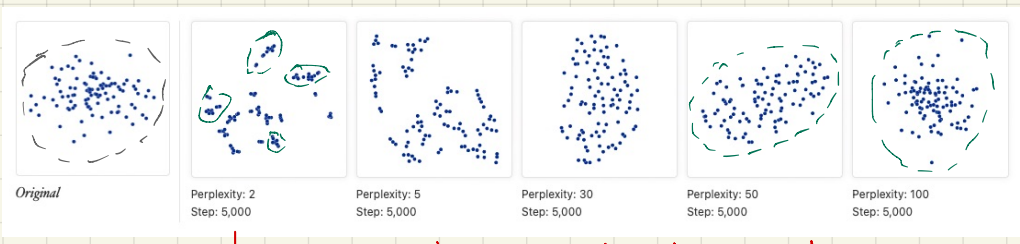
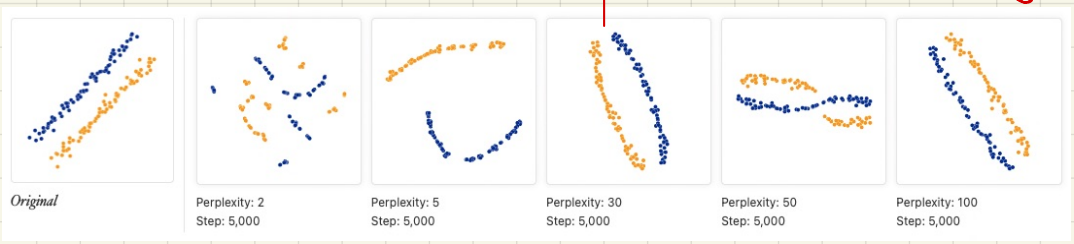
Case V :- You can see some shapes, sometimes



↳ Never give conclusion by looking at the small "perplexity" value.

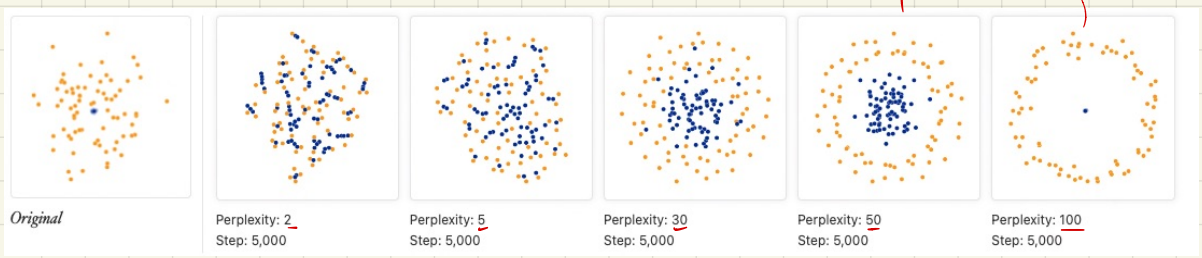
→ Never stop here. go further till you get stablize shape.



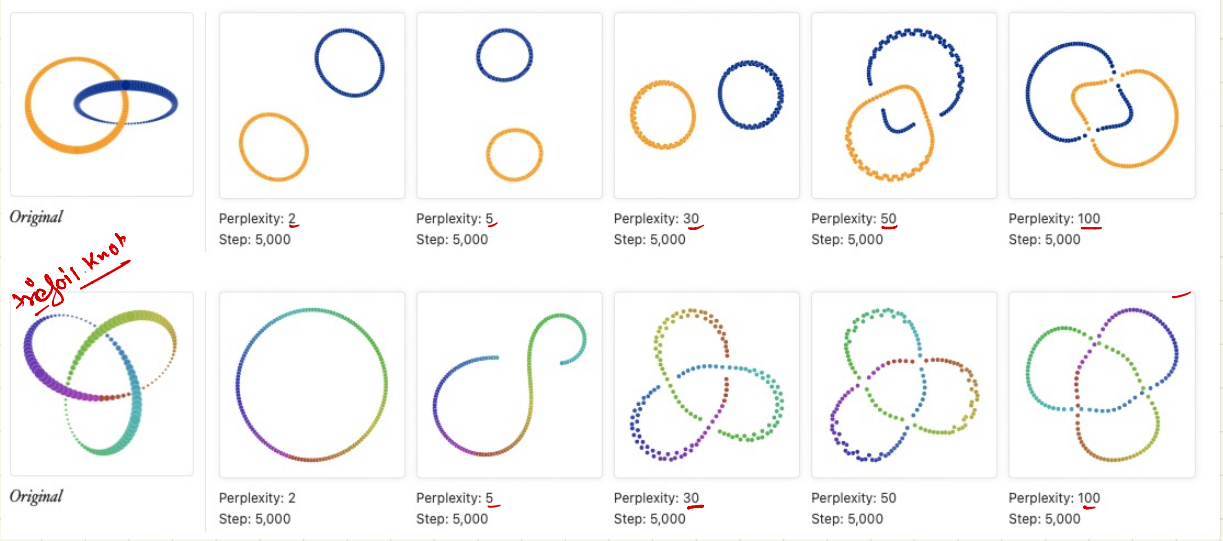
always inc^e you perplexity and see and always rerun.

