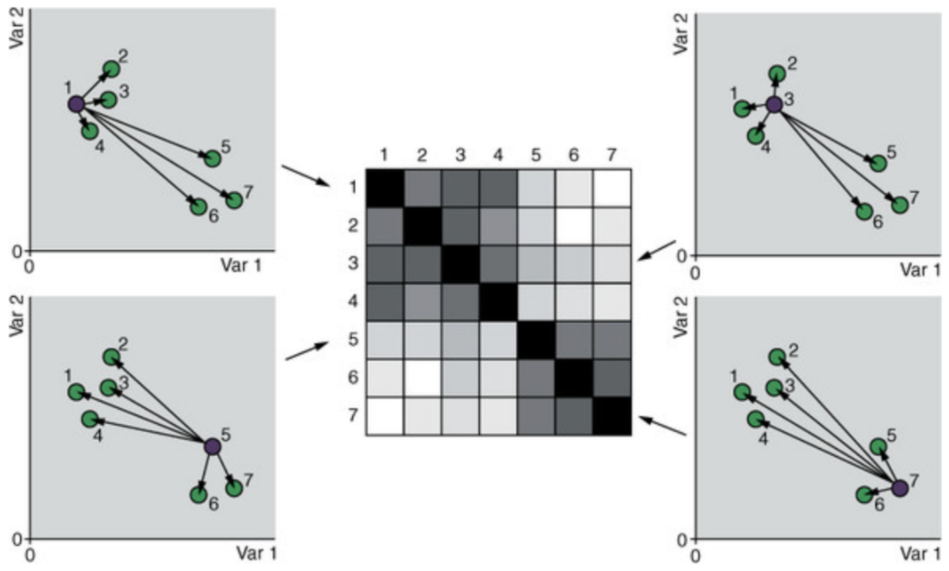


Figure 14.3. When converting distances in the lower-dimensional representation into probabilities, t-SNE fits a Student's t distribution over the current case instead of a normal distribution. The Student's t distribution has longer tails, meaning dissimilar cases are pushed further away to achieve the same probability as in the high-dimensional representation.

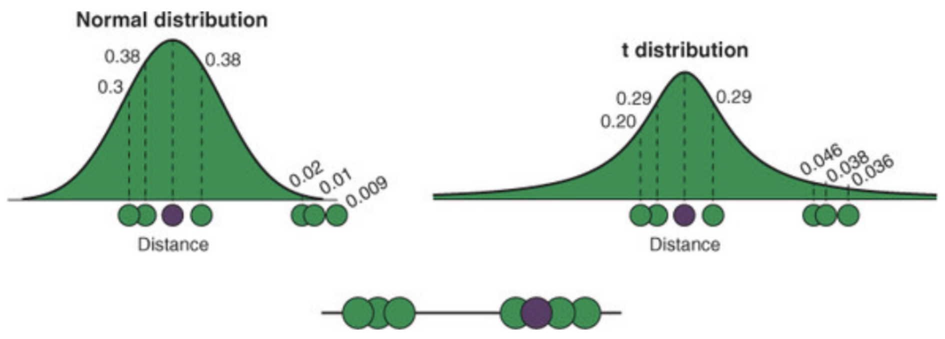
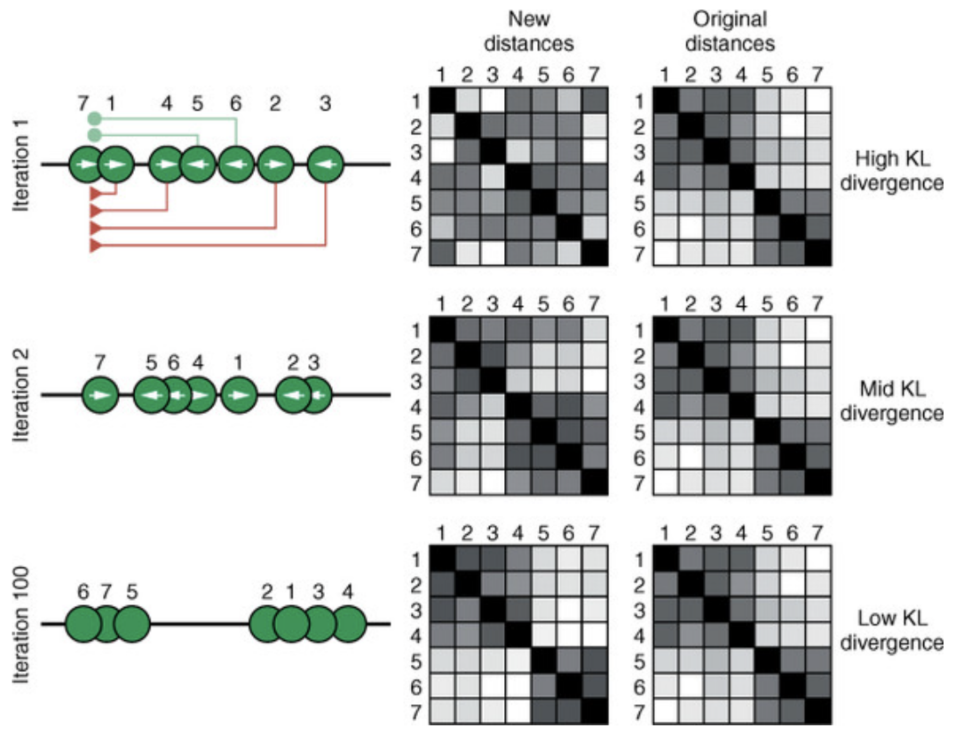


