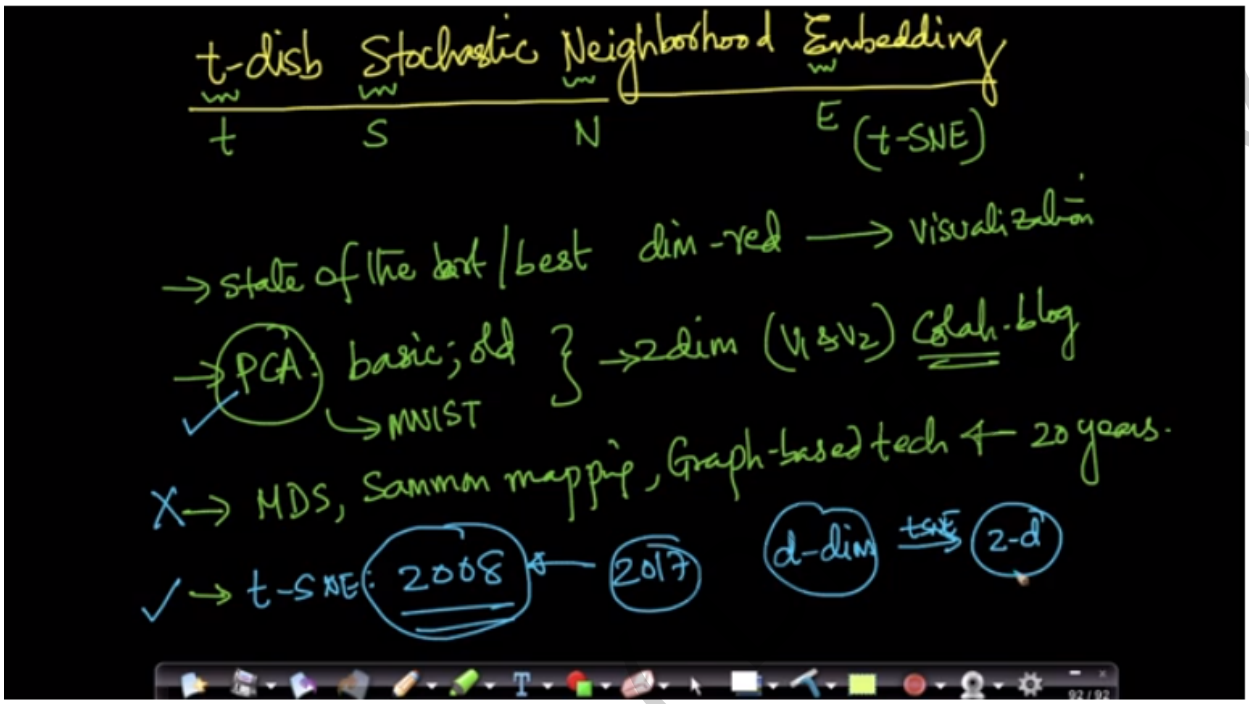
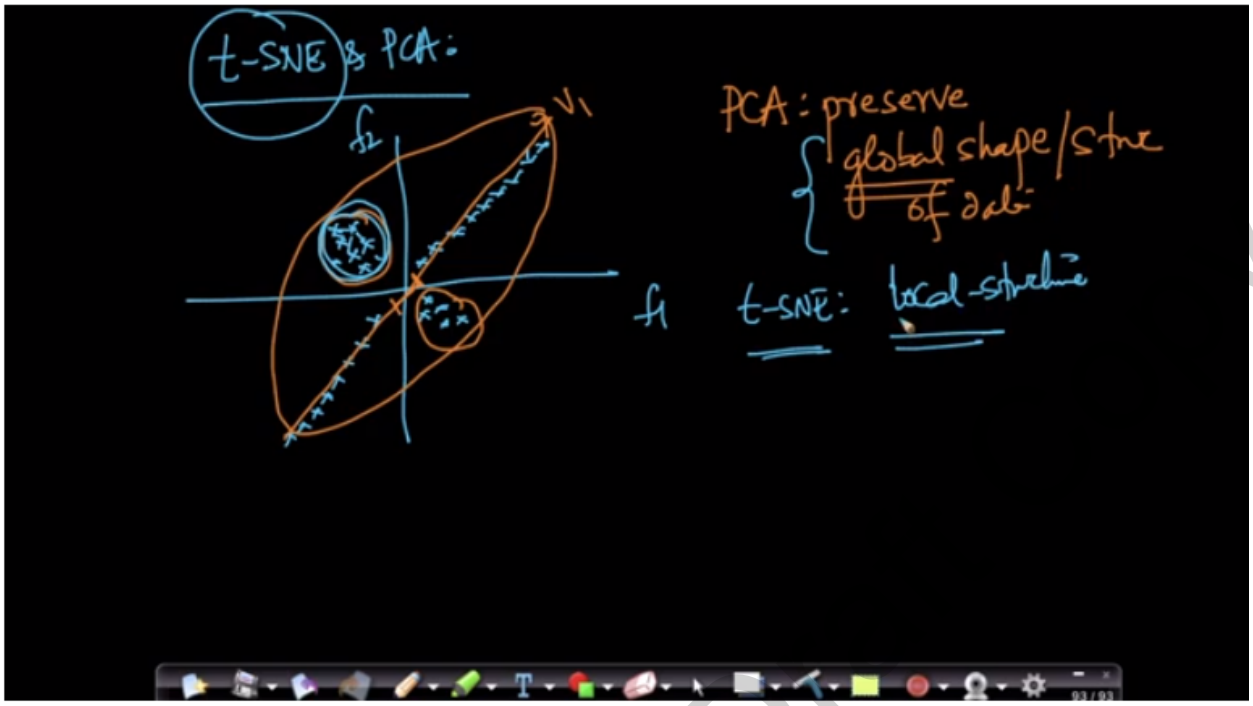
27.1 What is t-SNE?



Timestamp: 3:52

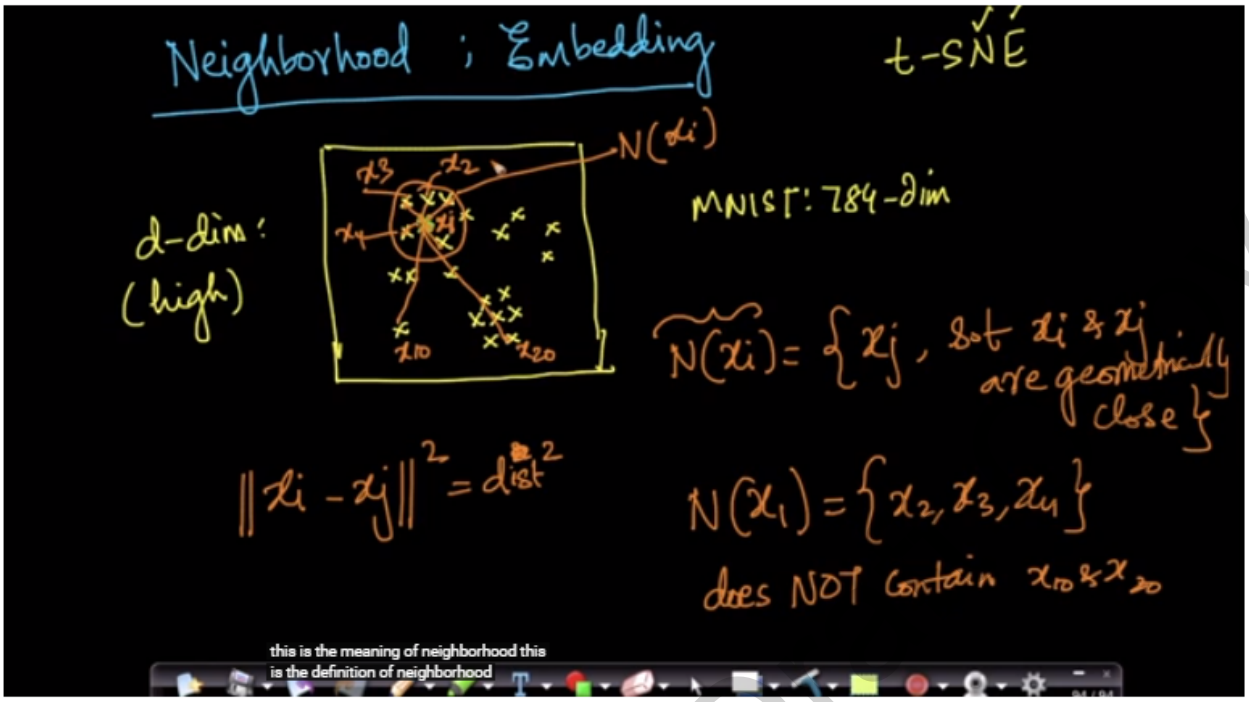
t-SNE is the state of the art technique for visualization, There exists other visualization techniques like MDS, sammon mapping,etc but t-SNE performs comparably well than most other techniques.



Timestamp: 6:21

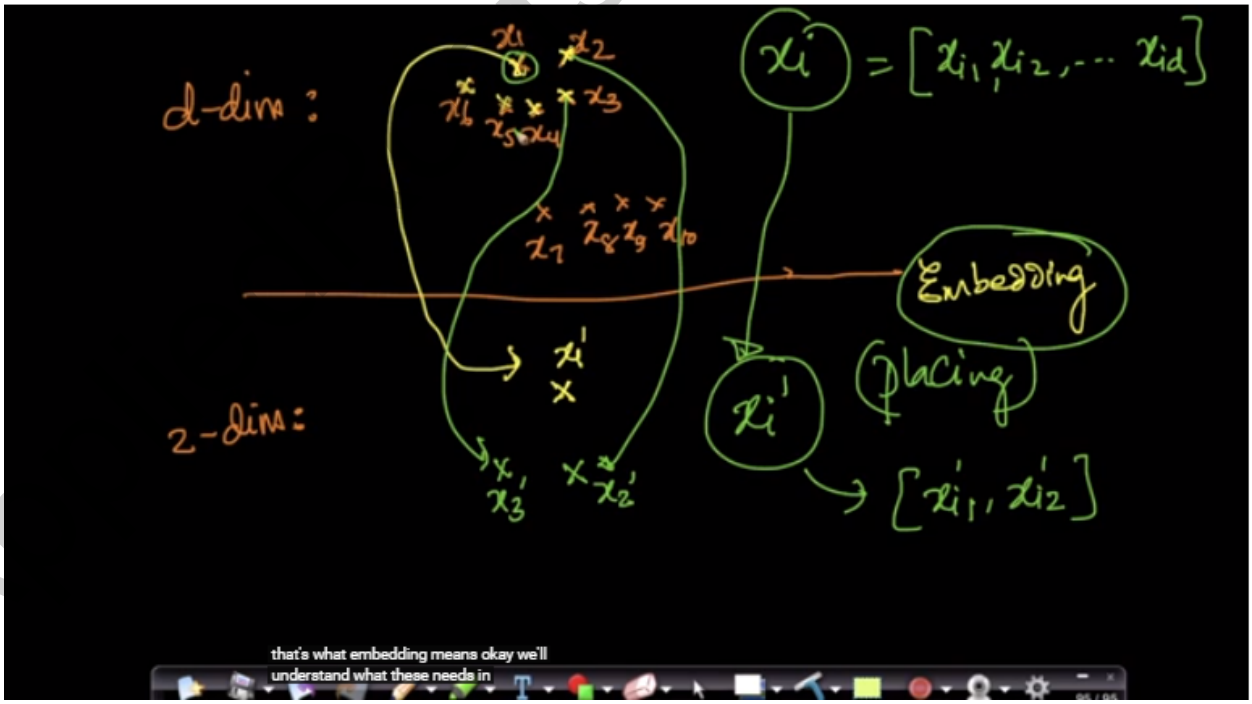
One of the key disadvantages of PCA that we discussed in “limitations of PCA” is that when they are nice clusters of points that are present in the original space when they are converted to lower dimensions, these get mixed up with other points. So PCA preserves only global structure and doesn’t preserve local structure. By changing parameters in t-SNE we can make it preserve local structure as well.

27.2 Neighborhood of a point, Embedding



Timestamp: 3:40

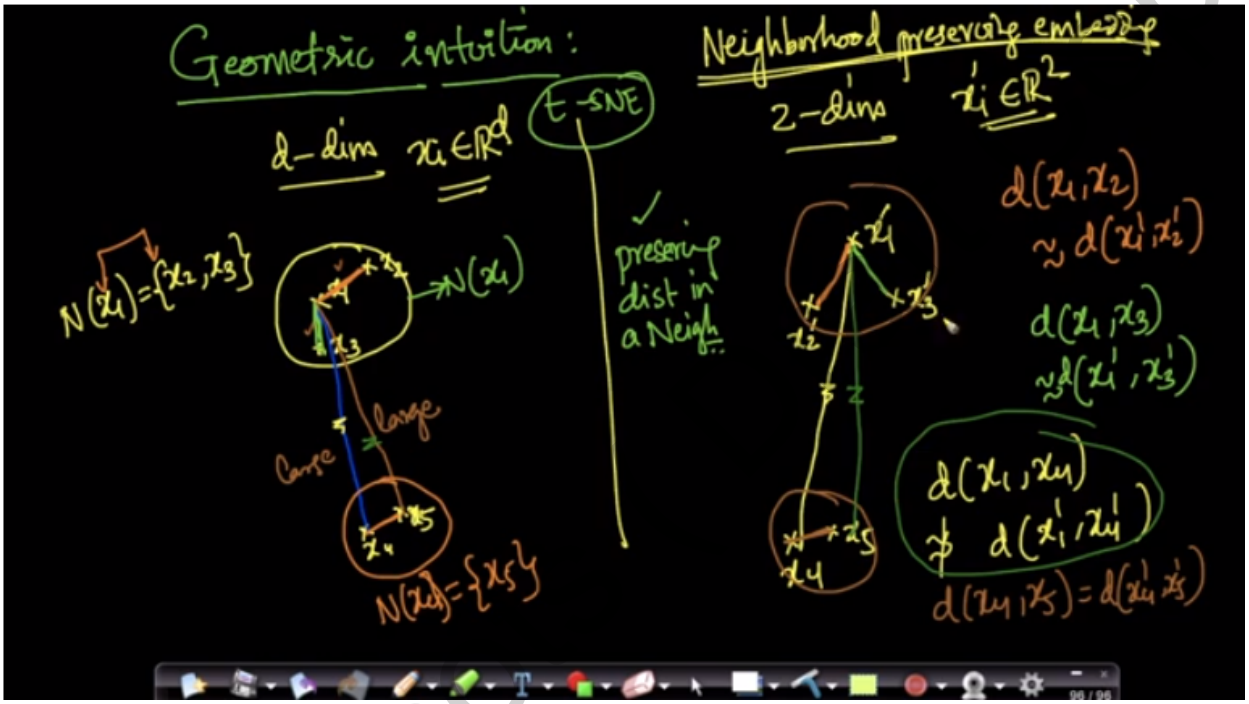
For any point, say x1, the points around x1 are called the neighborhood of point x1. In our case the neighborhood of x1 is the set of points x2,x3,x4,etc. The neighborhood of any point xi is the set of all points that are geometrically close to the point xi.



Timestamp: 6:14

For any point in the high dimensional space, if we find the corresponding point in the low dimensional space, then such a thing is called embedding.

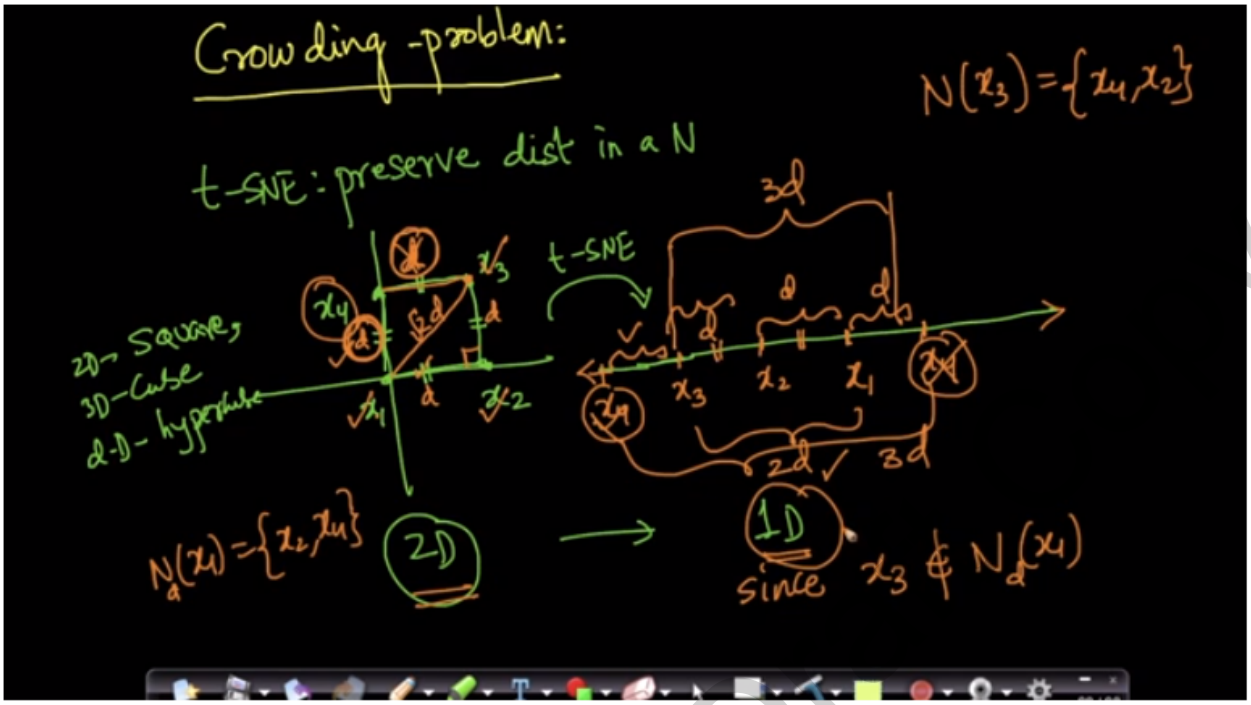
27.3 Geometric intuition of t-SNE



Timestamp: 6:18

Geometrically, what t-SNE does is, it tries to preserve the distance between the points in the same neighborhood even in the embeddings. What that means is that points that the distance between points belonging to the same neighborhood is almost similar to the distance between the points in the smaller dimension as well, the same is not guaranteed with the points that are not in the same neighborhood in the original space.

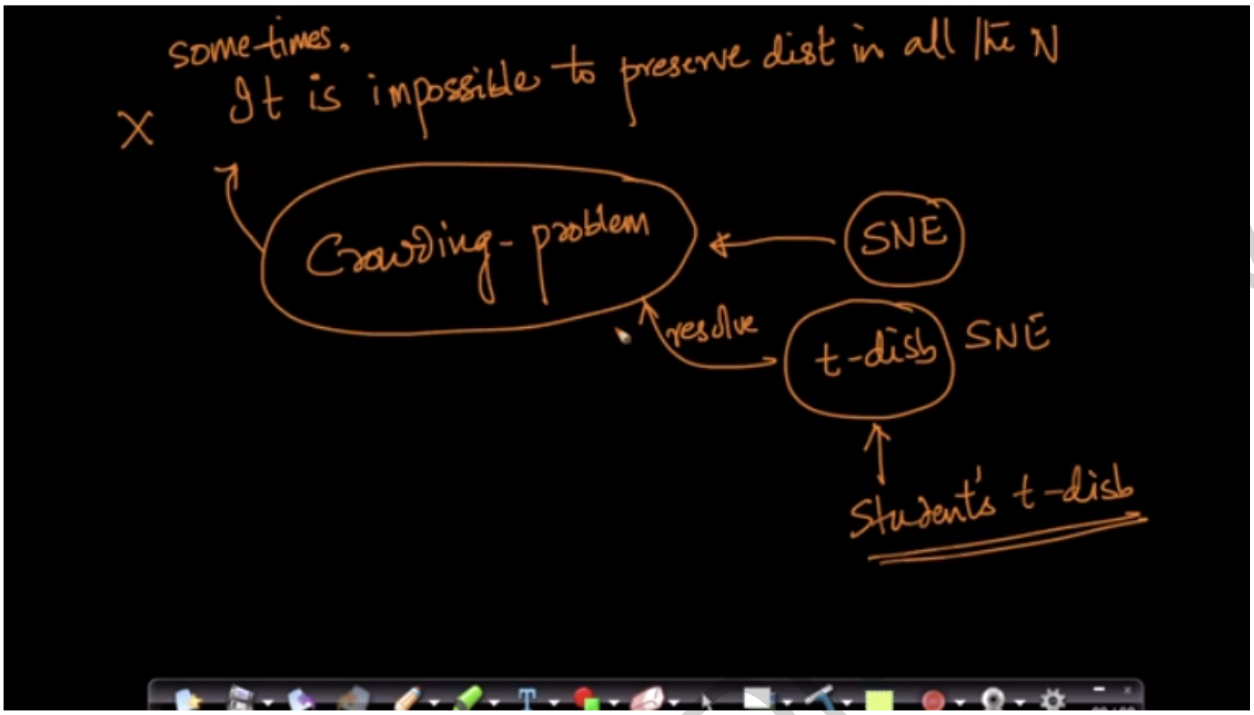
27.4 Crowding Problem



Timestamp: 5:47

Consider a scenario as above where we have 2-D dimensional data and we have to reduce it 1-D. Consider the points x1,x2,x3 and x4 in the original 2-D space and let us say that d is the neighborhood distance i.e two points with distance less than or equal to d, are said to be in the same neighborhood. So in the above figure x4 has neighbors x1 and x3, x3 have neighbors x4 and x2,etc.

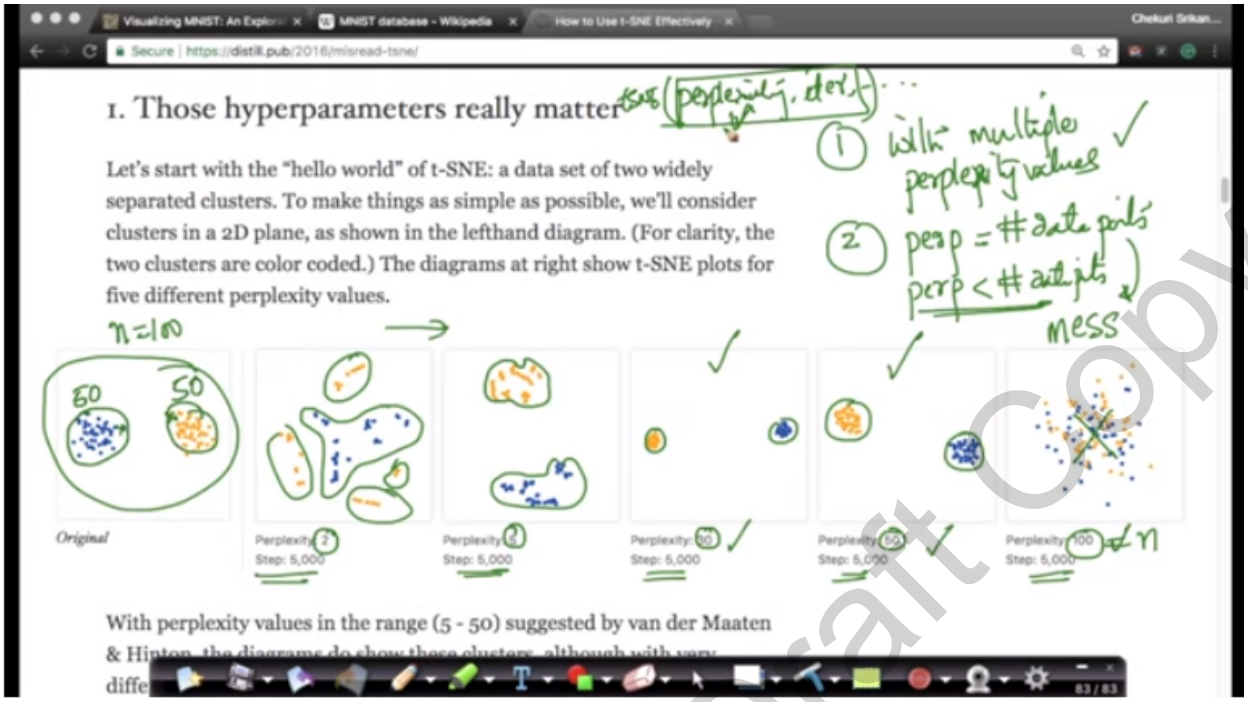
So in our 1-D embedding, we have to place x2 and x1 within a distance of d, since we have to preserve distance between points that belong to the same neighborhood, next we place x3 within d distance of x2 since x2 and x3 belong to same neighborhood, now we have to find a place for x4 in the 1-D, but we cannot place it correctly. This is because if we place it in d distance from x3, then it would be of distance > d from x1. Hence we won't be able to preserve the distance between x4 and x1 even though they belong to the same neighborhood. The same will be the case even if we try to place it at d distance from x1. Here we are suffering from crowding problem.



Timestamp: 7:44

Sometimes, it is impossible to preserve distances of all points in the neighborhood. This problem is called the crowding problem. SNE used to suffer from this crowding problem, later t-SNE was introduced to resolve this issue, t-SNE doesn't completely solve the problem, but it tries its best to minimize the errors due to this problem.

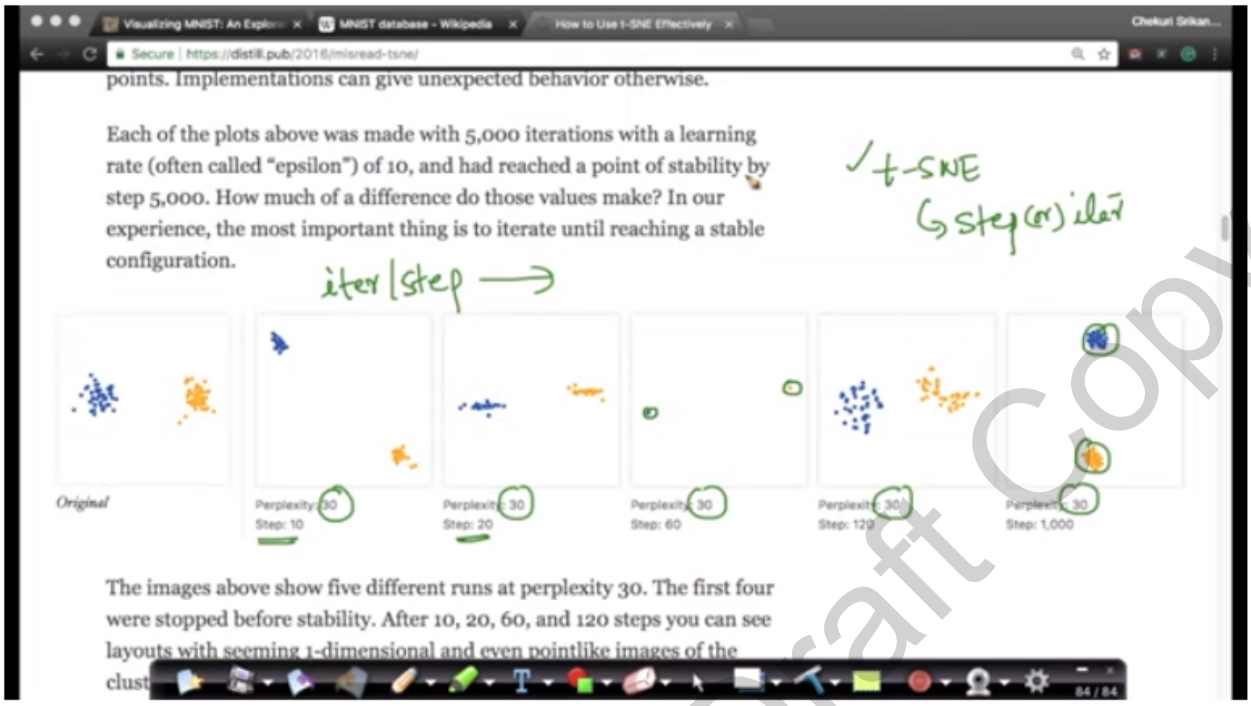
27.5 How to apply t-SNE and interpret its output



Timestamp: 15:30

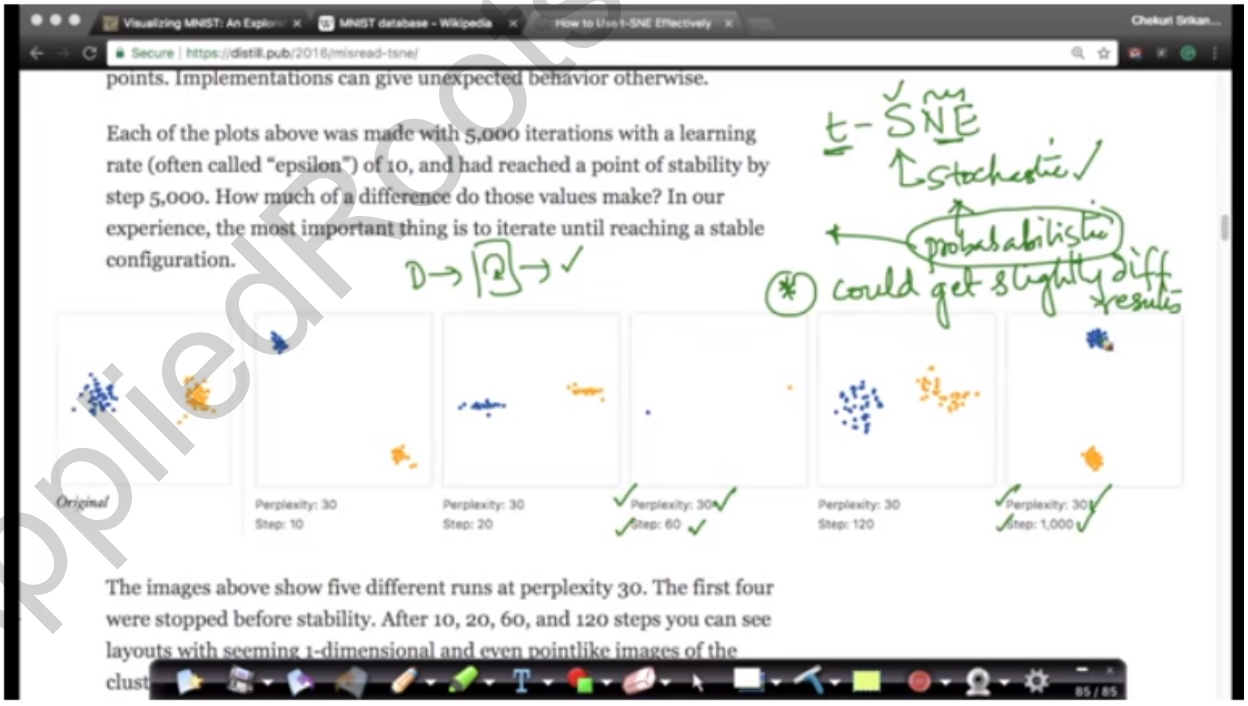
Note: In all the examples of this video, even though t-SNE is used to visualize higher dimensional data in 2-D, we try to visualize the data from 2-D itself post applying t-SNE, to give us the intuition of how t-SNE works.

1) Notice that with multiple values of perplexity you get different visualizations and the visualizations are close to the original data only for perplexities 30 and 50. The lesson here is to run t-SNE with different values of perplexity to get the right visualization
2) Never run t-SNE with perplexity=total_number_of_points,running t-SNE with perplexity as total number of points will create a mess as it tries to preserve the distances of all the points not only the neighborhood when ran with perplexity as total number of points



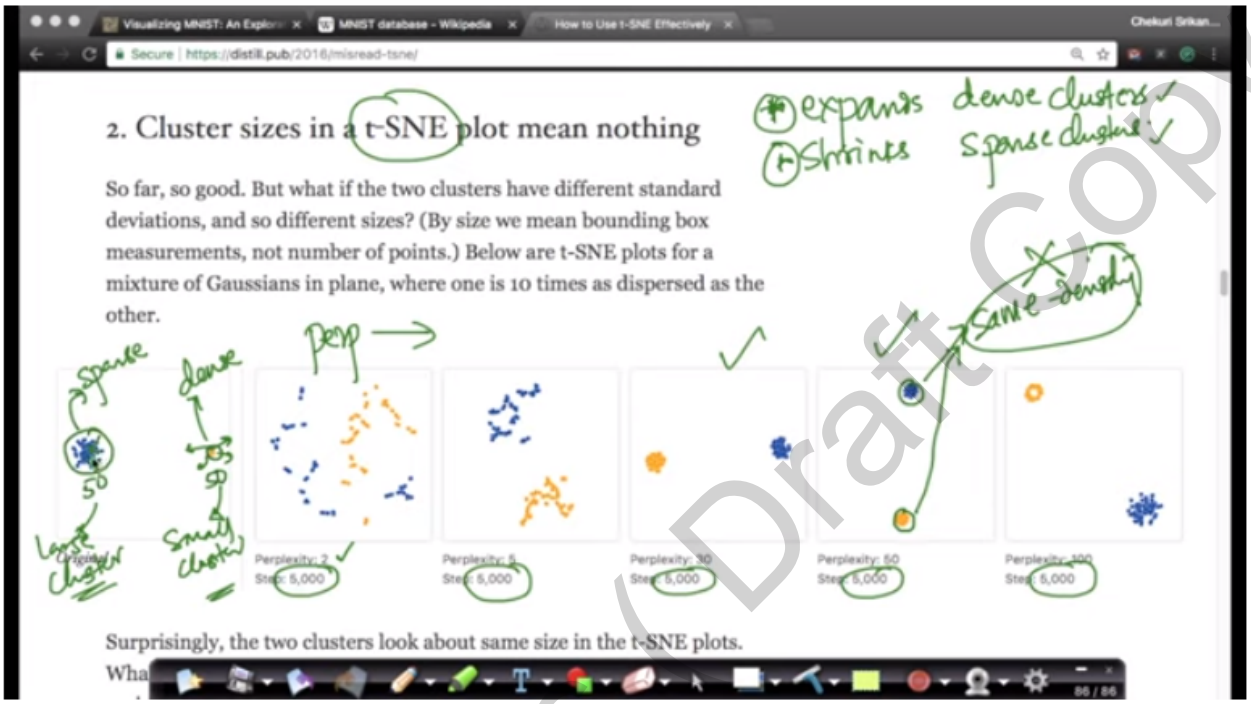
Timestamp: 16:49

Always run t-SNE for multiple iterations(step), until unless you see that the shapes are stable even with further iterations as shown in the figure.



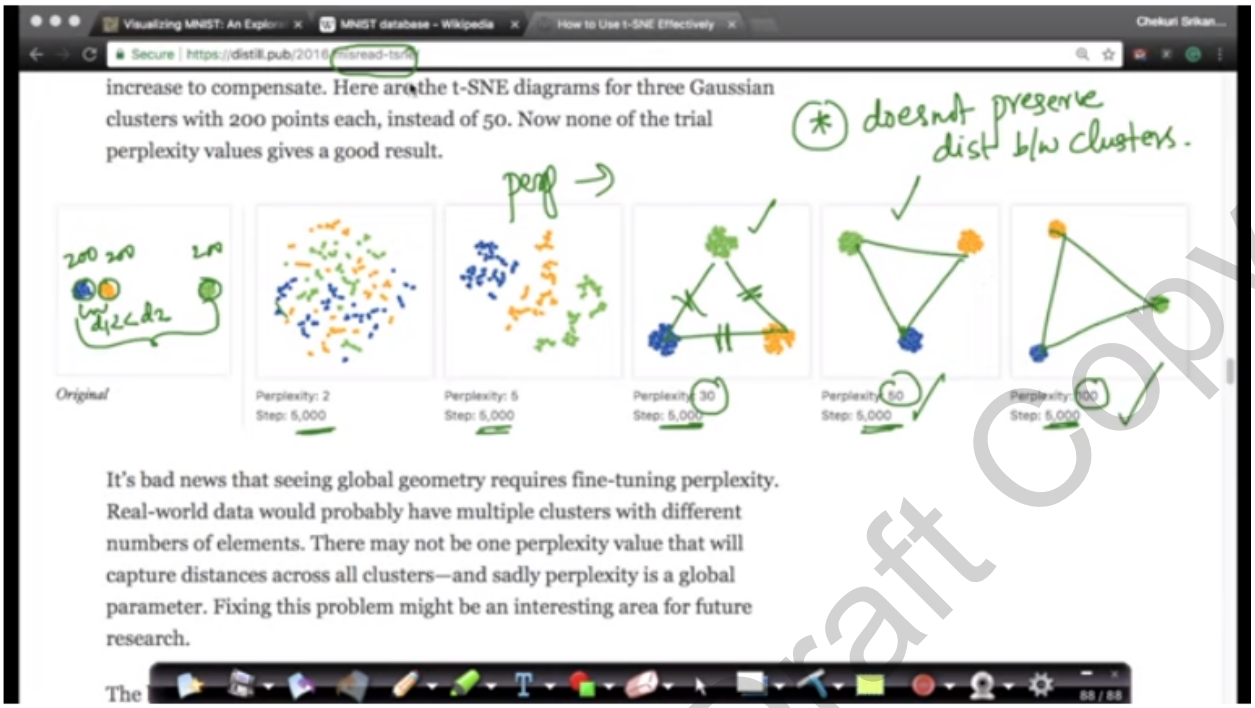
Timestamp: 18:45

Always run t-SNE multiple times by fixing the perplexity and number of iterations. Since t-SNE internally is a stochastic algorithm, even with the same perplexity and number of iterations it may give slightly different results.



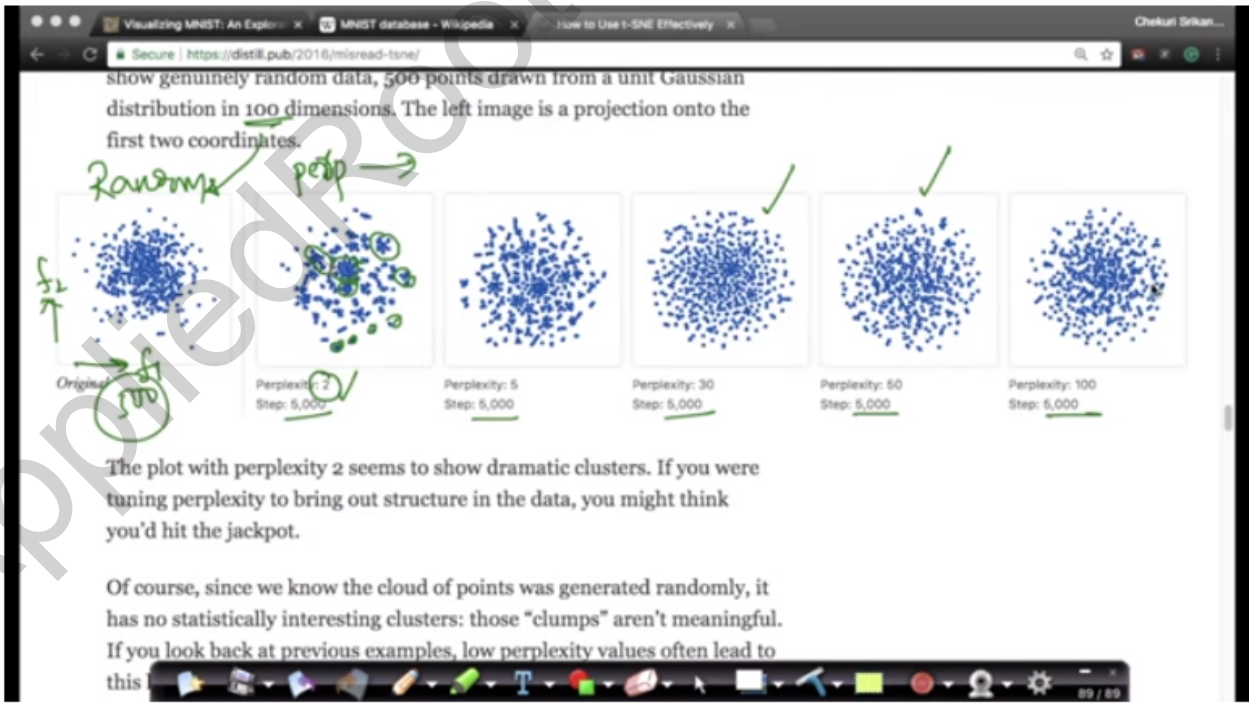
Timestamp: 22:07

You can’t come with the same density conclusion with t-SNE. If two clusters have same density in the embeddings then there is no guarantee that they have the same density in the original space. t-SNE typically tends to expand dense clusters and shrink sparse clusters



