# Written Report – 6.419x Module 2

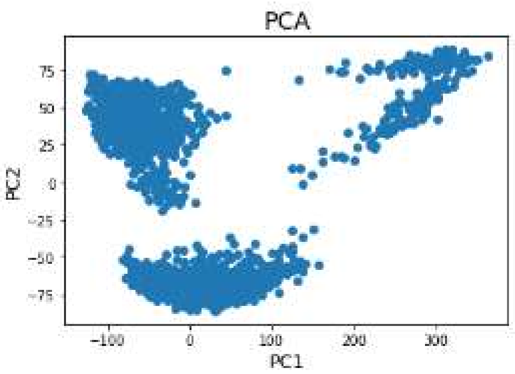
**Name:** (roeljpatricio)

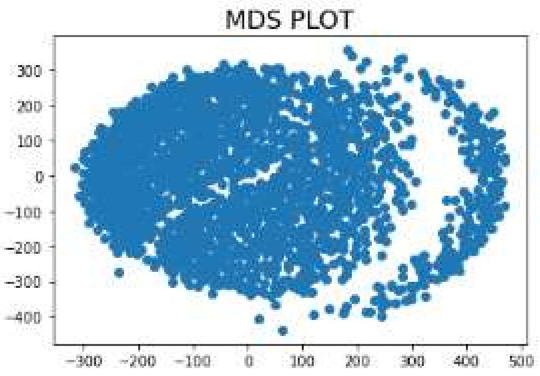
- **Part 1: Visualization**
  ***For easier reference, only questions were kept.***

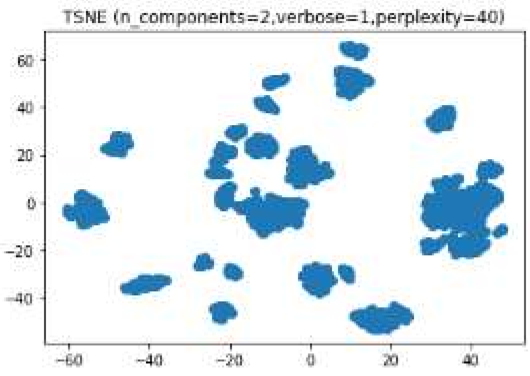
*1. (**3 points**) Provide at least one visualization which clearly shows the existence of the three main brain cell types described by the scientist, and explain how it shows this. Your visualization should support the idea that cells from a different group (for example, excitatory vs inhibitory) can differ greatly.*

**Solution:**

Let's start by plotting a few different visualizations



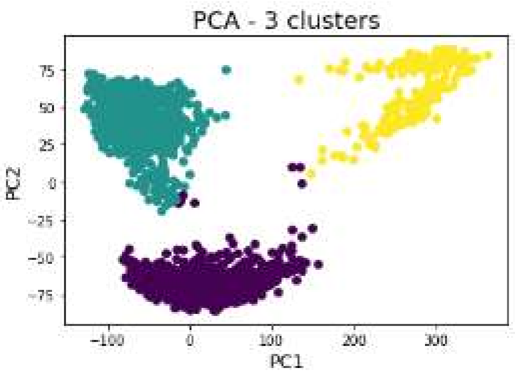


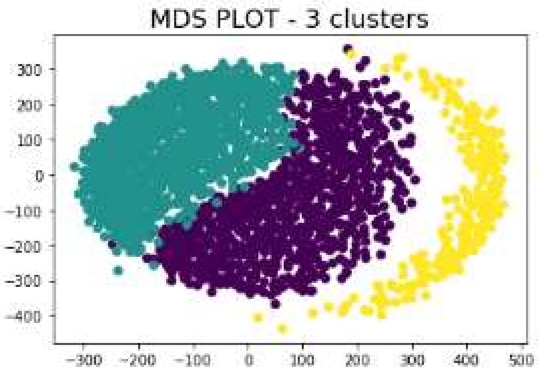


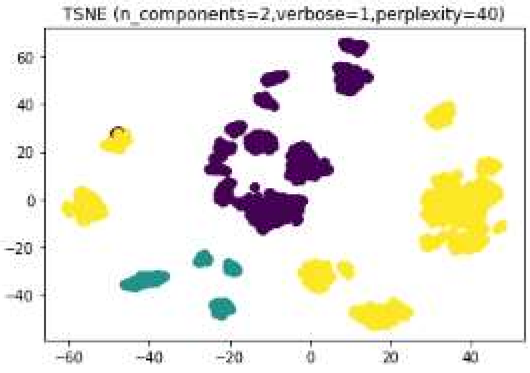
In the PCA plot of the first 2 PC we can clearly identify 3 main clusters, although one could hint that there could be more. The MDS plot also shows the presence of 3 possible clusters, though not so clearly (specially on the left side of the large circular shape, where there only is a tiny S shaped gap).

It is the t-SNE (50 first PC used) that shows the greatest discrepancy with this claim, although let's remember we are still eyeballing over the 2D visualization. Running KMeans will shed some light onto the visualizations.

**KMeans(n_clusters=3, n_init=100) over 50 PC**







With color coded clusters, we can see that there are 3 distinct clusters in all plots.

Let’s remember that all of these techniques are different and having all of them portraying 3 well defined different clusters is compelling.

The PCA tells us there are 3 different main types of cells because their 2 main components group each one of them in the same space vicinity. While we are only using 2 of the 2169 PC, this is already enough to show a pattern. KMeans ran on 50 PC confirms this claim by classifying the points according to those visual clusters.

As described before, there are 3 visual clusters in the MDS plot, which are also classified as such when KMeans is run.

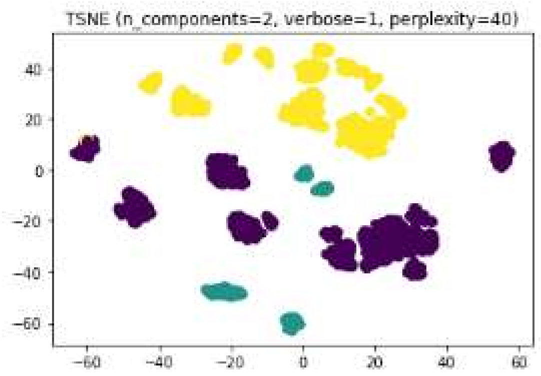
The t-SNE plot is the most interesting, as it already shows that there could be several other subgroups within each cluster, but specifically for this question observations are classified in 3 well defined groups.

*2. (**4 points**) Provide at least one visualization which supports the claim that within each of the three types, there are numerous possible sub-types for a cell. In your visualization, highlight which of the three main types these sub-types belong to. Again, explain how your visualization supports the claim.*

**Solution:**

As stated above, t-SNE shows several clusters. Using the same plot the 3 main groups are highlighted using the color labels obtained with KMeans.

We can see that the 3 main groups are maintained (meaning that the clustering is not “rainy”) and well defined. Although there green clusters are not next to each other, let’s remind ourselves that axis on the t-SNE plot don’t actually represent anything, and therefore their position is not a matter of concern[1]



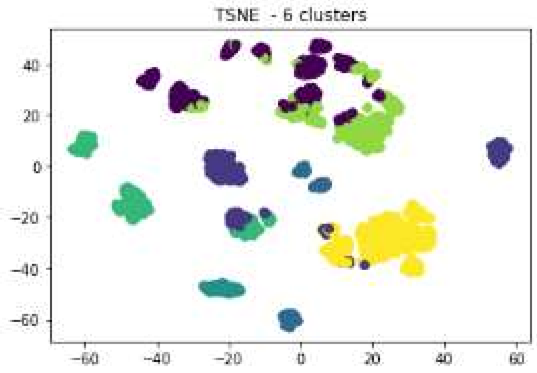
Recall that the objective of t-SNE is to maintain short distances between points short and long distances long (roughly speaking). Identifying subgroups of local structures is one of the strong points of t-SNE[2].

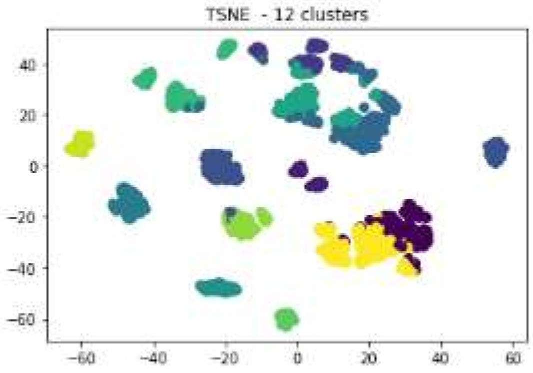
Running KMeans with several other cluster numbers confirms the existence of many more subgroups.

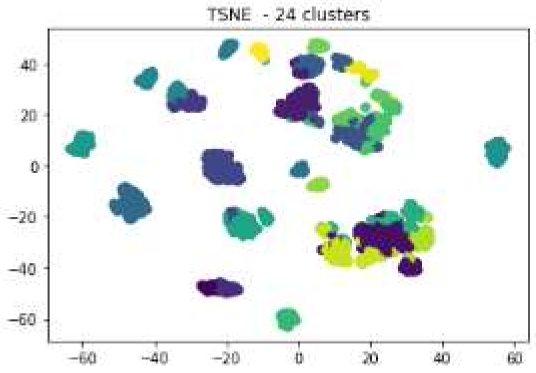
[1] It’s interesting to compare the different t-SNE plots and see how the clusters are the same but their positions change with different initializations (the plots were dn

[2] See sklearn docs https://scikit-learn.org/stable/modules/manifold.html#t-sne

All t-SNE were run with n_components=2 verbose=1 perplexity=40







Once again, note that clusters are well defined and consistent.

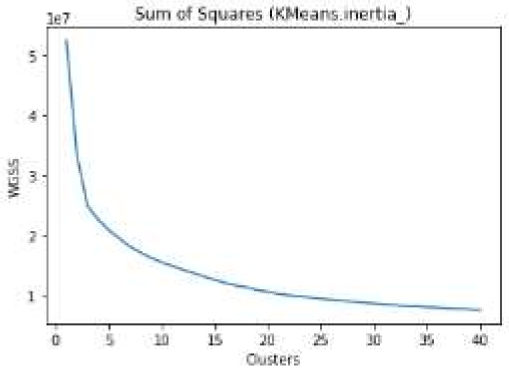
***Part 2: Unsupervised Feature Selection***

1. *(**4 points**) Using your clustering method(s) of choice, find a suitable clustering for the cells. Briefly explain how you chose the number of clusters by appropriate visualizations and/or numerical findings.*

**SolutionL**

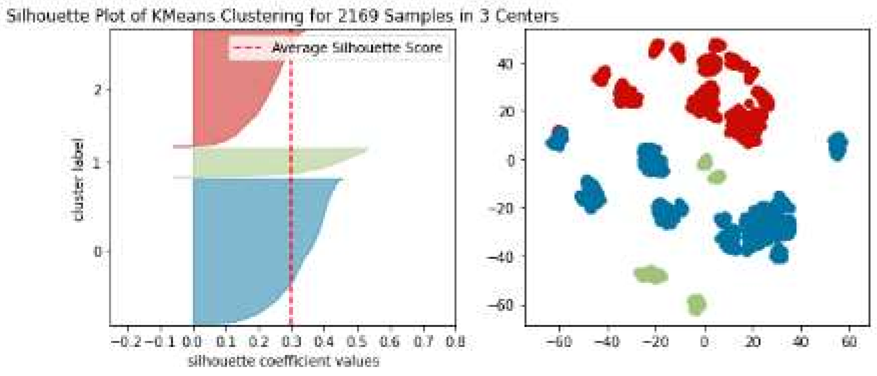
Looking at the t-SNE plots we can be quite convinced that the number of subgroups should be larger than 3.

Let's start by taking a look at the elbow plot



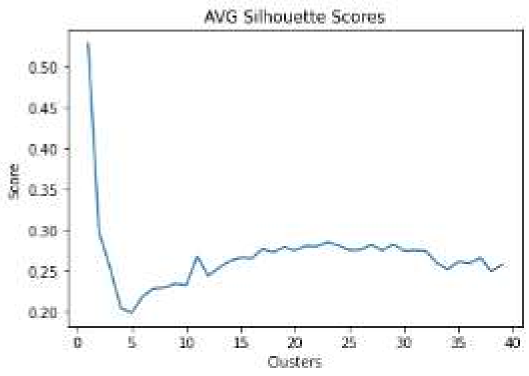
If we were to look closer (and I'm sure we all did) we would see that 3 is the number that has the break point. This is of course against our claim.

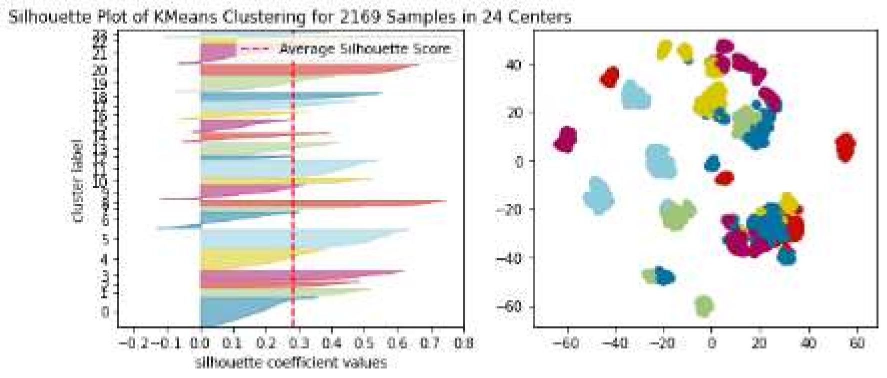
Now, let’s take a look at the silhouette score for 3 clusters



Which gives us an average sihouette score of 0.3 ( a pretty low score).

