Good Afternoon everyone! Today, I am going to present about t-distributed Stochastic Neighbour Embedding, shorted as t-SNE.

In the first few classes in ISI, I learned that before doing any kind of sophisticated analysis, we should always resort to some exploratory analysis, which will give us a feel for the data. One such analysis would be to plot it, but for data with hundreds of dimension it is almost impossible to visualize it in its entirety.

t-SNE is an unsupervised dimensional reduction technique, built specifically for this purpose. It is currently extensively used in various fields like signal processing, text analysis, bioinformatics.

Before proceeding with internal workings of tSNE, let us see how it behaves to real dataset. We take MNIST dataset which consists of images of handwritten digits, each being 28 cross 28 Black and White images. So, we can feed in 28 sqaured equal to 784 size vector to tSNE algorithm and try to visualize it in 2D.

Now as we plot the tSNE output in 2D, we see some nice clusters. Let us see whether these clusters are natural or artificially created by tSNE.

We add colors corresponding to different digits in the previous plot. We see very nice clustering of points corresponding to different digits. Note that, these true labels were not used as an input to tSNE algorithm, it learned the pattern somehow.

Now that I assume I have grabbed your attention about tSNE, let us formally define the problem of visualization.

We have Xi's which are d dimensional vector where d is very high. We have their representation Yi's which are p dimensional vectors in feature space where p is 2 or 3. We want Xi and Yi to be related in some way.

Now, do we want Yi's to capture the distances between Xi's. No, that won't be helpful. For example, consider the spiral data, where points are sequentially colored according to their position in manifold. Linear projection like PCA would mix up all the colors in the middle as you can see. While the ideal representation would be a line with sequential colors, as you see here.

The problem is the euclidean distance between those points is small, but the distance through manifold is large, which should be considered. Hence, the idea is to retain local distances, rather than focusing on global distance.

Before going on to details of tSNE, let us discuss SNE first. Define pj given i as the expression given.

What it means is that, suppose you have a Gaussian distribution centered as Xi with variance sigma-i squared times Identity matrix, then the likelihood of choosing Xj as your neighbour is that quantity. Note that p-i given j and p-j given i are different due to different normalization factors. Similarly, for Yi's in output space, we define similar quantities q-i given j, but considering variance to be 1/2 for easier calculation.

The sigma-i's are selected in such a way so that approximately same number of neighbours lie in a ball centered at Xi. Hence, dense region means low sigma-i, sparse region means high sigma-i.

So, for each i, we have probabilities Pi in feature space, probabilities Qi in output space, the obvious thing to do would be to minimize Kullback Leibler divergence between Pi and Qi. And take sum of all such divergences over i.

There are many other types of divergence as well, so why KL Divergence.

Reason 1, it is asymmetric. Let's say we have small yi and yj in output space being very distant, but xi and xj are close in input space. This is not very good situation, right? Note that, here p-j given i would be large, but q-j given i would be very small, so this discrepancy will be fairly large and there is large cost for that. So, small distances remain small in output space too, preserving local structure.

Reason 2 is that the gradient becomes simple. Think of it like this, on yi here, all other points is exerting a force towards it, multiplied by a constant, and those forces add up to determine where should it move.

Now consider lots of datapoints in 2D which are of about same distances from each other. In 1D, you can have only 3 such points. If you use SNE here, all the points would crush together at origin and becomes crowded. The reason is each KL between Pi and Qi is minimized independently.

To avoid crowding, we try to model pij jointly. Then we would have only one P and Q, and we could minimize KL divergence between them.

Now, suppose the data has an outlier X1. Then all distances from the outlier would be very large, hence all p1j's would be small, so the position of outlier in feature space, the y1 has little effect on cost function, its position is not well determined.

To make them not so small, we can just add a background noise, or simply average two conditional probabilities.

Now, it is good time to start building tSNE. Consider these three points, this and this distance is 1, hence this distance is square root 2. Now, to represent this 2D data into 1D, we keep the 1 unit distances as it is, but the other distance must become 2.

So, let square root 2 is the distance corresponding to pij equal to 0.1. Hence, qij would be approximately 0.1, which should correspond to distance of 2. Hence, we need heavy tail distribution. The first thing that comes to our mind is Cauchy, which is t with 1 degrees of freedom. In general you can use any t distribution, and hence the name.

Now, let us have a better look at MNIST dataset again.

As you can see, nice structures are there, with 4 and 9 and 7s being together here. These numbers basically differ by only 1 line when you write them. You can see a cluster of 0s in the left, while a clusters of 1s in the right.

I have simulated from mixture of 3 population of 10 variate normally distributed variables. The next animation will show you how tSNE ouput changes after each iteration.

I have also visualized Winsconsin Breast cancer data, which contains 32 variables related to breast cancers. These variables are fed to tSNE and 2D output space is visualized. I have annotated the plot according to whether the tumor is benign or malignant.

As you can see, there are 3 phases of algorithm. In the first one, everything is clustered around origin, which is due to a L2 penalty term. This helps the points to explore all possible directions for minimization which helps to circumvent local minima.

In next phase, pij's are multiplied by some constant greater than 1. The effect is that no qij's is large enough to capture pij's. But the model is encouraged to model large pij's by large qij's. As effect, we see clusters forming.

Finally we have complete visualization as the constant is brought back to 1 and things become stable.

Let us see some more examples. As you can see, the global structure may vary, here it was straight line, but tSNE output shows some curvature, as it focuses on retaining local distances.

Some more example.

Example of how it performs on Trefoil knot.

The pros of tSNE is that we can use it with nominal and categorical variables as well, as long as we have some similarity pij which can be thought as probability like measure. tSNE is also robust to outliers that outlying datapoints remain outliers in output space.

Cons is that to compute the distances in each iteration we require n squared computations, huge for large datasets. Since tSNE does not output a function between Xi and yi, we cannot use it for classification, as we do not know where a new datapoint would lie in output space.

To reduce the computation as mentioned, we use Barnes Hut approximation. Using a quadtree we subdivide the output space. Here we have A, B, C all being distant from point I. So, we can approximate those individual distances by distances between I and the cell containing these points. Then multiply the force by 3.

This is algorithmic details of the same approximation idea.

Now we consider tSNE performed using Barnes Hut approximation on this large scale real dataset which contains 14 million photographs.

We can see that lots of blues on top right here. It contains airplane and ships. Here you see vehicles, Close to that we have electronic devices here, and then necessary everyday items, then lots of rounded things like clocks, utensils. If you go below, we find all of good looking foods, ending with fruits, which is close to insects, animals and birds, basically wildlife. Here you see lots of domestic animals, and at center you see human faces.

Now suppose you are plotting words. Consider Bank and River, they should be close. Consider Bank and Money, they should be close. By triangle inequality, River and Money would be close, which does not make sense. So, we need multiple layers of output space to represent it properly.

We define something similar to as before, but with a probability pi_i(m) that yi would appear in map m. And model these similar to logistic model.

Here we see an application of it on real dataset. China is appeared with Porcelin, Bowl, Plate, Dish etc. In another map it appears with Dynasty, Rome, Empire etc.

Now, we have some recent results that proves the worth of tSNE. First it defines something called gamma spherical and gamma separable data, which determines what are the natural clusters of a dataset. Then it defines something called 1 minus epsilon visualization, where epsilon is kind of error in the visualized clusters. If there is no error it is full visualization. It has been proved that if tSNE output is 2D, then after sufficient iterations, tSNE leads to full visualization for gamma spherical and gamma separable data.

Here are some Future scopes. Thank you!