Submission - Exercise [4]

Visual Data Analysis

[Maryam Assaedi, maryam.assaedi@rwth-aachen.de]
[Mst. Mahfuja Akter , s6msakte@uni-bonn.de]
[Mahpara Hyder Chowdhury, s6machow@uni-bonn.de]

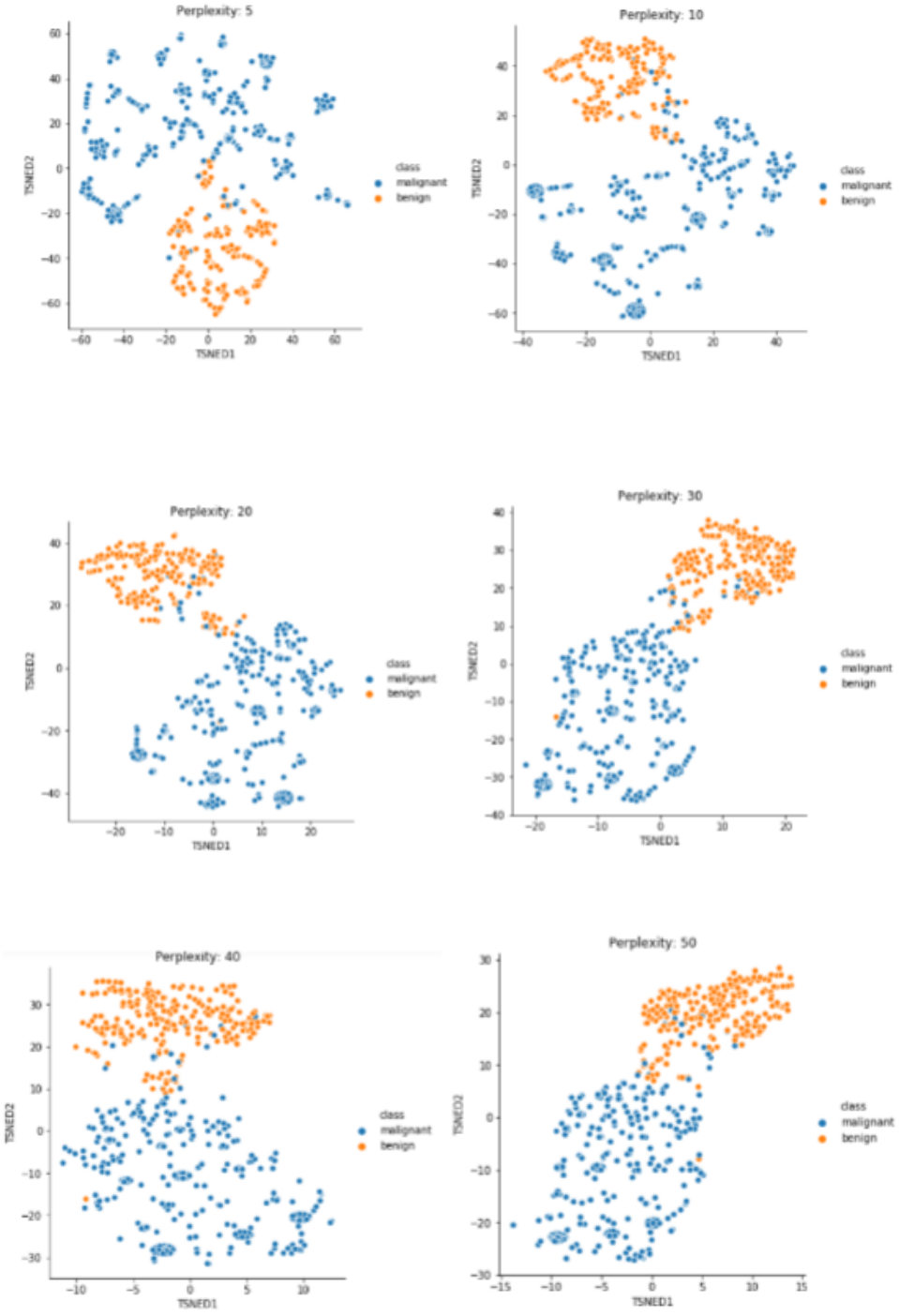
May 5, 2019

Exercise 1(Nonlinear Dimensionality Reduction)

In the previous assignment sheet, we used PCA for linear dimensionality reduction. Now, we will try out nonlinear techniques, compare them, and find out how their hyper parameters affect their results. You are free to use the implementations provided in the Python package scikit-learn.
a) Again read the breast-cancer-wisconsin.xlsx file from the previous sheet. Interpolate missing values as before, and keep all variables. We will now explore the results of t-SNE with different settings.
*) Run t-SNE with a random initial distribution of points and different perplexities, i.e., 5, 10, 20, 30, 40, and 50. Visualize the 2D data set in a scatter plot using different colors for cases from benign and malignant classes.
Answer:
We have applied t-SNE for random initialization(which is default settings) data with different perplexity.
Here is our implementation:

```
from sklearn.manifold import TSNE
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
a = np.random.randint(0, 698)
X = df.iloc[:, 1:10].values
#random_state=a,
tsne = TSNE(n_components=2,perplexity=5).fit_transform(X)

newTsneDf = pd.DataFrame(data = tsne, columns = ['TSNED1','TSNED2'])
df['class'].replace([2,4],['malignant','benign'], inplace = True)
tsnedf = pd.concat([newTsneDf, df[['class']]], axis = 1)

ax = sns.relplot(x="TSNED1", y = "TSNED2", hue="class", data = tsnedf)
plt.title("Perplexity: 5")
plt.show(ax)
```

Here is our plot for different perplexity:

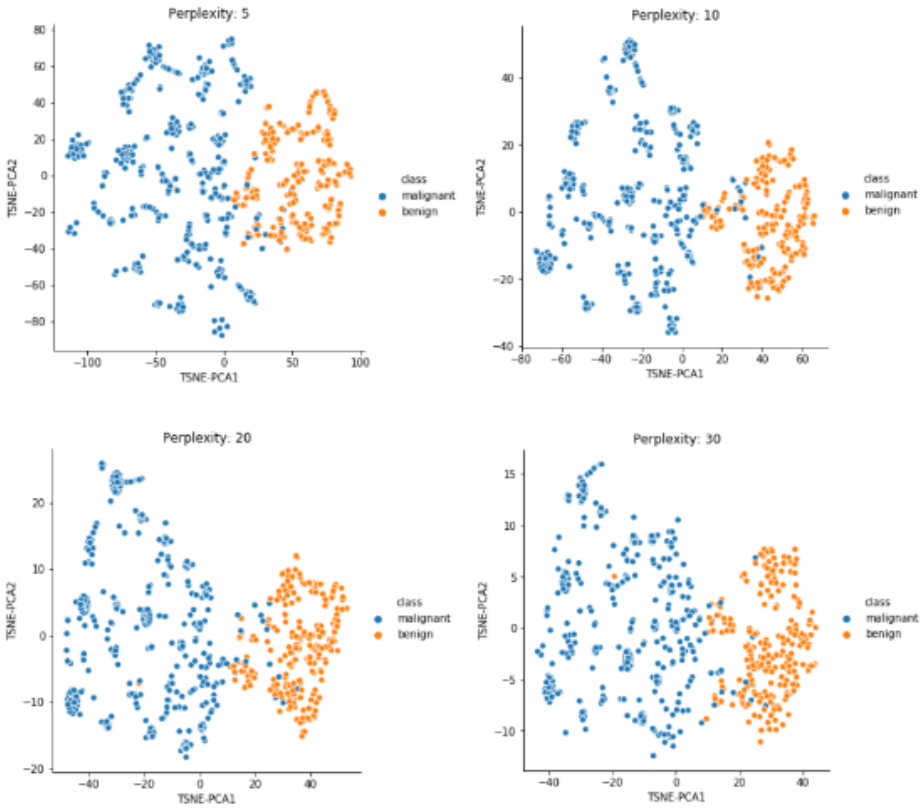


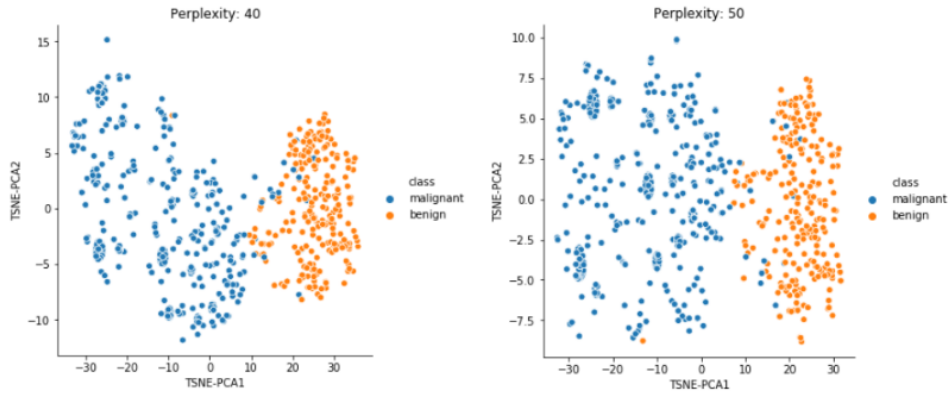
*) Repeat this experiment, except this time use PCA to create the initial distribution of points. Note that the implementation in scikit-learn allows you to select the initialization using a keyword parameter.

