t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a machine learning algorithm for visualization developed by Laurens van der Maaten and Geoffrey Hinton. It is a nonlinear dimensionality reduction technique well-suited 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 constructs a probability distribution over pairs of high-dimensional objects in such a way that similar objects have a high probability of being picked, while dissimilar points have an extremely small probability of being picked.

Probability Function

Let's take the example of a 2-D distribution. Pick a point out of the blue category from the below image and call it point i. Now using the Normal distribution equation, we can encode the distance of every other point, j, in form of probability that the point belongs to the same category as that of point i. Farther a point from i, lesser is the probability that the point belongs to the same group.

Figure 1: Dataset with two distinct groups

In the above dataset image one category have much higher density than the other one. The conditional probability function, which just takes ratio of probability of choosing point j w.r.t. point i and the sum of the probability of every point w.r.t. point i, makes sure that the local distribution of each data is represented equally, regardless of its density.

The basic idea behind it is to minimize the cost function, that describes the divergence of our (low-dimensional) representation to the actual (high-dimensional) data, using the good old fashioned Gradient Descent. Or simply stated, a cost function that represents the error in our representation. That cost function is the Kullback–Leibler divergence (KL Divergence).

t-SNE Algorithm

Step 1: Given a set of N high-dimensional objects $x_1,...,x_N$, t-SNE first computes probabilities p_{ij} that are proportional to the similarity of objects x_i and x_j, as follows:

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

This measures how close x_j is from x_i, considering a Gaussian distribution around x_i with a given variance. This variance is different for every point; it is chosen such that points in dense areas are given a smaller variance than points in sparse areas.

The algorithm starts by converting the shortest distance (a straight line) between the points into probability of similarity of points. Where, the similarity between points is: the conditional probability that x_i would pick x_j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian (normal distribution) centered at x_i .

Step 2: For the low-dimensional counterparts y_i and y_j of the high-dimensional datapoints x_i and x_j it is possible to compute a similar conditional probability, which we denote by $q_{j|i}$

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

Note that $p_{i|i}$ and $p_{j|j}$ are set to zero as we only want to model pair wise similarity.

In simple terms step 1 and step2 calculate the conditional probability of similarity between a pair of points in High dimensional space and in low dimensional space

We will try to understand this in detail:

Let us map 3D space to 2D space. What step1 and step2 are doing is calculating the probability of similarity of points in 3D space and calculating the probability of similarity of points in the corresponding 2D space. Logically, the conditional probabilities $p_{j|i}$ and $q_{j|i}$ must be equal for a perfect representation of the similarity of the datapoints in the different dimensional spaces, i.e the difference between $p_{j|i}$ and $q_{j|i}$ must be zero for the perfect replication of the plot in high and low dimensions. By this logic SNE attempts to minimize this difference of conditional probability.

Step 3: Now here is the difference between the SNE and t-SNE algorithms.

To measure the minimization of sum of difference of conditional probability SNE minimizes the sum of Kullback-Leibler divergences overall data points using a gradient descent method. We must know that KL divergences are asymmetric in nature.

In other words, the SNE cost function focuses on retaining the local structure of the data in the map.

So t-SNE also tries to minimize the sum of the difference in conditional probabilities. But it does that by using the symmetric version of the SNE cost function, with simple gradients.

Step 4: The remaining parameter to be selected is the variance σ_i of the student's t-distribution that is centered over each high-dimensional datapoint x_i. It is not likely that there is a single value of σ_i that is optimal for all data points in the data set because the density of the data is likely to

vary. In dense regions, a smaller value of σ_i is usually more appropriate than in sparser regions. Any particular value of includes a probability distribution, P_i, over all of the other data points. This distribution has an entropy which increases as σ_i increases. t-SNE performs a binary search for the value of σ_i that produces a P_i with a fixed perplexity that is specified by σ_i the user. The perplexity is defined as

$$\text{Perp}(P_i) = 2^{H(P_i)},$$

Where $H(P_i)$ is the Shannon Entropy of p_i measured in bits,

$$H(p_i) = -\sum_j p_{j|i} log_2 p_{j|i}$$

The 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. The minimization of the cost function is performed using gradient decent. And physically, the gradient may be interpreted as the resultant force created by a set of difference between the map point and all other map points.

What does t-SNE actually do?

We have looked into the mathematical description of how does the algorithms works.

To summarize t-SNE a non-linear dimensionality reduction algorithm finds patterns in the data by identifying observed clusters based on similarity of data points with multiple features. But it 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.

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.

References

A blog on more details on t-SNE:
https://medium.com/@layog/i-dont-understand-t-sne-part-1-50f507acd4f9Visualizing Data using t-SNE:
http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf