**Explain what the algorithm is used for**.

t-SNE stands for **t-Distributed Stochastic neighbor embedding**, is a dimension reduction/data visualization method.

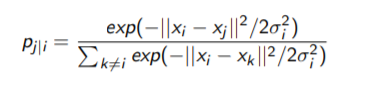
Proposed by Laurens van der Maaten & Geoffrey Hinton in 2008.

t-SNE tends to preserve local structure at the same time preserving the global structure as much as possible.

t-SNE is based on SNE which aim is to match distributions of distances between points in high and low dimensional space via conditional probabilities. SNE assumes that both high and low dimension space are gaussian distributed.

**SNE:**

* Let be the *i th* object in high dimensional space
* Let *yi* be the *i th* object in low dimensional space
* Construct:

Formula to calculate conditional probability of P of j given i for High dimensional space  


Here,

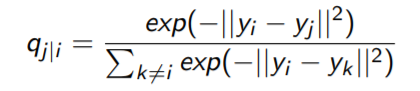
Xi and Xj are two data points,

**σ** is a variance,

In numerator we take exponential difference of two points and take known (i.e. nearest integer o/p) and squared it and divided by two time by variance.

We take exponent to avoid getting 0.

Formula to calculate conditional probability of q of j given i for Low dimensional space.



In low dimensional space we don’t have Variance or need not to divide by 2σ²,

Since it assumes that,

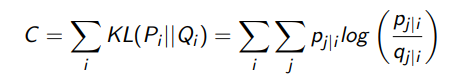
σ = 1/√2

2σ² = 1

* If your comparing 2 similar points then,

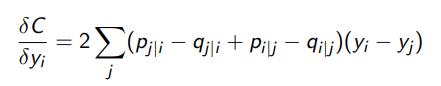


* After comparing these 2 conditional probabilities in both low and high dimensional space, the objective is to produce the difference between this too so that when we project the data from high dimension to lower its looks as similar as data in high dimension
* To maintain that data as much as possible we use cost function **C.**
* Match these functions by minimizing sum of Kullback-Leibler divergences:



It’s a sum of, KL divergence between the probability distribution of P of i in Low dimensional space to the prob dist. of Q of j in low dimensional space.

* Since KL divergence is asymmetric,
  + large cost for representing nearby data points in high dimensional map by widely separated points in the low dimensional map
  + smaller coast for representing widely separated data points in high dimensional map by nearby points in the low dimension
* Hence local structure is highly preserved
* **σ**i is associated with a parameter called perplexity which can be loosely interpreted as the number of close neighbors each point has
* Gradient of the cost function:



* When we differentiate with respect to Yi then this is our cost function
* Given the cost function SNE uses gradient descent for optimization
* In addition to the gradient of the cost function it also has a momentum term to speed up the optimization and to avoid local optima
* Gradient Descent formula:



Where,

α(t): Momentum at iteration t

Y (t): Solution at iteration t

η: Learning rate

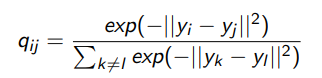
In SNE you define similarities, define cost function, get the gradient for the cost function and try to minimize it in order to get low dimensional map.

* SNE has two main drawbacks:
  + Cost function is difficult to optimize
  + Crowding problem: Idea of crowding problem is, if you have points that are moderately far apart and points which are near-by then SNE tend to clump all of them together.

* That’s why we have t-SNE to overcome these drawbacks
* Novel features in t-SNE :
  + cost function has two distinct features:
  + Cost function is symmetrized version of that in SNE. i.e. (pi|j = pj|i and qi|j = qj|i)
  + t-distribution is used to compute the similarities between data points in the low dimensional space.

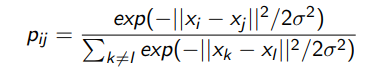
**Symmetric SNE:**

* The main feature in symmetric SNE is that **p**ij = **p**ji and **p**ii = **q**ii = 0 for all i,j
* The Low dimension is represented as:

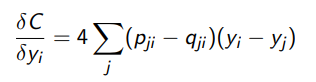


Since, it’s a symmetric version of similarities therefor there is no **q** of I given j.

* The high dimension is represented as:

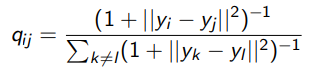


* Gradient of the cost function:

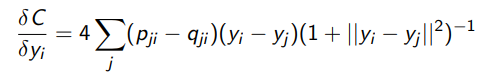


**t-SNE:**

* In t-SNE a t distribution with one degree of freedom is used to represent the low dimensional map:



* + In t-SNE the reason why t-Distribution is used is that its robust to outliers.
  + Unlike a Gaussian distribution it doesn’t have exponent in it so faster to evaluate
* Gradient of the cost function:



In addition to symmetric SNE we have additional function in t-SNE

**The general idea of t-SNE algorithm:**

* Data: data set X with data points = x1, x2, ..., xn each of these points have very high dimension
* cost function parameter: Perplexity Perp, perplexity is associated with variance σ in the cost function
* Optimization parameters: number of iterations T, learning rate η, momentum α(t)

We use all of them to get low dimensional representation Y.

**t-SNE in step by step:**

1. Compute pairwise affinities pj|i with perplexity Perp
2. Set **p**ij = (**p**i|j + **p**j|i)/ 2n (n - Number of data points)
3. Sample initial solution Y (0) = y1, y2, ..., yn from N (mean=0, variance=10-4I)

Where, N = Normally distributed population,

Mean = 0,

Variance= 10-4I

I =identity matrix, so if we have **n** data points then it will be a **n** by **n** identity matrix

For each iteration of poits :

1. Compute low-dimensional affinities **q**ij
2. Compute gradient δC/ δy
3. Set,



Repeat until it converges.

**How fast does it run?**

The algorithm takes around 9-10 minutes to plot the data in lower dimension with max-iterations are set to 1000 and perplexity is set to 20.

**Will scaling effect it?**

Scaling is necessary if you want the different dimensions to be treated with equal importance, since the 2-norm will be more heavily influenced by dimensions with large variance.

**Did I explain my idea clearly overall? (How effective are you at explaining what you are doing?**

t-SNE stands for **t-Distributed Stochastic neighbor embedding**, is a dimension reduction/data visualization method.

To understand t-SNE more clearly and mathematically we started with algorithm it’s based on

The **SNE**.

Idea behind SNE is explained with drawbacks and later we introduced to **symmetric SNE**.

t-SNE is explained properly with how it overcomes the drawbacks of SNE and why we use t-distribution over other.

The mathematical equations and their annotations are also explained to get understanding of the algorithm clearly.

To support the statistical explanation, I also implemented Jupyter notebook implementing t-SNE and plotted the t-SNE graph in low dimensional space.

**Citation**:

<https://www.youtube.com/watch?v=NEaUSP4YerM>

<https://www.youtube.com/watch?v=ohQXphVSEQM>

<https://www.youtube.com/watch?v=W-9L6v_rFIE>

<https://stats.stackexchange.com/questions/164917/should-data-be-centeredscaled-before-applying-t-sne>

<https://stackoverflow.com/questions/45824724/is-t-snes-computational-bottleneck-its-memory-complexity>