Answer:
We have applied t-SNE again using pca for initialization data with different perplexity.
Here is our implementation:

```
tsnepca = TSNE(n_components=2,init= 'pca',perplexity=5).fit_transform(X)

newTsnepcaDf = pd.DataFrame(data = tsnepca, columns = ['TSNE-PCA1','TSNE-PCA2'])
newTsnepcaDf = pd.concat([newTsnepcaDf, df[['class']]], axis = 1)

ax = sns.relplot(x="TSNE-PCA1", y = "TSNE-PCA2", hue="class", data = newTsnepcaDf)
plt.title("Perplexity: 5")
plt.show(ax)
```

Here is our plot for different perplexity:





*) Compare the diagrams from random initialization to the ones with PCA initialization. For which settings did the 2D embeddings fail to nicely separate the two data clusters? Does it depend on perplexity, initialization, or both? Why?
Answer:
When perplexity it too low(2-5) and too high(50¡) then T-SNE can not separate data nicely. because t-SNE works with neighbor data which come from perplexity. When neighbor data is too low it couldn't work better with low neighbor data and when neighbor data is unnecessarily high, also it can not give better result.
And distribution depends on initialization also.PCA initialization cannot be used with precomputed distances and is more stable than random initialization.

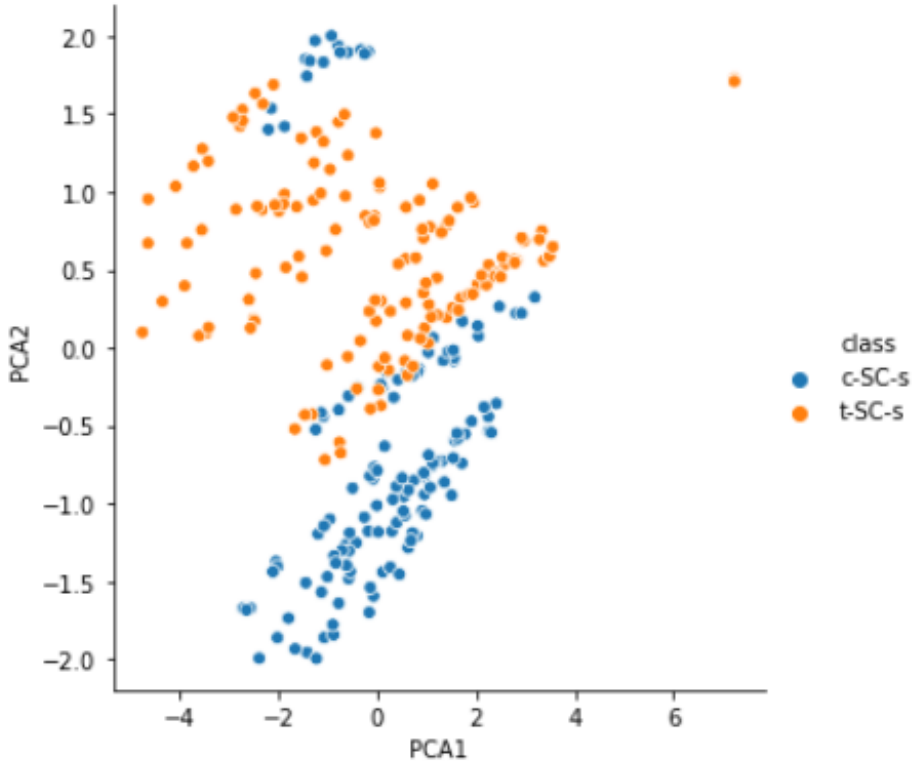
b) We will now compare PCA to ISOMAP on a Mice Protein Expression Dataset, available as Data_Cortex_Nuclear.xls on our lecture webpage. It contains expression levels of 77 proteins, measured in the cerebral cortex of 8 classes of mice. The classes result from two genotypes (Ts65Dn, which serves as a mouse model of human down syndrome, vs. normal controls), two treatments (injection of the drug memantine vs. a saline solution as a control), and two experimental conditions related to context fear conditioning (context-shock, which should lead to learning, vs. shock-context, in which no learning takes place). Counting all repeated measurements, there are 1080 samples overall, some with missing data. You can find more information on the data in the corresponding publication.
*) Read the data set and interpolate missing values in a reasonable way. Extract subgroups c-SC-s and t-SC-s. How many mice were measured for each of these groups?
Answer:
We imputed the data set and apply zero(0) in place of missing data entity. We have tried also with mean value but it's not giving better outcome. We found 135 mice in each of subgroups c-SC-s and t-SC-s
Here is our implementation.

```
import pandas as pd
df = pd.read_excel("Data_Cortex_Nuclear.xls")
#df.mean()
df2 = df.fillna(0)

#b1
col_names = list(df2) # get column names from existing dataframe
groups = df2.groupby('class') # creates a new object of groups of data
output_df = pd.DataFrame(columns = col_names)
count = 1
for index, group_df in groups:
    if(index == 'c-SC-s') or (index == 't-SC-s'):
        print(index,len(group_df))
```

```
c-SC-s 135
t-SC-s 135
```