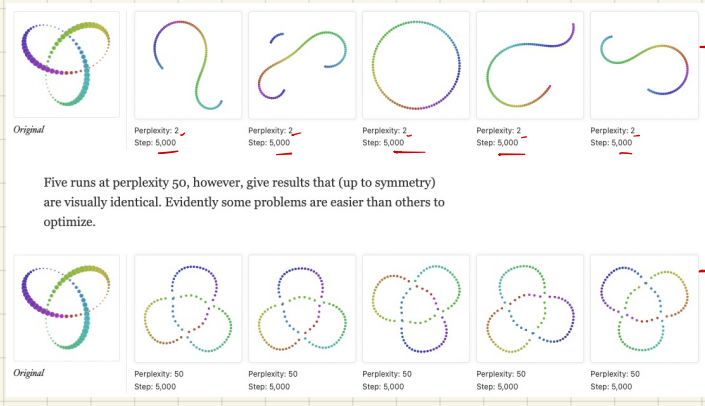
→ (study of shape)

Case-VI : For topology you may need more than one plot.



→ you will not get the exact shape, but you can get the sense of shape.





→ every run I'm getting different shape. so, it's not stable.

→ here, in every 'run' I am getting stable shape. So, now I know this perplexity and steps are stable.

Note :- - Always run steps/iter till shapes stabilize

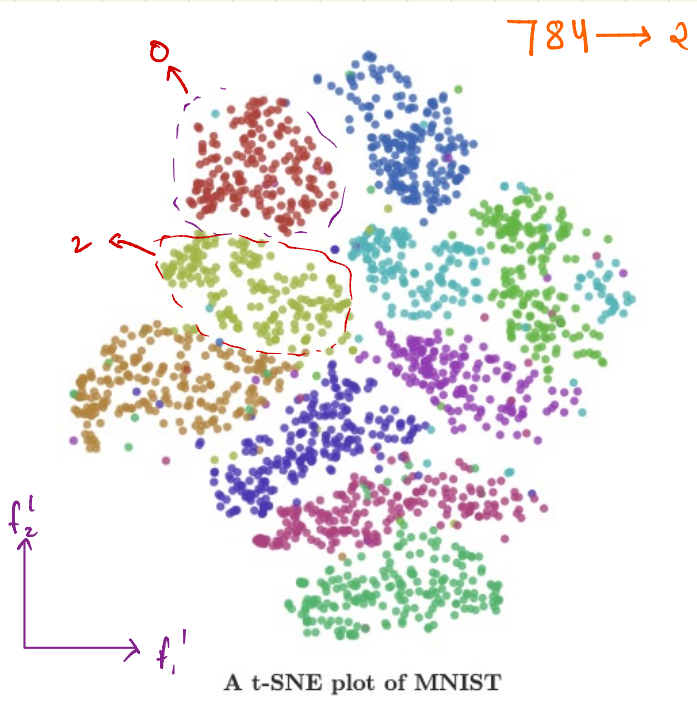
- Never run t-SNE once (run multiple times)
- perplexity $2 \leq P < n$
- re-run t-SNE with fix plexity & fix step → to see you are getting similar shape everytime or not.

* Epsilon → you can ignore it (optimize) → How fast you must change to get shape.
  ↳ as Epsilon inc" by the time you will reach smaller no. of iterations you will solve the problem. (Not always guarantee)

t-SNE on MNIST

Cristopher olah

784 → 2 dim.



A t-SNE plot of MNIST

A t-SNE plot of MNIST ones

The ones cluster is stretched horizontally. As we look at digits from left to right, we see a consistent pattern.

/ → / → / → | → \ → \

They move from forward leaning ones, like /, into straighter like |, and finally to slightly backwards leaning ones, like \. It seems that in MNIST, the primary factor of variation in the ones is tilting. This is likely because MNIST normalizes digits in a number of ways, centering and scaling them. After that, the easiest way to be "far apart" is to rotate and not overlap very much.

→ we cannot interpret cluster sizes or inter-cluster distances. (so I can't say '0' are well spread and they are near to '2's.)

→ The thing I can visualize is all my 'zeros' are together and so other numbers.

→ It's not only separating your one's but also separating slant ones' in one region. (whatever visually look similar are getting group, based on visual similarity.)

