Introduction

https://www.analyticsvidhya.com/blog/2017/01/t-sne-implementation-r-python/ (https://www.analyticsvidhya.com/blog/2017/01/t-sne-implementation-r-python/)
https://www.youtube.com/watch?v=NEaUSP4YerM (https://www.youtube.com/watch?v=NEaUSP4YerM): a very intuitive video.
https://www.oreilly.com/learning/an-illustrated-introduction-to-the-t-sne-algorithm (https://www.oreilly.com/learning/an-illustrated-introduction-to-the-t-sne-algorithm)
In the last link, there is a 64 dimensional digits picture data. Using t-SNE can group the data even before any classification. This can have a very intuitive picture on what t-SNE does.

- We know PCA can do dimensionality reduction and visualization. While PCA is a linear algorithm (1933), t-Distributed Stochastic Neighbor Embedding (t-SNE) is a new (2008) and much more effective non-linear algorithm than PCA. See pros and cons of both algorithm in the end.
- t-SNE is a non-linear dimensionality reduction algorithm used for exploring high-dimensional data. It maps multi-dimensional data to two or more dimensions suitable for human observation.
- **PCA, hierarchical clustering, and t-SNE are major algorithms for visualizing data in lower dimensions.**
- **t-SNE is not a classification algorithm but a dimension reduction technique. However, it can be used for other classification.** It has limitation though, as many things have been changed (see the end of this notes).

Limitations of PCA

- PCA is a linear algorithm. It will not be able to interpret nonlinear relationship between features. On the other hand, t-SNE is based on probability distributions with random walk on neighborhood graphs to find the structure within the data.
- A major problem with, linear dimensionality reduction algorithms is that they concentrate on placing dissimilar data points far apart in a lower dimension representation. But in order to represent high dimension data on low dimension, non-linear manifold, it is important that similar datapoints must be represented close together, which is not what linear dimensionality reduction algorithms do.
- Local approaches seek to map nearby points on the manifold to nearby points in the low-dimensional representation. Global approaches on the other hand attempt to preserve geometry at all scales, i.e mapping nearby points to nearby points and far away points to far away points
- It is important to know that most of the nonlinear techniques other than t-SNE are not capable of **retaining both the local and global structure of the data at the same time.**
- **Summary PCA focuses on global while t-SNE focus on local similarity.**

Algorithmic details of t-SNE

t-SNE is an improvement on the Stochastic Neighbor Embedding (SNE) algorithm.

General idea

- For a data point x_i in the original data space. In the example of [https://www.oreilly.com/learning/an-illustrated-introduction-to-the-t-sne-algorithm (https://www.oreilly.com/learning/an-illustrated-introduction-to-the-t-sne-algorithm)](https://www.oreilly.com/learning/an-illustrated-introduction-to-the-t-sne-algorithm), we have 1797 such data points, each point is a 64 dimensional digit picture.
- We want to map the above high-dimensional data points to a low-dimensional 2D space with map points y_i. The criterion for such a mapping is as follows: If the points are close together in the original space, then they should be close in the mapping lower space too. Now how to measure whether two points are close?

Defining similarity as conditional probability in original data space

- Now use the conditional probability below as a quantity to measure the closeness in the two spaces. First in the original space, we have
$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2/2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-\|x_i - x_k\|^2/2\sigma_i^2\right)}$$
Check the youtube video on why we need normalize with the denominator. Further more, the conditional probability is symmetrized as $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2N}$. Obviously, the bigger the conditional probability, the closer the two points. The σ_i is calculated on specific data via a binary search.
- From the above definitions, we can calculate the similarity matrix p_{ij}. Even from this intermediate result, we can normally observe some grouping effects.

Defining similarity as a conditional probability in mapping data space

- Similar to the original data space, we define in the mapping space a conditional probability to indicate the similarity of two points,
$$q_{j|i} = \frac{f(\|y_i - y_j\|)}{\sum_{k\neq i} f(\|y_i - y_k\|)}, \text{ with } f(z) = \frac{1}{1+z^2}$$
This is a t-Student distribution (one degree of freedom), or Cauchy distribution.
- So far the data similarity p_{ij} is fixed, the map similarity matrix q_{ij} depends on the map points, which will be determined step by step.
- The reason why we choosing a different distribution in mapping space will be detailed later.

How map points are determined? A physical analogy

Assume that our map points are all connected with springs. The stiffness of a spring connecting points i and j depends on the mismatch between the similarity of the two data points and the similarity of the two map points, that is, $p_{ij} - q_{ij}$. Now, we let the system evolve according to the laws of physics. **If two map points are far apart while the data points are close, they are attracted together.** If they are nearby while the data points are dissimilar, they are repelled. The final mapping is obtained when the equilibrium is reached.

Algorithm

Remarkably, the physical analogy above stems naturally from the mathematical algorithm. It corresponds to minimizing the Kullback-Leiber divergence between the two distributions p_{ij} and q_{ij}:

$$\mathrm{KL}(P||Q) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

This measures the distance between two similarity matrices.