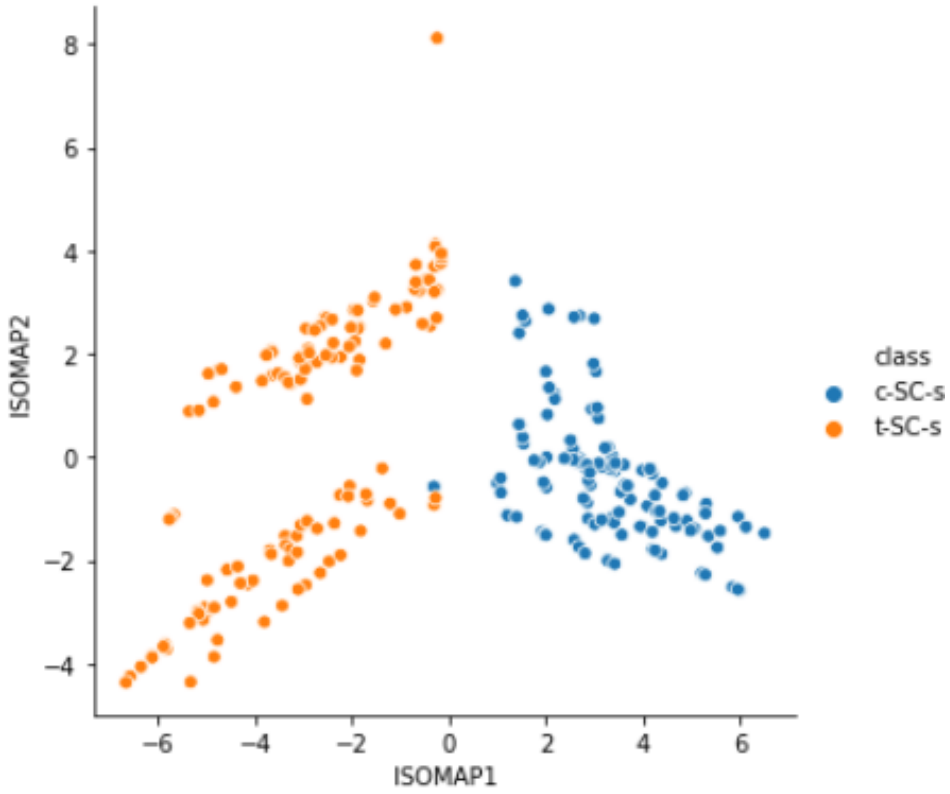
*) Only for the mice from c-SC-s and t-SC-s classes, use PCA to reduce the 77 dimensional data set to two dimensions. Make sure not to include the remaining columns, which contain meta information. Visualize the 2D data set in a scatter plot using different colors for instances from each class.
Answer:
Here is our implementation and plot for PCA

```
#b2
import numpy as np
import seaborn as sns
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

df3 = df2[df2.loc[:, 'class'].isin(df[df['class'] == 'c-SC-s']['class'])]
df3 = df3.append(df2[df2.loc[:, 'class'].isin(df[df['class'] == 't-SC-s']['class'])])
finaldf = df3.drop(['MouseID','Genotype','Treatment','Behavior'], axis=1)

X = finaldf.iloc[:, 0:77].values
pca  = PCA(n_components = 2).fit_transform(X)
principalDf = pd.DataFrame(pca)

principalDf.reset_index(drop=True, inplace=True)
finaldf.reset_index(drop=True, inplace=True)

principalDf = pd.concat([principalDf, finaldf[['class']]], axis = 1)
principalDf.columns=['PCA1','PCA2','class']

ax = sns.relplot(x="PCA1", y = "PCA2", hue="class", data = principalDf)
plt.show(ax)
```



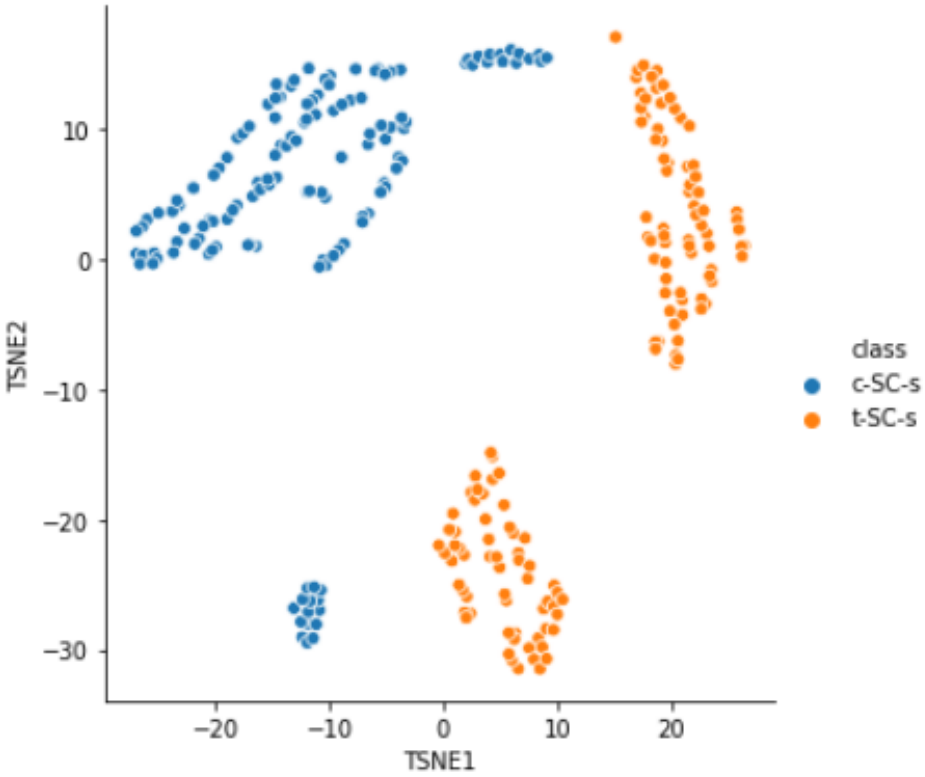
*) Produce a corresponding plot with ISOMAP. Try different values for the number of neighbors when constructing the neighborhood graph. Can you find a setting that separates the groups more clearly than PCA?
Answer: Yes, we have found better group separated data in ISOMAP. Here is our implementation and plot:

```
#b3
from sklearn.manifold import Isomap

iso  = Isomap(n_neighbors=10, n_components=2).fit_transform(X)
newIsoDf = pd.DataFrame(data = iso, columns = ['ISOMAP1','ISOMAP2'])

newIsoDf = pd.concat([newIsoDf, finaldf[['class']]], axis = 1)
bx = sns.relplot(x="ISOMAP1", y = "ISOMAP2", hue="class", data = newIsoDf)
plt.show(bx)
```