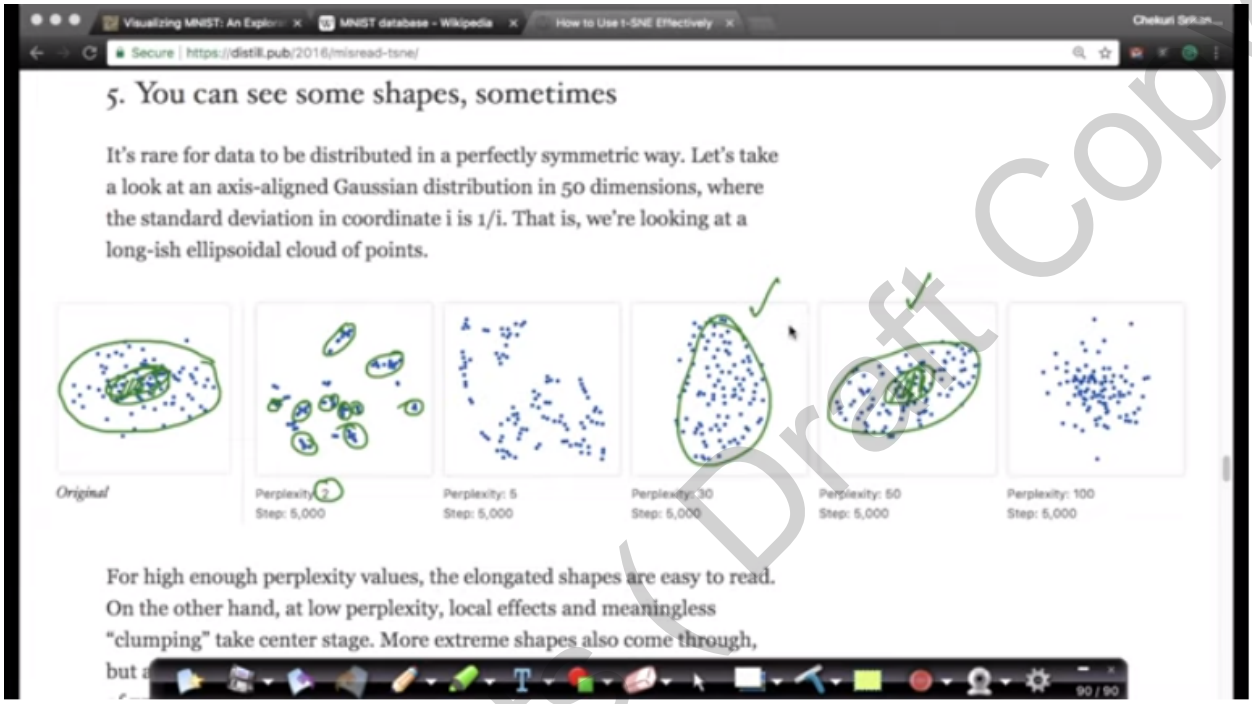
Timestamp: 22:59

t-SNE doesn't preserve the distance between clusters. You can see in the above figure that cluster1 and cluster2 are closer than cluster3 but after running t-SNE , it is showing that they are of equal distances.



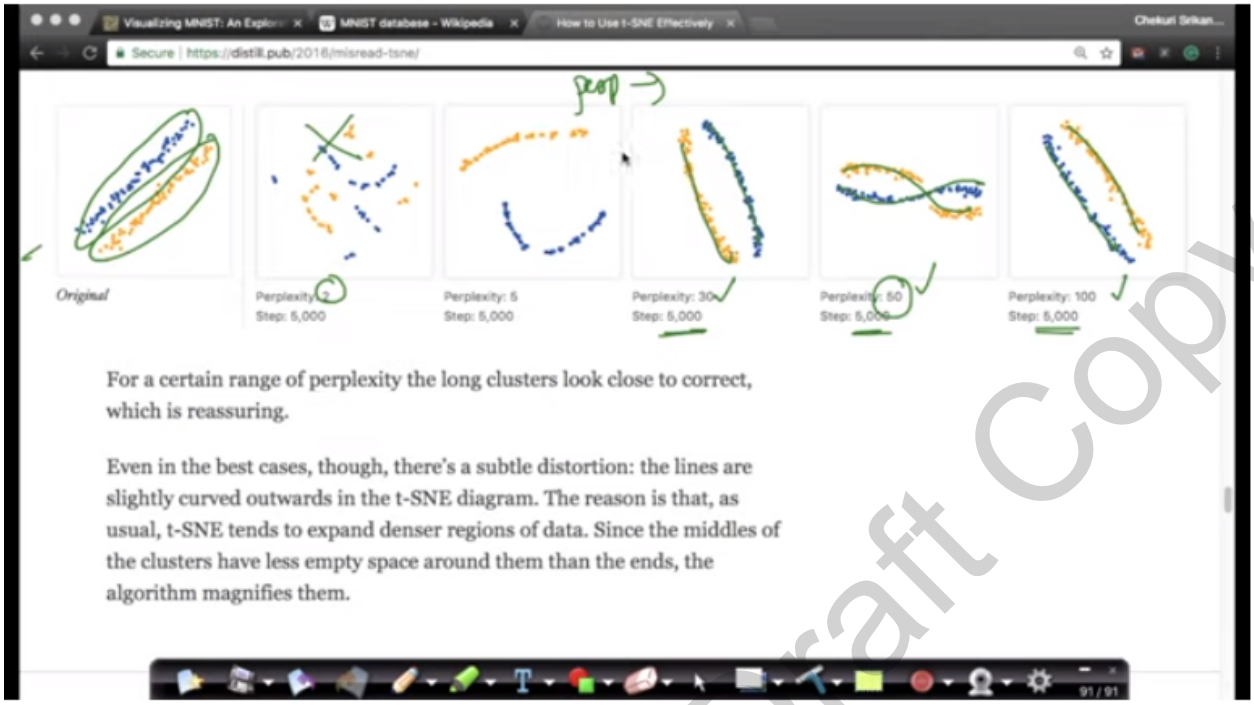
Timestamp: 27:51

With t-SNE you might come up with wrong conclusions if you run with a single perplexity value. In the above picture, we can see some clusters when run with perplexity=2, but the original data is completely random. Running with other perplexity values shows that the data is indeed random. Hence running t-SNE with different perplexity values is preferred.



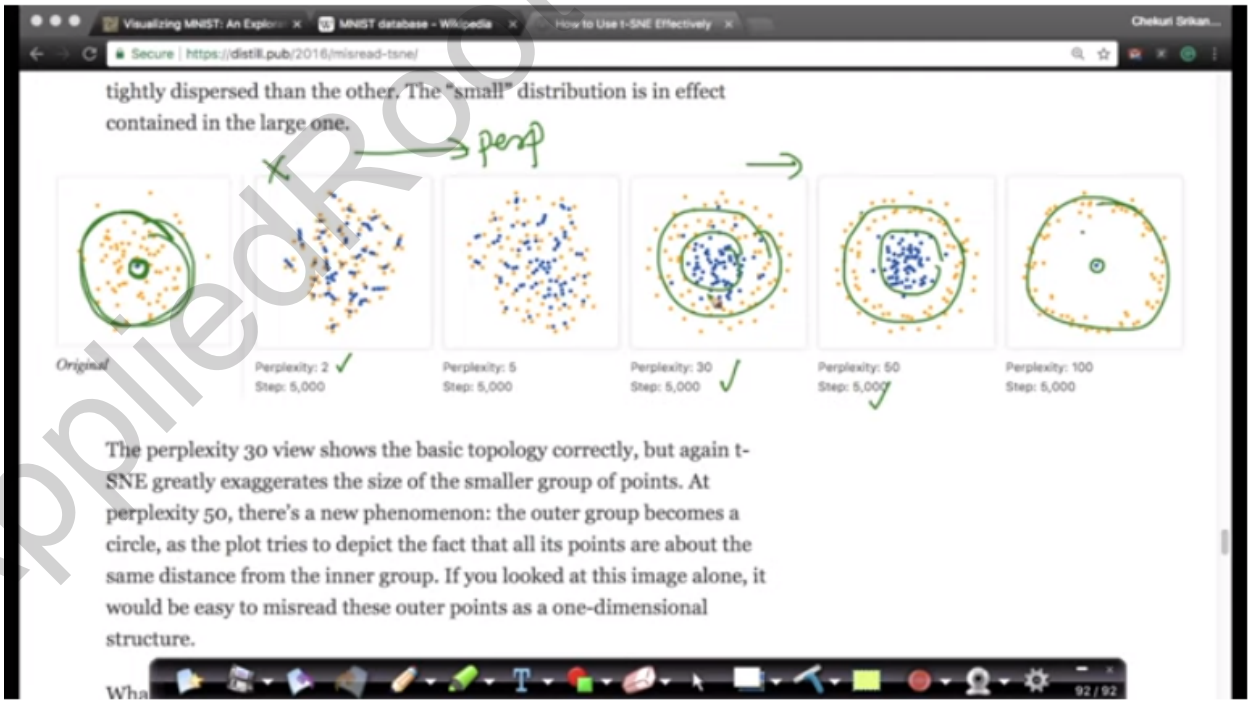
Timestamp: 28:52

Similar example, do not run t-SNE with single perplexity values rather run with different perplexity values as there is a risk of concluding that there are certain shapes of clusters with the points in our data when they are not.