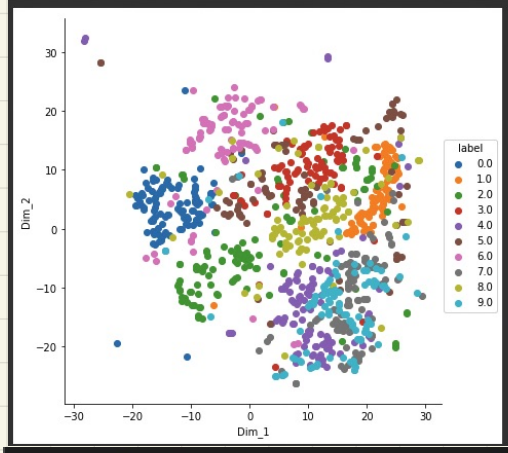
Code example of t-SNE

```
from sklearn.manifold import TSNE

# Picking the top 1000 points as TSNE takes a lot of time for 15K points
data_1000 = standardized_data[0:1000,:]
labels_1000 = labels[0:1000]

model = TSNE(n_components=2, random_state=0)
# configuring the parameteres
# the number of components = 2
# default perplexity = 30
# default learning rate = 200
# default Maximum number of iterations for the optimization = 1000

tsne_data = model.fit_transform(data_1000)


# creating a new data frame which help us in ploting the result data
tsne_data = np.vstack((tsne_data.T, labels_1000)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "label"))

# Ploting the result of tsne
sn.FacetGrid(tsne_df, hue="label", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_legend()
plt.show()
```

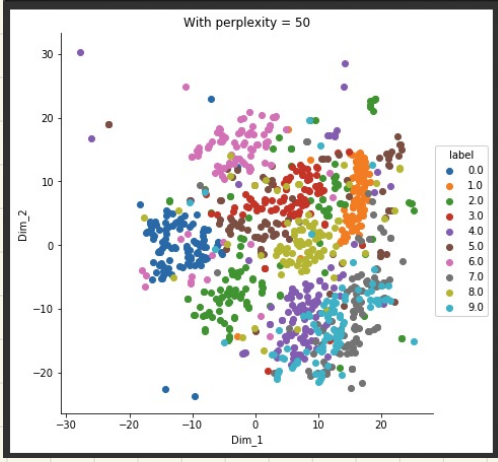
2-dim.

if you don't define this next time you ran TSNE. You will get slightly different result.

TSNE :- randomized algorithm (give slightly diff. result every-time you run).

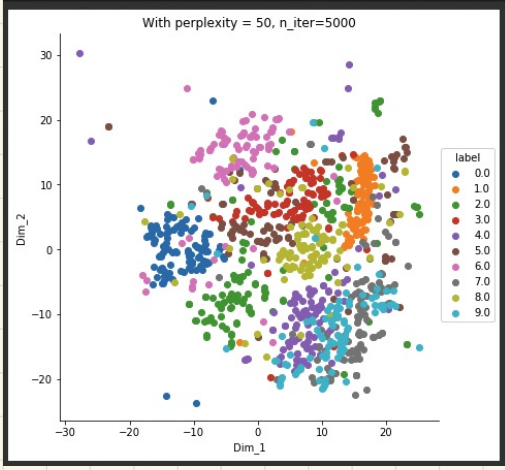



```
model = TSNE(n_components=2, random_state=0, perplexity=50)
tsne_data = model.fit_transform(data_1000)

# creating a new data fram which help us in ploting the result data
tsne_data = np.vstack((tsne_data.T, labels_1000)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "label"))

# Ploting the result of tsne
sn.FacetGrid(tsne_df, hue="label", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_legend()
plt.title('With perplexity = 50')
plt.show()
```



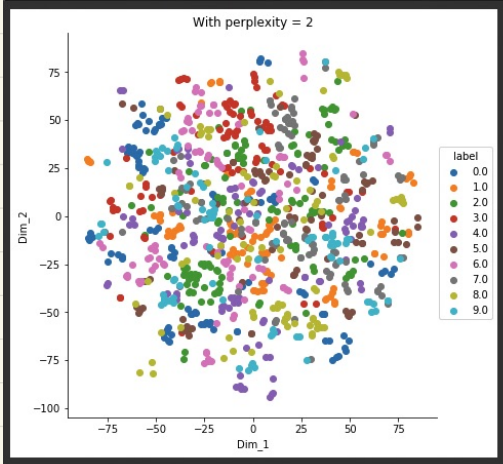
both are doing almost the same work. which means perplexity at 30 or 50 seems to make sense at 1000 iterations.

```
model = TSNE(n_components=2, random_state=0, perplexity=50,  n_iter=5000)
tsne_data = model.fit_transform(data_1000)

# creating a new data fram which help us in ploting the result data
tsne_data = np.vstack((tsne_data.T, labels_1000)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "label"))

# Ploting the result of tsne
sn.FacetGrid(tsne_df, hue="label", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_legend()
plt.title('With perplexity = 50, n_iter=5000')
plt.show()
```



→ here also the shape is very stable with previous result. So, 1000 iteration also good.

```
model = TSNE(n_components=2, random_state=0, perplexity=2)
tsne_data = model.fit_transform(data_1000)

# creating a new data fram which help us in ploting the result data
tsne_data = np.vstack((tsne_data.T, labels_1000)).T
tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "label"))

# Ploting the result of tsne
sn.FacetGrid(tsne_df, hue="label", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').add_legend()
plt.title('With perplexity = 2')
plt.show()
```



→ at low perplexity we can see the total mess.

```
#Excercise: Run the same analysis using 42K points with various
#values of perplexity and iterations.

# If you use all of the points, you can expect plots like this blog below:
# http://colah.github.io/posts/2014-10-Visualizing-MNIST/
```