*) Finally, try t-SNE with different settings. Does this allow you to separate the groups better than with PCA or ISOMAP?
Answer: Yes, we have found better group separated data in t-SNE than PCA and ISOMAP.
Here is our implementation and plot:

```
#b4
from sklearn.manifold import TSNE
tsne = TSNE(n_components=2,perplexity=20).fit_transform(X)

newTsneDf = pd.DataFrame(data = tsne, columns = ['TSNE1','TSNE2'])
tsnedf = pd.concat([newTsneDf, finaldf[['class']]], axis = 1)

ax = sns.relplot(x="TSNE1", y = "TSNE2", hue="class", data = tsnedf)
plt.show(ax)
```

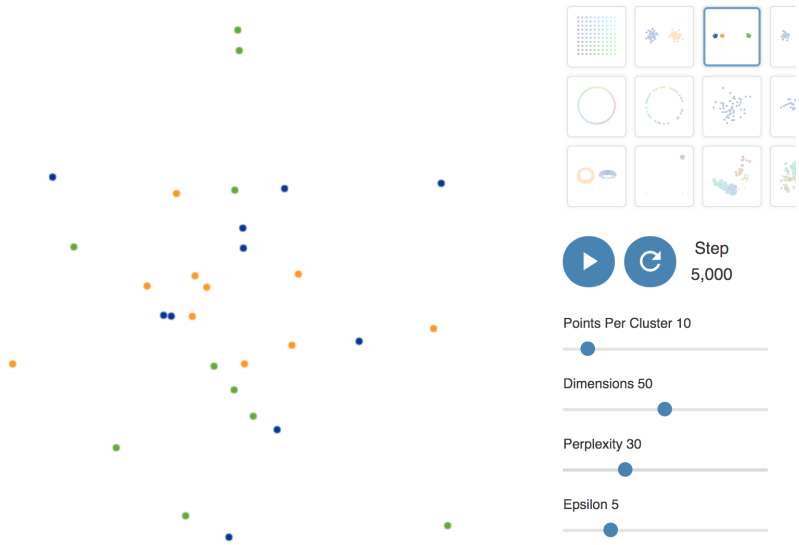


Exercise 2(Pitfalls in t-SNE)

a) Pick the "three clusters with equal numbers of points" data set. Set the number of points per class to 10, and number of dimensions to 50. Once run the demo with perplexity=29, and once with perplexity=30. Explain why there is a big difference in the final 2D embedding?

For perplexity = 29 we get this result

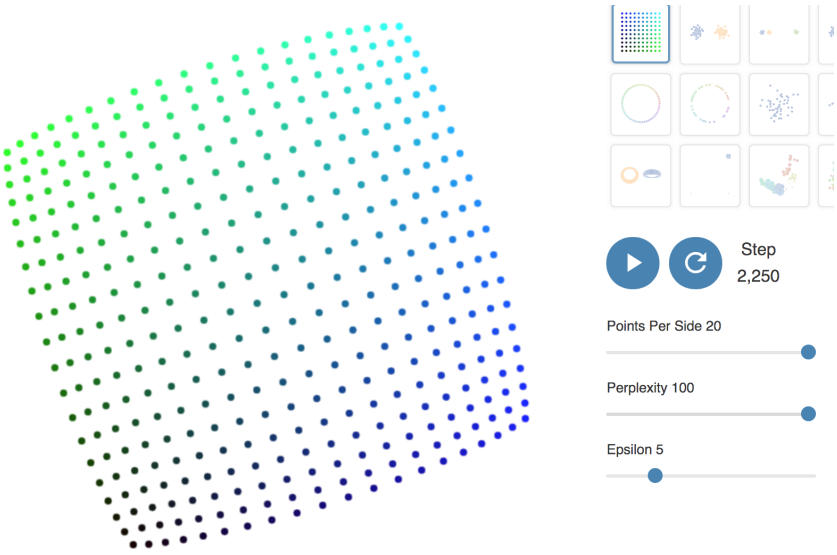
which gives a nice simulation of the 3 clusters that we have.
For perplexity = 30 we get this result



we see here no clusters and all the points are randomly scattered. This is due to the fact that the number of points for each cluster is 10 and since we have 3 clusters, then the total number of points is 30. The perplexity that we have at this point is also 30. This causes a problem as all the points want to be at equidistant from each other. This results in the seemingly random scattering of the points that we see.