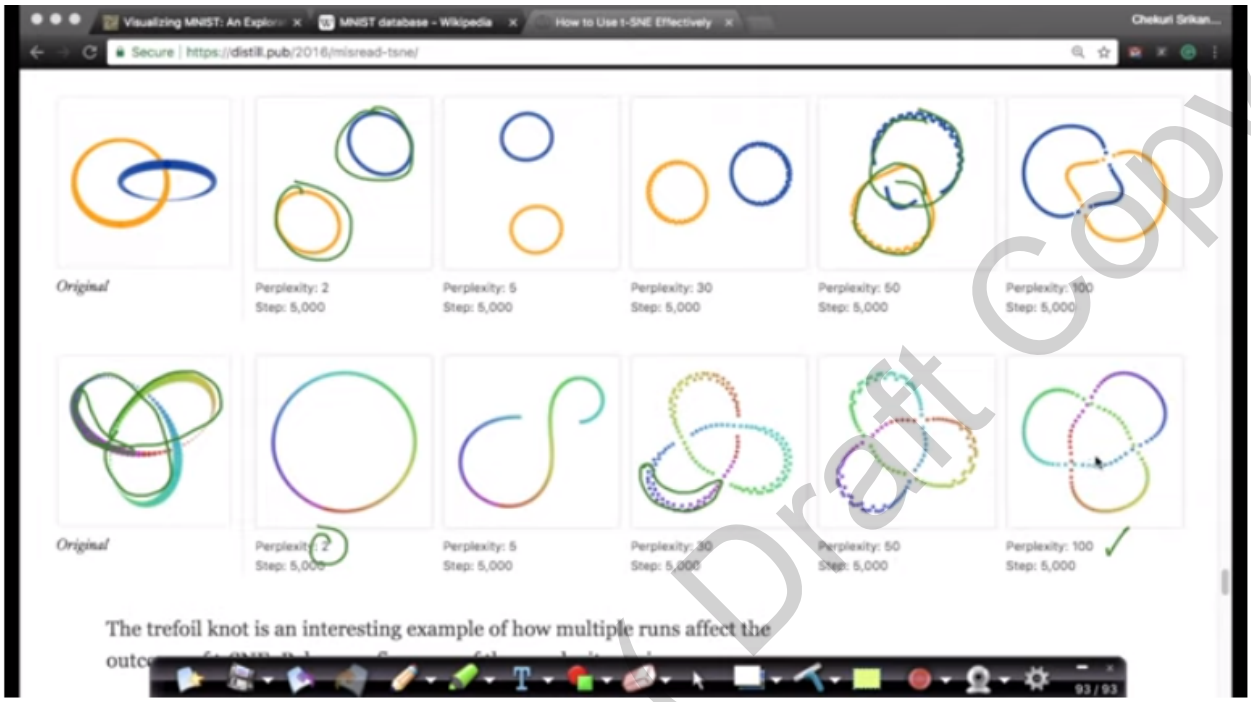
Timestamp: 30:36

Again, never run t-SNE with a single perplexity value, sometimes with some perplexity values we may get different shapes than the original data. Another takeaway is to run t-SNE multiple times with the same perplexity value and number of iterations.



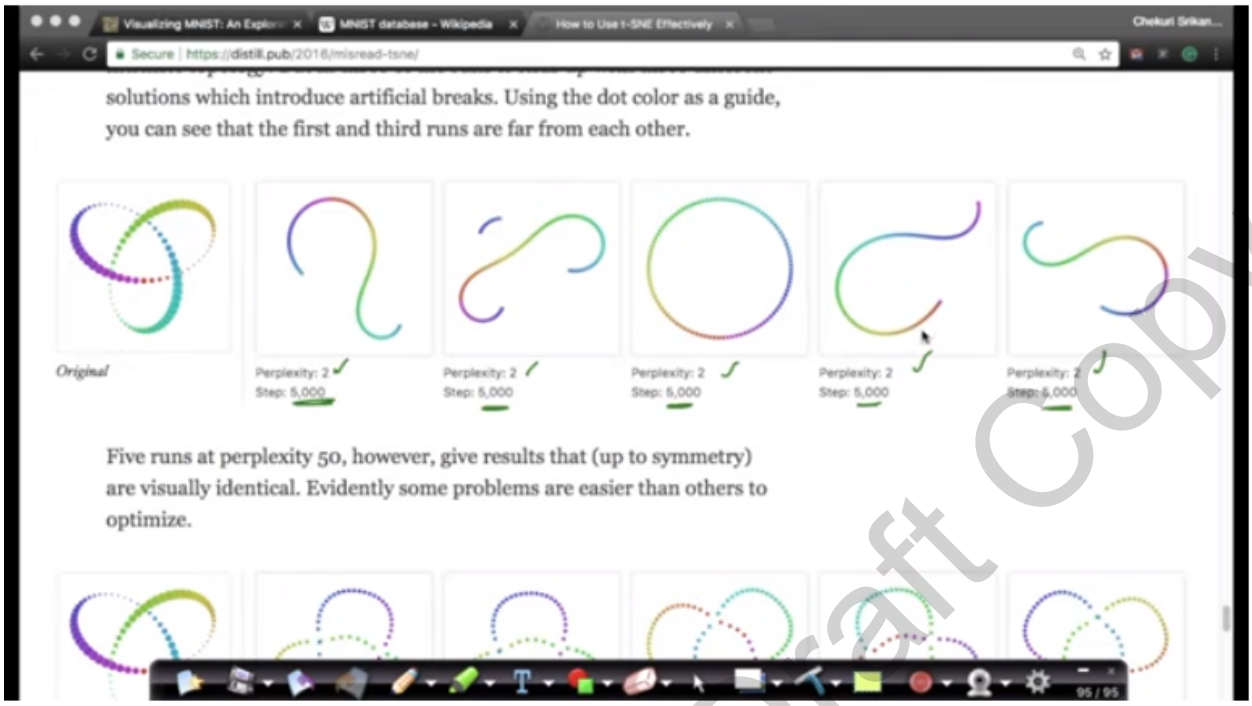
Timestamp: 31:59

Running with different perplexity values gives us the sense of overall topology of our data. We can see from the above data that as we increase the perplexity values we begin to realize that blue points lie inside orange points in the original data.

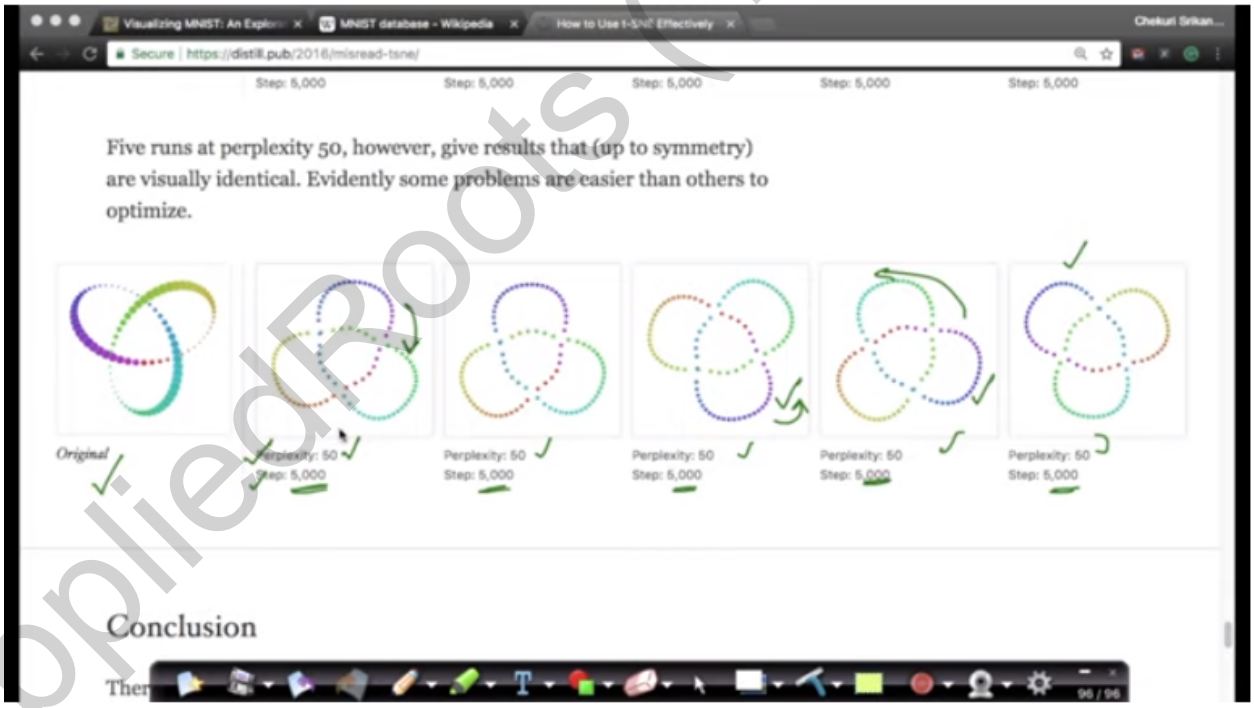


Timestamp: 32:38

From the above figure we can observe that with increasing perplexity, the topology of the visualization looks more and more similar to the topology of the original data.



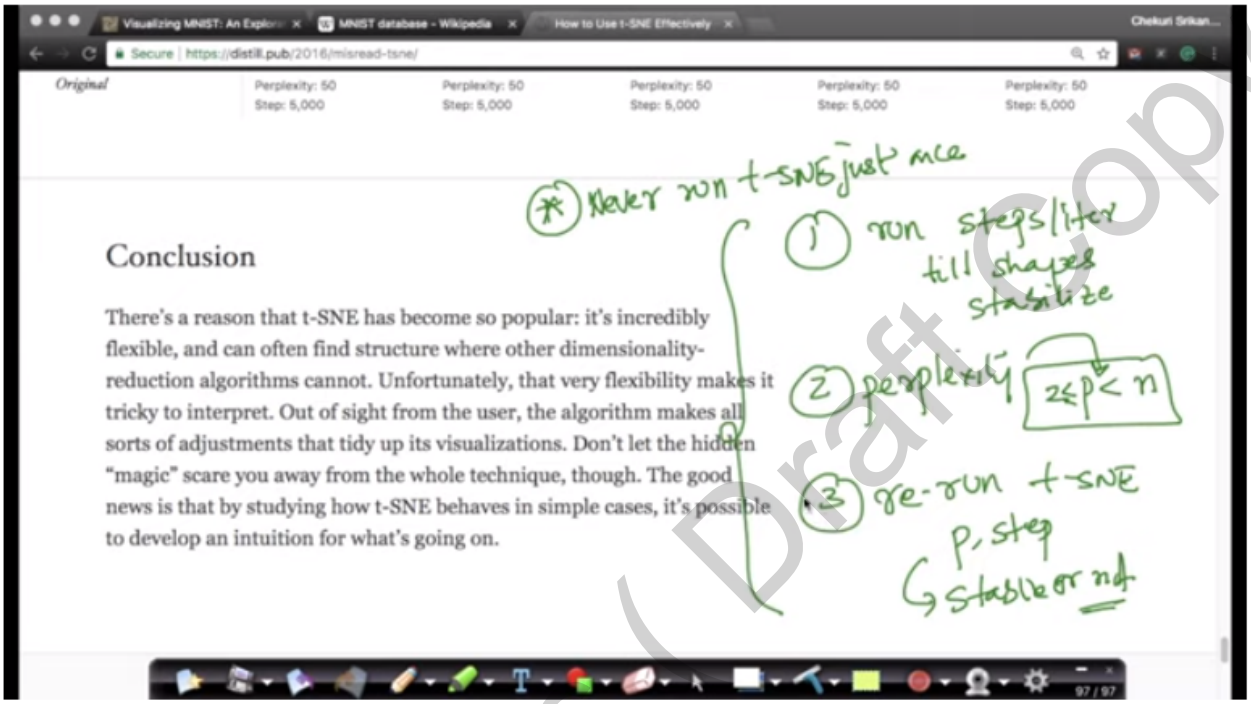
Timestamp: 33:39



Timestamp: 34:25

In the first figure we can see that with the same perplexity=2 and same number of iterations, we are seeing different shapes and the shapes are not stable. This means perplexity=2 is not the right perplexity value for the dataset.