In order to compare the silhouette scores, we run KMeans from 2 to 40 clusters and plot their average scores. We will see on the AVG Silhouette Scores plot that the maximum revolts around 20-25, and extracting the largest value we find that 24 centers has the highest average score with a score of 0.28487.





2. *(**6 points**) We will now treat your cluster assignments as labels for supervised learning. Fit a logistic regression model to the original data (not principal components), with your clustering as the target labels. Since the data is high-dimensional, make sure to regularize your model using your choice of* $l1$*,* $l2$ *, or elastic net, and separate the data into training and validation or use cross-validation to select your model. Report your choice of regularization parameter and validation performance.*

**Solution:**

From now on the reduced data set will be used. KMeans was re run to get new target labels.

The regression model used was set up as below with 5 folds (default value)

LogisticRegressionCV(Cs=[0.01,0.1,1,10],penalty='l2',solver='liblinear',multi_class='ovr')

The best results were attained using l2 regularization, below is the array with the regularization parameters for 24 clusters. Taking advantage of the One Versus Rest approach, CV allows us to have different regularization values for different clusters.

[ 0.01, 0.01, 1. , 10. , 0.01, 0.1 , 1. , 0.01, 1. ,0.01, 1. , 0.01, 10. , 0.1 , 0.01, 1. , 0.01, 1. , 10. , 1. , 10. , 0.01, 10. , 1. ]

Under this set up the validation performance was 1.0.

3. *(**9 points**) Select the features with the top 100 corresponding coefficient values (since this is a multi-class model, you can rank the coefficients using the maximum absolute value over classes, or the sum of absolute values). Take the evaluation training data in `p2_evaluation` and use a subset of the genes consisting of the features you selected. Train a logistic regression classifier on this training data, and evaluate its performance on the evaluation test data. Report your score. (Don't forget to take the log transform before training an testing.)*
   *Compare the obtained score with two baselines: random features (take a random selection of 100 genes), and high-variance features (take the 100 genes with highest variance). Finally, compare the variances of the features you selected with the highest variance features by plotting a histogram of the variances of features selected by both methods.*

**Solution:**

The features were selected using the sum through the clusters of the absolute values of the features (20000 in the reduced case).

To select them a mask was created. 'all100' is an array containing all the features

mask = np.where(all100 > np.sort(allfeatures)[19899])

With the top 100 features, fitting a logistic regression model as below

LogisticRegressionCV(Cs=[0.01,0.1,1,10],penalty='l2',solver='liblinear',multi_class='ovr')

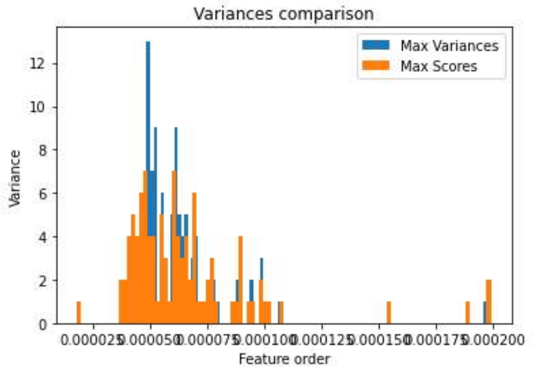
We obtain a **validation score of 0.937**. Saga returns a slightly larger value (0.94) but because of the computational cost we will stick with liblinear.

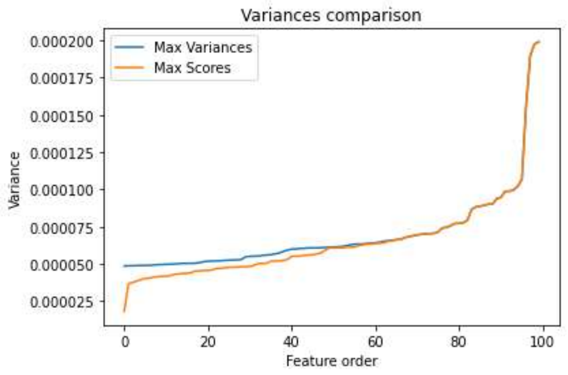
In contrast, when using the same model as above but with **randomized features**, the **validation score is only 0.554.**

On the other hand, the **validation score** we obtain when we use the **top 100 variance** features is **0.935.**

We can see that the validation score of the top 100 variance and the top 100 features is very similar.