Figure 14.4. Cases are randomly initialized over the new axes (one axis is shown here). The probability matrix is computed for this axis, and the cases are shuffled around to make this matrix resemble the original, high-dimensional matrix by minimizing the Kullback-Leibler (KL) divergence. During shuffling, cases are attracted toward cases that are similar to them (lines with circles) and repulsed away from cases that are dissimilar (lines with triangles).



2 UMAP

UMAP [Uniform Manifold Approximation and Projection; McInnes, Healy, and Melville (2018)] is a competitive dimension reduction methods with t-SNE in increasing speed and preseving global structure. It is based on Laplacian Eigenmaps, considering Riemannian geometry and algebraic topology. Instead of optimizing the similarities in t-SNE, UMAP works by minimizing the error between two topological representations from the input space and embedding space. The error is measured by the cross entropy C of two local fuzzy simplical sets built from the nearest neighborhood graph, with edge weights representing the likelihood that two points are connected. UMAP extends a raidus outwards at each point until overlap with the nearest neighbors, with a *min_dist* as a desired separation between nearby embedded points to control global structure. Finally, similar to t-SNE, stochastic gradient descent is used for the optimization of object function. The algorithm and hyperparameters are detailed in Section 4 of McInnes, Healy, and Melville (2018).

```
Algorithm 2: UMAP
Input     : high-dimensional data $x_i$ for all $i = 1, \dots, N$
Output    : low-dimensional embedding $y_i$ for all $i = 1, \dots, N$
parameter: number of nearest neighbors $K$, minimum distance between embedded points min_dist, amount of
           optimization work n_epochs
1: Construct the weighted $K$-nearest neighborhood graph with the distance to the nearest neighbor as $\rho_i$ for each $x_i$;
2: Compute the exponential probability of distances between $x_i$ and $x_j$ as $p_{j|i}$ using
       $p_{i|j} = e^{-\frac{d(x_i, x_j) - \rho_i}{\sigma_i}}$,
   where $d$ can be any distance metric, such that $\log_2 K = \sum_i p_{ij}$;
3: Construct a local fuzzy simplicial set by setting $p_{ij} = p_{j|i} + p_{i|j} - p_{i|j} p_{j|i}$;
4: Compute the distribution of distances between $y_i$ and $y_j$ as
       $q_{ij} = \left(1 + a(y_i - y_j)^{2b}\right)^{-1}$,
   where $a$ and $b$ are found using min_dist parameter such that
       $q_{ij} \approx \begin{cases} 1 & \text{if } y_i - y_j \leq min\_dist \\ e^{-(y_i - y_j) - min\_dist} & \text{if } y_i - y_j > min\_dist; \end{cases}$
5: The cost function is constructed using a binary fuzzy set cross entropy (CE) as
       $CE(X, Y) = \sum_i \sum_j \left[ p_{ij}(X) \log\left(\frac{p_{ij}(X)}{q_{ij}(Y)}\right) + (1 - p_{ij}(X)) \log\left(\frac{1 - p_{ij}(X)}{1 - q_{ij}(Y)}\right) \right]$
6: Initialize the embedding coordinates $y_i$ using the graph Laplacian similar to Laplacian Eigenmaps in Algorithm
   refalg:le;
7: Optimize $y_i$ by minimizing the cross entropy using stochastic gradient descent with n_epochs.
```

2.1 Parameters

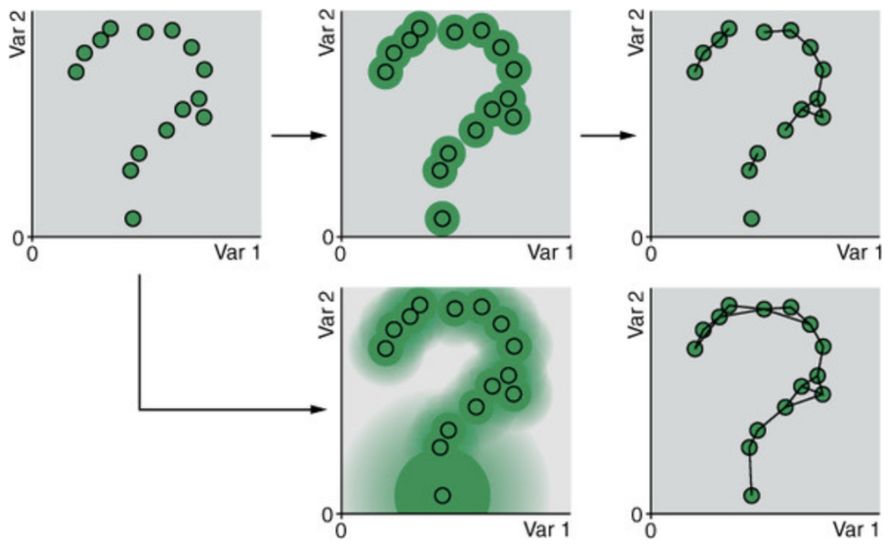
- K: number of nearest neighbors
- *min_dist*: the minimum distance between points in low-dimensional space.