Instead if we run t-SNE with perplexity=5 and same number of iterations, we see that the shapes are more or less stable. Hence it is always necessary to run t-SNE multiple times with the same perplexity value and number of iterations.



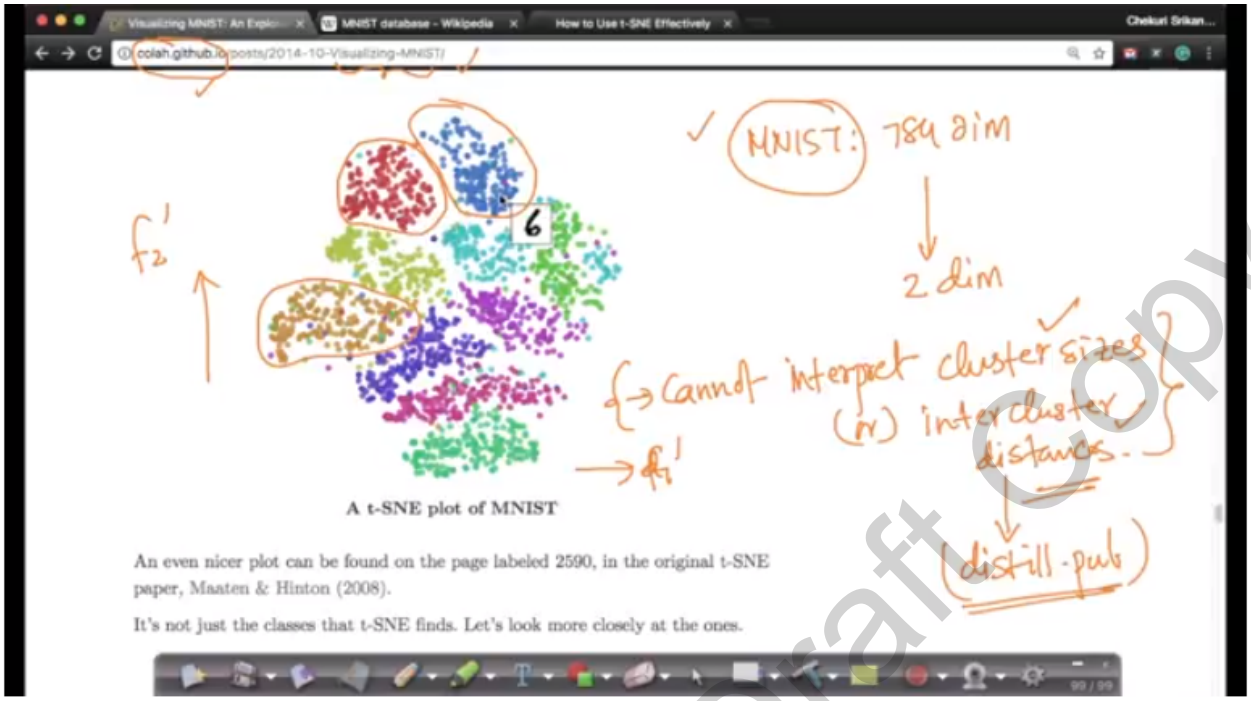
Timestamp: 36:03

To conclude,

1) Always run t-SNE for more iterations until the shapes stabilize
2) Try with different perplexity values in the range 2 to n (not including n as it creates mess).
3) Rerun t-SNE with the same number of iterations and perplexity value and check if the shapes are stable or not

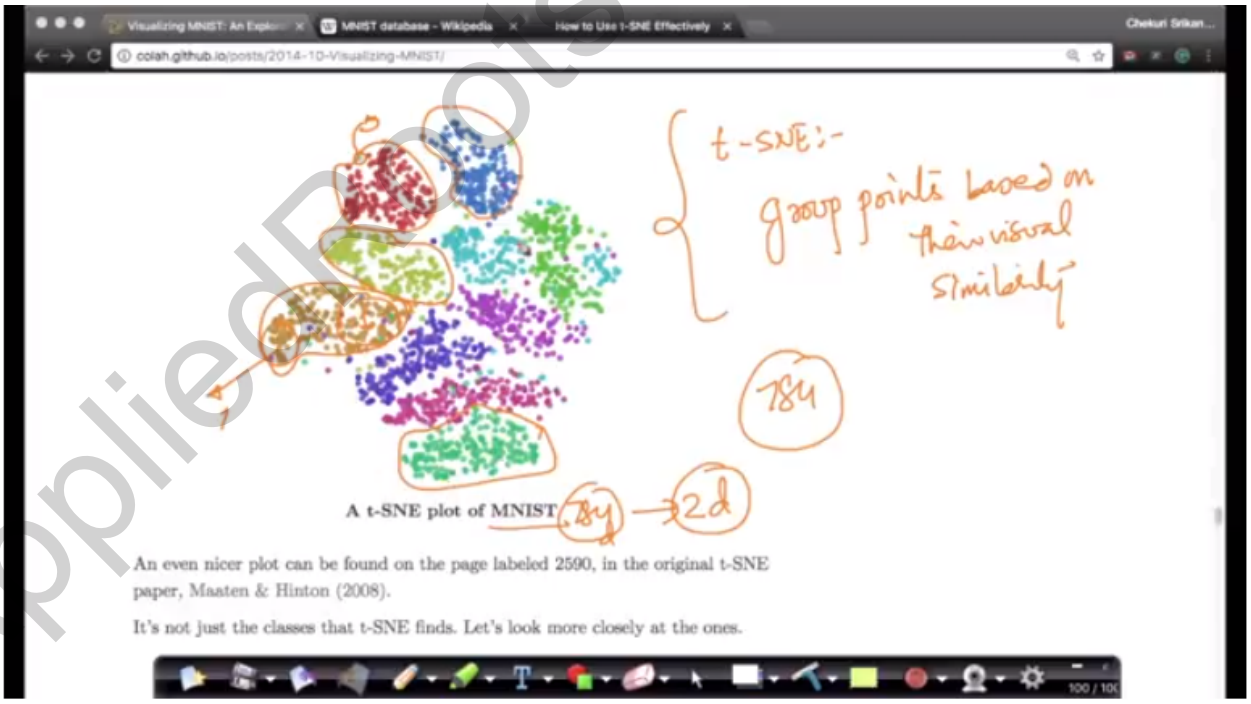
Never run t-SNE just once.

27.6 t-SNE on MNIST



Timestamp: 3:47

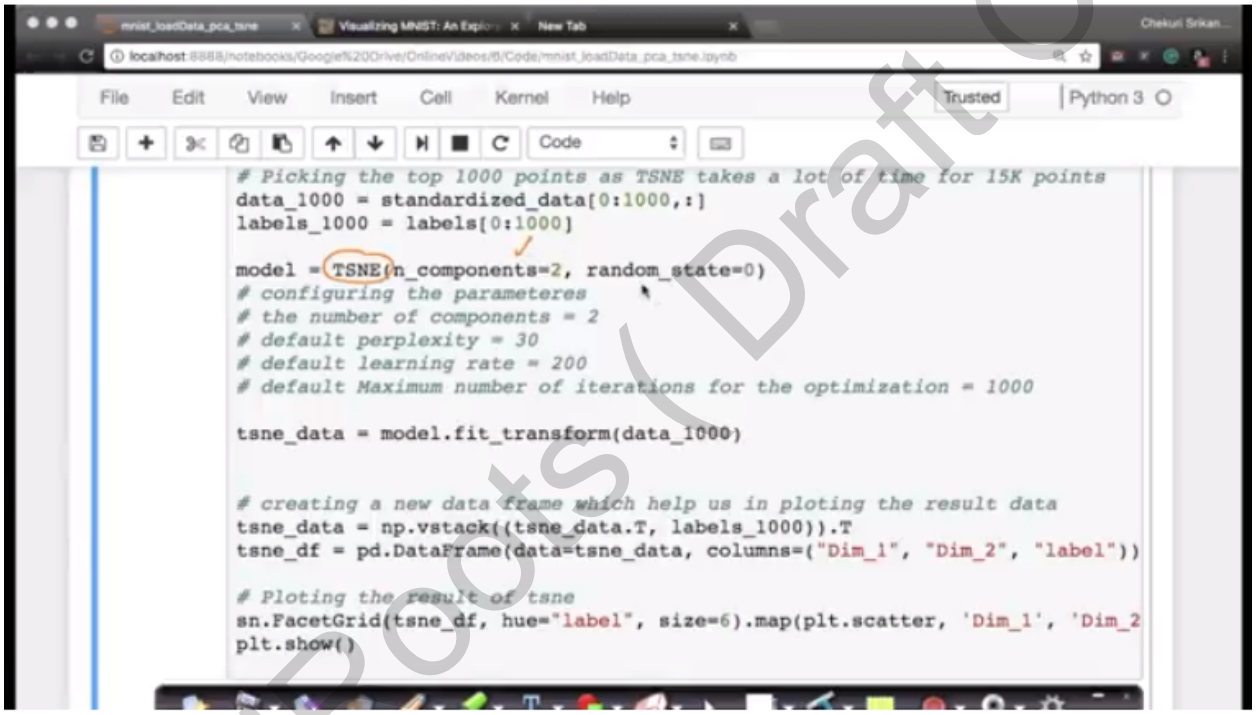
The above plot shows t-SNE applied on MNIST data. We cannot interpret cluster sizes or inter cluster distances using t-SNE.