b) Try the example "square grid with equal spacing between points", with 20 points per side. In the resulting plot with perplexity=100, why are distances between points in the middle of the square larger

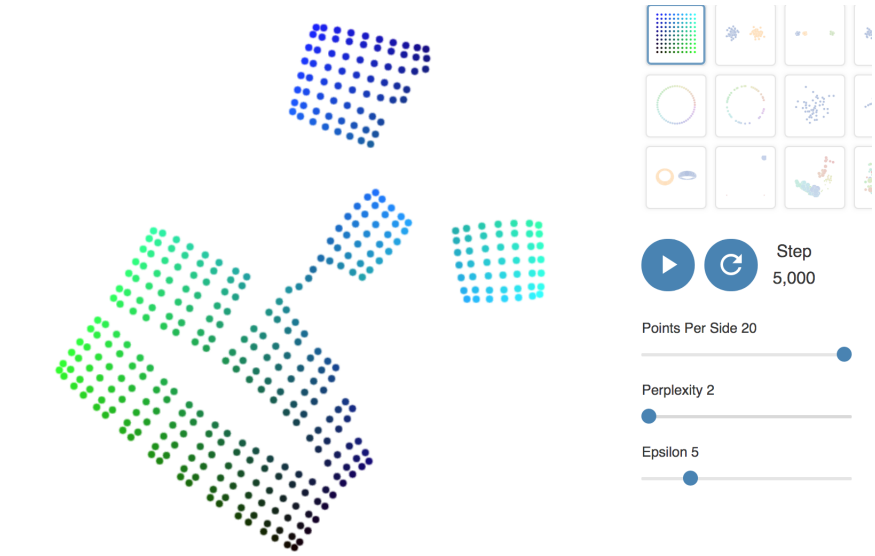
than near the boundary?
After simulating the problem described, we get the following results:



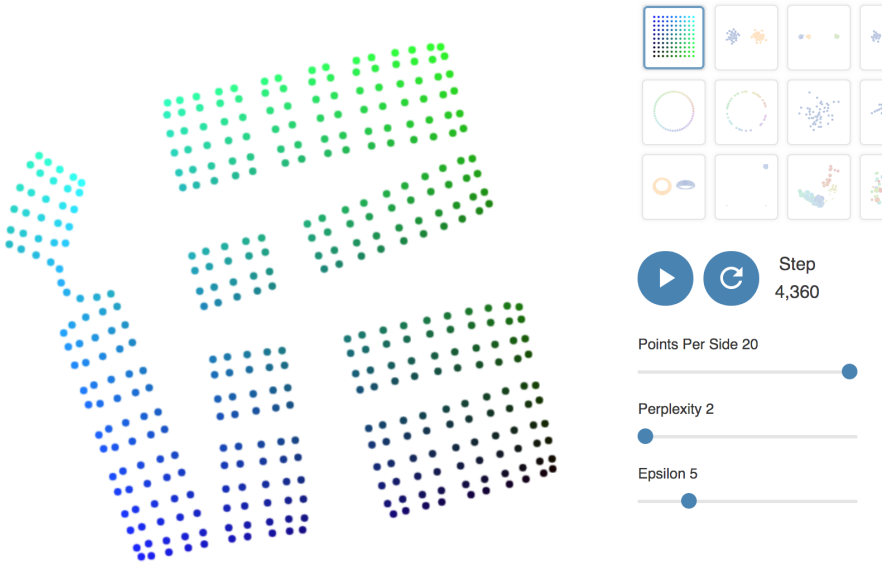
We see a clear difference in distances between the points in the middle of the square grid and the points near the boundary. We can explain this phenomenon by the fact that each point want to be closer to its 100th neighbor in a way. If we look at the points at the boundary, we can clearly see which points they are the nearest to them. But as we go to the center, the points gets pulled to each side because they are the 100th neighbor of each of the corner points making it hard for them to group together with any of them. This causes the points on the border to come closer together which the points in the middle to appear to have greater distance between them.

c) Pick "square grid with equal spacing between points" data set, with 20 points per side, and perplexity=2. Run the t-SNE multiple times. You will observe that the square grid sometimes breaks down into separate smaller clusters. Why?