To minimize this score, we perform a gradient descent. The gradient can be computed analytically:

$$\frac{\partial \mathrm{KL}(P||Q)}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij}) g(\|x_i - x_j\|) u_{ij} \text{ where } g(z) = \frac{z}{1+z^2}$$

Here, u_{ij} is a unit vector going from y_j to y_i. This gradient expresses the sum of all spring forces applied to map point i.

Once obtaining the gradient, then we use iterative approach to obtain the final results for map points y_i. This seems much like the process in Expectation maximization.

Comments

$$\mathrm{KL}(P||Q) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}} = -\sum_{ij} p_{ij} \log \frac{q_{ij}}{p_{ij}} = -\sum_{ij} p_{ij} \log q_{ij} + \sum_{ij} p_{ij} \log p_{ij}$$

The last term $\sum_{ij} p_{ij} \log p_{ij}$ is entropy of p_{ij} and here is a constant. So it will not contribute to the gradient. Therefore, it is possible we only minimize the cross-entropy term $-\sum_{ij} p_{ij} \log q_{ij}$ to obtain the same results.

Why t-Student distribution in map space?

- It is well known that the volume of the N-dimensional ball of radius r scales as rN. When N is large, if we pick random points uniformly in the ball, most points will be close to the surface, and very few will be near the center. See the demonstration in https://www.oreilly.com/learning/an-illustrated-introduction-to-the-t-sne-algorithm (https://www.oreilly.com/learning/an-illustrated-introduction-to-the-t-sne-algorithm)
- When reducing the dimensionality of a dataset, if we used the same Gaussian distribution for the data points and the map points, we would get an imbalance in the distribution of the distances of a point's neighbors. This is because the distribution of the distances is so different between a high-dimensional space and a low-dimensional space. This imbalance would lead to an excess of attraction forces and a sometimes unappealing mapping.
- The t-SNE algorithm works around this problem by using a t-Student with one degree of freedom distribution for the map points. This distribution has a much heavier tail than the Gaussian distribution, which compensates the original imbalance. Basically, the using of t_SNE will help avoid the congestion of data points in the map space.

Time and Space Complexity

The algorithm computes pairwise conditional probabilities and tries to minimize the sum of the difference of the probabilities in higher and lower dimensions. This involves a lot of calculations and computations.

The normal t-SNE has a quadratic time and space complexity in the number of data points. This makes it particularly slow and resource draining while applying it to data sets comprising of more than 10,000 observations. **However, one could for example obtain an $O(N \log N)$ complexity by using the Barnes-Hut algorithm to accelerate the N-body simulation via a quadtree or an octree.**

What does t-SNE actually do and where to use?

- t-SNE is a non-linear dimensionality reduction algorithm which finds patterns in the data by identifying observed clusters based on similarity of data points with multiple features.
- **t-SNE is not a clustering algorithm. It is a dimensionality reduction algorithm. This is because it maps the multi-dimensional data to a lower dimensional space, the input features are no longer identifiable. Thus you cannot make any inference based only on the output of t-SNE. So essentially it is mainly a data exploration and visualization technique. However, even with other classification algorithm, the results might not be good due to the loss of global information. See later.**
- But t-SNE can be used in the process of classification and clustering by using its output as the input feature for other classification algorithms. (Find some examples later)

Use cases

Main usage

- The main problem while using t-SNE is the black box type nature of the algorithm. Moreoever, it doesn't always provide a similar output on successive runs.
- Therefore, the best way to use the algorithm is to use it for exploratory data analysis. It will give you a very good sense of patterns hidden inside the data.
- It can also be used as an input parameter for other classification & clustering algorithms.
- t-SNE can be used on almost all high dimensional data sets. But it is extensively applied in Image processing, NLP, genomic data and speech processing. Below are a few examples:

Facial Expression Recognition

t-SNE is used for reducing the high-dimensional data into a relatively low-dimensional subspace and then using other algorithms like AdaBoostM2, Random Forests, Logistic Regression, NNs and others as multi-classifier for the expression classification.

Identifying Tumor subpopulations

Mass spectrometry imaging (MSI) is a technology that simultaneously provides the spatial distribution for hundreds of biomolecules directly from tissue. Spatially mapped t-SNE, a nonlinear visualization of the data that is able to better resolve the biomolecular intratumor heterogeneity.

Text comparison using wordvec

Word vector representations capture many linguistic properties such as gender, tense, plurality and even semantic concepts like “capital city of”. Using dimensionality reduction, a 2D map can be computed where semantically similar words are close to each other. This combination of techniques can be used to provide a bird’s-eye view of different text sources, including text summaries and their source material. This enables users to explore a text source like a geographical map.

t-SNE compared to other dimensionality reduction algorithms

- t-SNE outputs provide better results than PCA and other linear dimensionality reduction models. This is because a linear method such as classical scaling is not good at modeling curved manifolds. It focuses on preserving the distances between **widely separated data points** rather than on preserving the distances between nearby data points.