Timestamp: 6:11

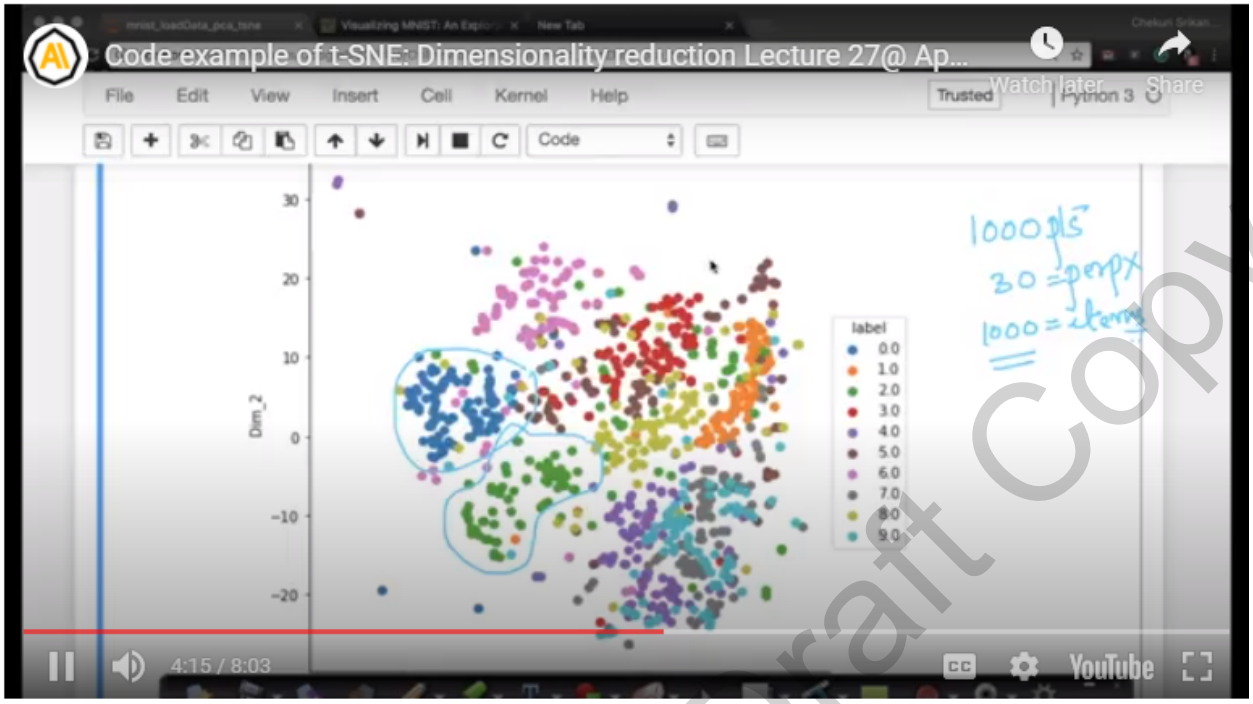
If we visualize the data points that are close together we can see that they are grouped based on visual similarity. If we see the visualization above we can see that most of the points of each class are well separated and grouped together, therefore we can easily fit simple machine learning models to separate these points.

27.7 code example of t-SNE



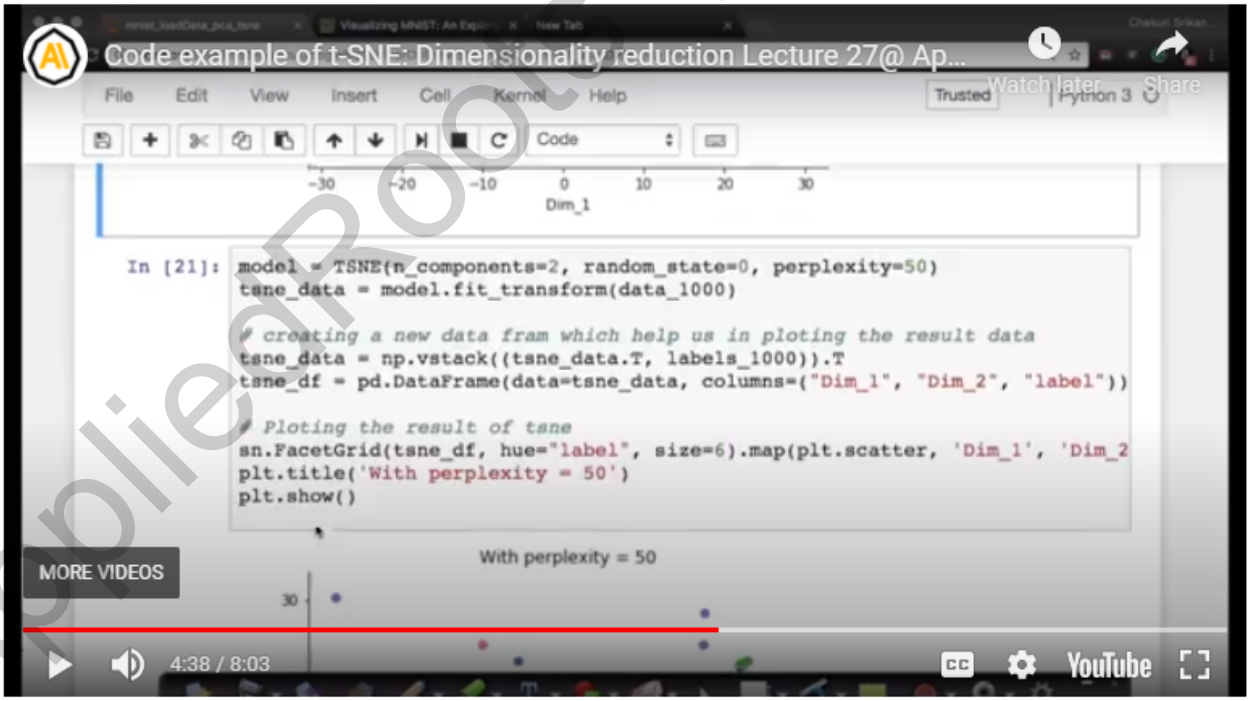
Timestamp: 1:39

t-SNE was tried in the first 1000 data points of MNIST dataset using the code above with default values of perplexity and number of iterations which are 30 and 1000 respectively, we got the below plot.



Timestamp: 4:14

Now t-SNE is run with perplexity=50, with 1000 iterations using the below code



Timestamp: 4:38

We got the below visualization for the same as below

Timestamp: 5:00

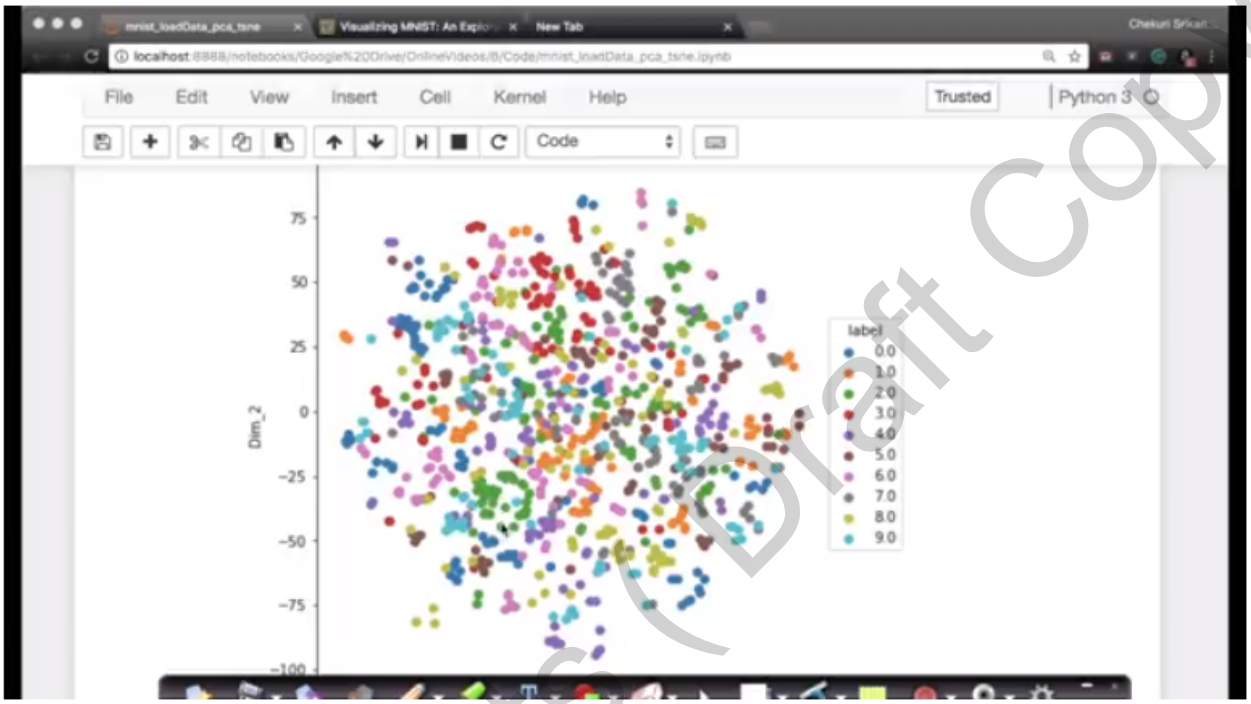
This plot looks similar to the plot with perplexity=30, the visualization is more or less similar.

Now t-SNE is run with perplexity=50 and the number of iterations to be 5000, we got the below visualization

Timestamp: 5:26

Even with 5000 iterations we got similar visualization meaning that our visualization is stable. Notice that with only using 10K data points we got the above visualizations, with using more data points the visualization would be much more better.

Now we try with perplexity=2, then we get the below visualization.



Timestamp: 6:49

From the above visualization, we can see that perplexity=2 may not work well as it fails to separate the data points.