We ran this t-SNE multiple times, the first time we got this results



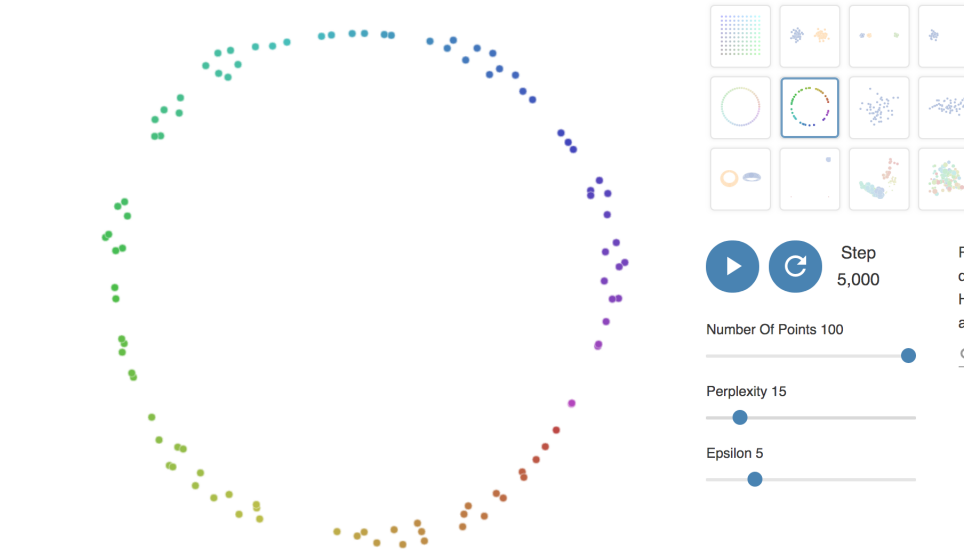
a second time running the t-SNE we get this



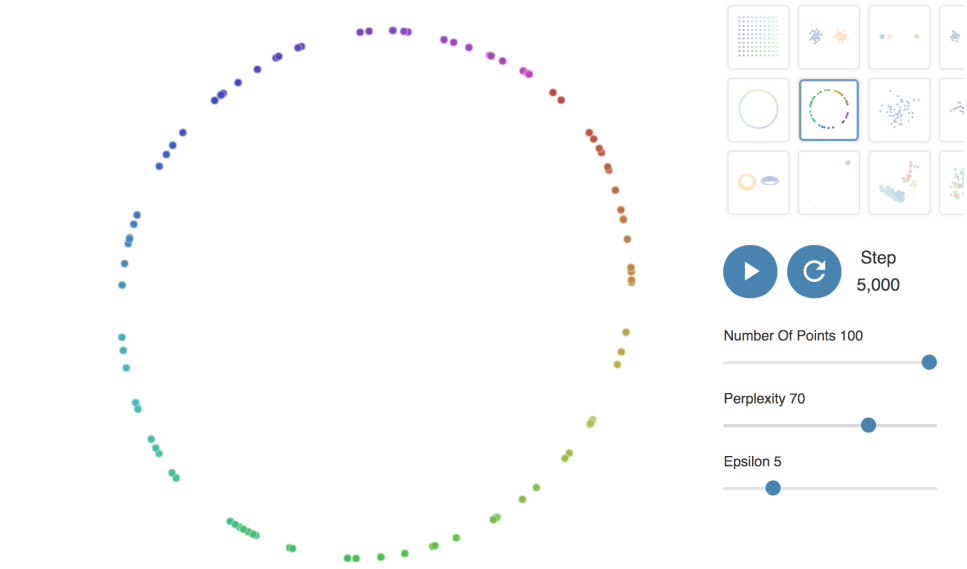
there is a huge difference in the two outcomes even though the parameter setting is the same. This is quite normal to happen with having a very small perplexity. At this low perplexity, the clusters have no meaning and can not give any meaningful effect and can not be interpreted.

d) Use different perplexities for "points randomly distributed in a circle" with 100 points. Around what perplexity value does the resulting visualization start to resemble the input data set? Explain why the perplexity has to be large enough for the result to look like

the input.
The visualization start to resemble the input data around perplexity = 15 as follows:



however we get better and better results as we go above perplexity = 50 as follows:



Since the points are colored with respect to their angle, we see that similar colored points are close to each other. Low perplexity only causes small clusters of same colored points and is not enough to produce a well defined circle. But

as we increase the perplexity, more points should be grouped together, so each point is grouped next to its 15th neighbor (in case of perplexity = 15) and this causes the circle to form as they are defined by the angle. In other words, a point is grouped to the neighboring 15 points and each adjacent point is grouped to it's 15 neighbours and their close relations is enough to start forming the circumference of the circle.