- The Gaussian kernel employed in the high-dimensional space by t-SNE defines a **soft border** between the local and global structure of the data. And for pairs of data points that are close together relative to the standard deviation of the Gaussian, the importance of modeling their separations is almost independent of the magnitudes of those separations. Moreover, t-SNE determines the local neighborhood size for each data point separately based on the local density of the data (by forcing each conditional probability distribution to have the same perplexity). This is because the algorithm defines a soft border between the local and global structure of the data. And unlike other non-linear dimensionality reduction algorithms, it performs better than any of them.

Common Fallacies

- For the algorithm to execute properly, the perplexity should be smaller than the number of points. The suggested perplexity is in the range of (5 to 50)
- Cluster sizes in any t-SNE plot must not be evaluated for standard deviation, dispersion or any other similar measures. This is because t-SNE expands denser clusters and contracts sparser clusters to even out cluster sizes. This is one of the reasons for the crisp and clear plots it produces.
- Distances between clusters may change because global geometry is closely related to optimal perplexity. And in a dataset with many clusters with different number of elements one perplexity cannot optimize distances for all clusters.
- Patterns may be found in random noise as well, so multiple runs of the algorithm with different sets of hyperparameter must be checked before deciding if a pattern exists in the data.
- Different cluster shapes may be observed at different perplexity levels.
- Topology cannot be analyzed based on a single t-SNE plot, multiple plots must be observed before making any assessment.

Limitation of t-SNE

https://stats.stackexchange.com/questions/238538/are-there-cases-where-pca-is-more-suitable-than-t-sne/249520#249520
(https://stats.stackexchange.com/questions/238538/are-there-cases-where-pca-is-more-suitable-than-t-sne/249520#249520)

- t -SNE is a great piece of Machine Learning but one can find many reasons to use PCA instead of it.
- Stochasticity of final solution. PCA is deterministic; t-SNE is not. One gets a nice visualization and then her colleague gets another visualization and then they get artistic which looks better and if a difference of 0.03% in the KL(P||Q) divergence is meaningful... In PCA the correct answer to the question posed is guaranteed. t-SNE might have multiple minima that might lead to different solutions. This necessitates multiple runs as well as raises questions about the reproducibility of the results.
- Interpretability of mapping. This relates to the above point but let's assume that a team has agreed in a particular random seed/run. Now the question becomes what this shows... t-SNE tries to map only local / neighbours correctly so our insights from that embedding should be very cautious; **global trends are not accurately represented** (and that can be potentially a great thing for visualisation).

On the other hand, PCA is just a diagonal rotation of our initial covariance matrix and the eigenvectors represent a new axial system in the space spanned by our original data. We can directly explain what a particular PCA does.

- Application to new/unseen data. **t-SNE is not learning a function from the original space to the new (lower) dimensional one and that's a problem. On that matter, t-SNE is a non-parametric learning algorithm so approximating with parametric algorithm is an ill-posed problem.** The embedding is learned by directly moving the data across the low dimensional space. That means one does not get an eigenvector or a similar construct to use in new data. In contrast, using PCA the eigenvectors offer a new axes system what can be directly used to project new data. `[Apparently one could try training a deep-network to learn the t-SNE mapping (you can hear Dr. van der Maaten at ~46' of this video suggesting something along this lines) but clearly no easy solution exists.]`
- Incomplete data. Natively t-SNE does not deal with incomplete data. In fairness, PCA does not deal with them either but numerous extensions of PCA for incomplete data (eg. probabilistic PCA) are out there and are almost standard modelling routines. t-SNE currently cannot handle incomplete data (aside obviously training a probabilistic PCA first and passing the PC scores to t-SNE as inputs).
- The k is not (too) small case. t-SNE solves a problem known as the crowding problem, effectively that somewhat similar points in higher dimension collapsing on top of each other in lower dimensions. Now as you increase the dimensions used the crowding problem gets less severe ie. the problem you are trying to solve through the use of t-SNE gets attenuated. You can work around this issue but it is not trivial. Therefore if you need a k dimensional vector as the reduced set and k is not quite small the optimality of the produce solution is in question. PCA on the other hand offer always the k best linear combination in terms of variance explained. (Thanks to @amoeba for noticing I made a mess when first trying to outline this point.)
- I do not mention issues about computational requirements (eg. speed or memory size) nor issues about selecting relevant hyperparameters (eg. perplexity). I think these are internal issues of the t-SNE methodology and are irrelevant when comparing it to another algorithm.
- To summarise, t-SNE is great but as all algorithms has its limitations when it comes to its applicability. I use t-SNE almost on any new dataset I get my hands on as an explanatory data analysis tool. I think though it has certain limitations that do not make it nearly as applicable as PCA. Let me stress that PCA is not perfect either; for example, the PCA-based visualisations are often inferior to those of t-SNE.

- The idea of SNE and t-SNE is to place neighbors close to each other, (almost) completly ignoring the global structure. This is excellent for visualization, because similar items can be plotted next to each other (and not on top of each other, c.f. crowding).
- **This is not good for further analysis. Global structure is lost, some objects may have been blocked from moving to their neighbors, and separation between different groups is not preserved quantitatively**. Which is largely why e.g. clustering on the projection usually does not work very well.
- PCA is quite the opposite. It tries to preserve the global properties (eigenvectors with high variance) while it may lose low-variance deviations between neighbors.