  This parameter controls how tightly UMAP clumps points together, with low values leading to more tightly packed embeddings. Larger values of *min_dist* will make UMAP pack points together more loosely, focusing instead on the preservation of the broad topological structure.

2.2 Illustration

Below are some figures to illustrate the steps in UMAP from the same book.

Figure 14.8. How UMAP learns a manifold. UMAP expands a search region around each case. A naive form of this is shown in the top row, where the radius of each search region is the same. When cases with overlapping search regions are connected by edges, there are gaps in the manifold. In the bottom row, the search region extends to the nearest neighbor and then extends outward in a fuzzy way, with a radius inversely related to the density of data in that region. This results in a complete manifold.



3 Compare

The biggest difference between the the output of UMAP when compared with t-SNE is this balance between local and global structure - UMAP is often better at preserving global structure.

3.1 Visualization examples

The following blog contains visualization examples for comparing t-SNE and UMAP, as well as the effect of hyperparameters on the embedding.

`https://pair-code.github.io/understanding-umap/`

3.2 t-SNE disadvantages

- tSNE does not scale well for rapidly increasing sample sizes.
- tSNE does not preserve global data structure. Only within cluster distances are meaningful while between cluster similarities are not guaranteed, therefore it is widely acknowledged that clustering on tSNE is not a very good idea.
- tSNE can practically only embed into 2 or 3 dimensions.
- tSNE can not work with high-dimensional data directly, Autoencoder or PCA are often used for performing a pre-dimensionality reduction before plugging it into the tSNE.
- tSNE consumes too much memory for its computations which becomes especially obvious when using large perplexity.

3.3 Takeaways from the blog

- Hyperparameters really matter
- Cluster sizes in a UMAP plot mean nothing
- Distances between clusters might not mean anything
- Random noise doesn't always look random

  Especially at low values of n_neighbors, spurious clustering can be observed.
- You may need more than one plot

4 Things related to our paper

t-SNE Variants: implemented via Barnes-Hut approximations for gradient descent process, which also limits the implementation to two or three dimensions. Tang et al. (2016) extended t-SNE with the LargeVis algorithm where random projection trees are used to reduce time cost.

L.J.P. van der Maaten. Accelerating t-SNE using Tree-Based Algorithms. Journal of Machine Learning Research 15(Oct):3221-3245, 2014.

UMAP: allow any distance metric for constructing high-dimensional distance probabilities using NN-Descent (Dong, Moses, and Li 2011) algorithm; used procrustes measure to measure stability of the embedding.

5 Estimating intrinsic dimensions

There are multiple methods to estimate the intrinsic dimensions, with 17 of them implemented in R package *Rdimtools*.

`https://kisungyou.com/Rdimtools/reference/index.html#section--dimension-estimation`

- Running 5 of them on MNIST dataset gives around 12 dimensions as an output.
- How to deal with the implementation of t-SNE then?
- How to unify this in the paper?

6 TODO

- Run simulations for both dimension 2 and 12 on MNIST and check if the conclusion holds.
- For electricity dataset, since the goal is to find anomalies, we could choose dimension 2 for simplicity. (otherwise compute densities for high-dimensional embedding)

References

Dong, Wei, Charikar Moses, and Kai Li. 2011. "Efficient k-Nearest Neighbor Graph Construction for Generic Similarity Measures." In *Proceedings of the 20th International Conference on World Wide Web*, 577–86. WWW '11. New York, NY, USA: Association for Computing Machinery.

McInnes, Leland, John Healy, and James Melville. 2018. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," February. `https://arxiv.org/abs/1802.03426`.

Tang, Jian, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. 2016. "Visualizing Large-Scale and High-Dimensional Data." In *Proceedings of the 25th International Conference on World Wide Web*, 287–97. WWW '16. Republic; Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee.

Van der Maaten, Laurens, and Geoffrey Hinton. 2008. "Visualizing Data Using t-SNE." *Journal of Machine Learning Research* 9 (Nov): 2579–2605.

Van Der Maaten, Laurens. 2014. "Accelerating t-SNE Using Tree-Based Algorithms." *Journal of Machine Learning Research* 15 (1): 3221–45.