Exercise 3 (Approximated tSNE)

a) Briefly mention the steps involved in the tSNE algorithm. At which point does A-tSNE introduce an approximation?
The tSNE algorithm is used to convert high dimension data into 2 or 3 dimensions that can be visualized and used to infer relationships. It first interprets the distances between points in the high dimension space as a probability distribution. Then it outlines the similarities between points in low dimension. Finally it optimizes the locations of the points in the low dimension space by minimizing the cost function described by Kullback-Leibler.
A-tSNE introduces an approximation in the first step of the tSNE algorithm. So instead of calculated exact distances between points with tSNE does, it approximates the distances using Approximated K-Nearest Neighborhood (KNN).
b) The paper frequently refers to the concept of Progressive Visual Analytics. What is meant by this? In which sense could the original tSNE algorithm be used for Progressive Visual Analytics? How has this been extended by A-tSNE?
Progressive Visual Analytics allows the analyst to directly interact with the analytics process whilst showing the intermediate results of the transformations, thus allowing the analyst to tune the parameters needed to optimize the results required.
tSNE can be used for Progressive Visual Analytics as it can show the transformation of the data to 2 or 3 dimensions which is then used for interpretations. However, the tSNE algorithm is slow to compute and requires time to show the intermediate results making it unsuitable for Progressive Visual Analytics. In this case, A-tSNE is more suitable to the task as it only approximates the distances at the initialization step of tSNE. It also provides the possibility of refining the level of approximation that can be changed and adjusted by the user.
c) In the first case study (Allen Mouse Brain atlas), the authors propose to pre-process the data with PCA before running A-tSNE on it. Briefly explain why.
The data set that the authors used had some empty data. This missing data caused the creation of small clusters. And so they thought to first do the dimension reduction using PCA and take the first 10 components and use them instead.

Exercise 4 (Graph Visualization)

a) Use the Breast Cancer Dataset breast-cancer-wisconsin.xlsx and fill in the missing values, as in Exercise 1 a). Then compute the Pear-

son correlation between any pair of variables, and store them in a matrix.
We have filled the missing values with 0 and computed the Pearson correlation for every pair of variables and stored them in a matrix. Below is the code.

```
#a
import pandas as pd
df = pd.read_excel("breast-cancer-wisconsin.xlsx")

data = pd.isna(df)
df[data] = 0
#print(df)
```

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
#import forcelayout as fl

correlations = df.corr()
correlations.style.background_gradient(cmap='coolwarm').set_precision(2)
```

b) Create a graph from the correlation matrix and visualize it with a force-directed layout. Represent each variable as a node in the graph. Insert an edge between two variables whenever the Pearson correlation between them exceeds the threshold $\rho > 0.6$

Here, we have plotted a graph from the matrix and added edges to the nodes where the Pearson correlation between them exceeds the threshold $\rho > 0.6$.

The code and the final graph is being represented at the end of the task.

c) Modify the visual attributes of edges to reflect the magnitude of the correlation.

Here, we have modified the attributes of the edges to reflect the magnitude of the correlation. The thickness of the edges vary according the the correlation.

The code and the final graph is being represented at the end of the task.

d) Produce an alternative visualization with a circular layout. Color the nodes so that there are four set of nodes, one color for having at least one correlation more than 0.9 **to other nodes, another for having at least a correlation** $0.8 < \rho_{max} <= 0.9$**, one for having a correlation** $0.6 < \rho_{max} <= 0.8$ **and the last for the remaining nodes.**

Here, we have colored the nodes according to the given requirements

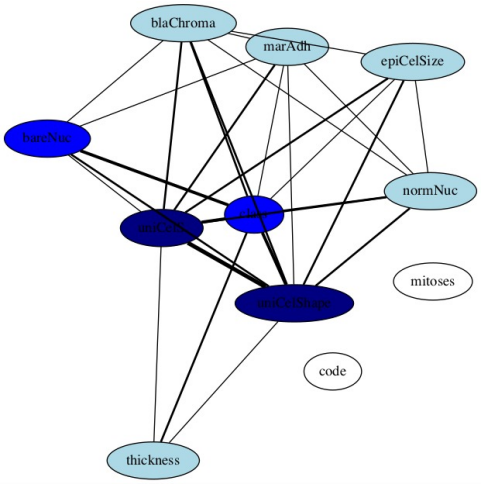
$$0.9 > \quad navy$$

$$0.8 < \rho_{max} <= 0.9 \quad blue$$

$$0.6 < \rho_{max} <= 0.8 \quad lightblue$$

$$0.6 =< \quad white$$

Below is the code and the corresponding graph



e) Answer the following questions: At the selected threshold, which nodes are disconnected from the rest of the graph and what do they indicate? If two nodes A and B are strongly correlated, and node C is strongly correlated with node B, can we conclude that node C will be also strongly correlated with node A? Based on the visualization, which variables would you propose to predict the class?

Node 'code' and 'mitoses' are disconnected from the graph. They are indicating that they have least correlation to the rest of the data.

If two nodes A and B are strongly correlated, and node C is strongly correlated with node B, then we cannot conclude that node C will be also strongly correlated with node A. For example, in the graph, 'bareNuc' and 'class' has a strong relationship, 'class' and 'uniCelS' has a strong relationship as well, but that does not mean that 'bareNuc' and 'uniCelS' has a strong relationship.

Based on the visualization, we would predict the 'class' variable should be class because this entity have strong correlation with many of other attributes.