By looking at the histogram we can see that the variances on both selection methods seem to follow the same shape. Doing a line plot (graph on the right)





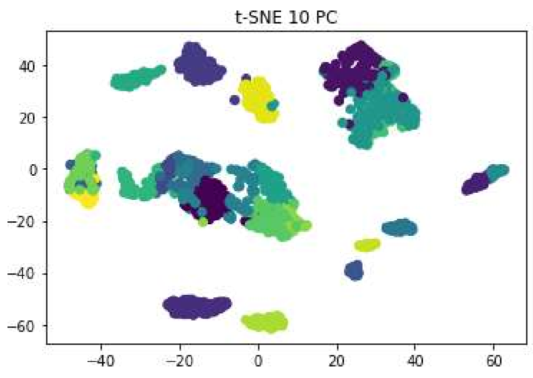
It seems natural that those features with the largest variance are the ones having the largest impact on the assigned labels, since they would explain why points differ from each other.

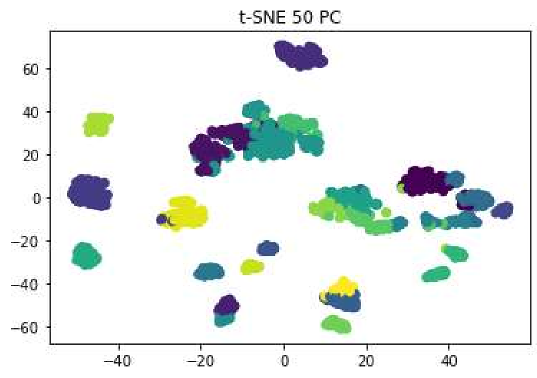
Also, our ground truth comes from our predicted labels. These labels were obtained using KMeans, which was used with on the projected PCA log transformed data points. It's logical that, under these labels, the features with the largest variance are the most important ones, since those labels are coming from the PCA transformed data (and PCA's objective is, indeed, to look for the directions with the largest variances).

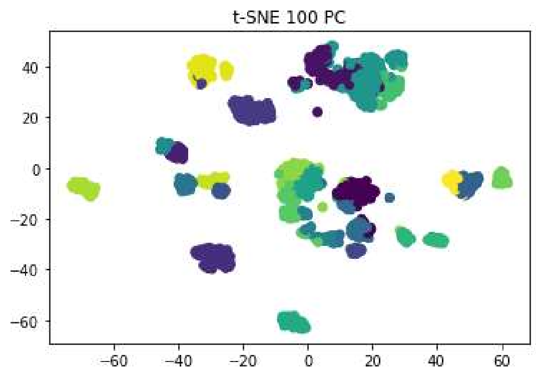
- **Problem 3: Influence of Hyper-parameter (Written Report)**
    1. ***(3 points)*** *When we created the T-SNE plot in Problem 1, we ran T-SNE on the top 50 PC's of the data. But we could have easily chosen a different number of PC's to represent the data. Run T-SNE using 10, 50, 100, 250, and 500 PC's, and plot the resulting visualization for each. What do you observe as you increase the number of PC's used?*

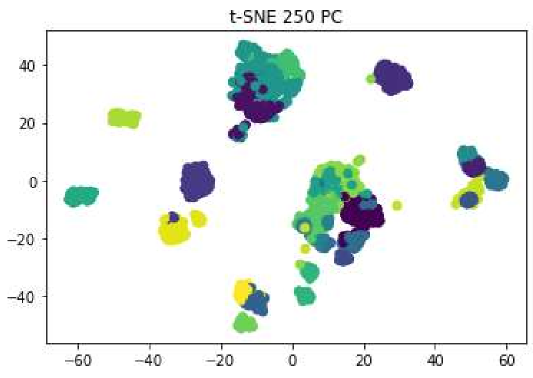
**Solution:**

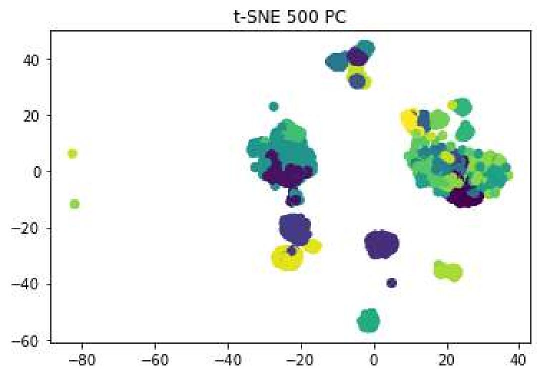
Let's start plotting with different PCA numbers. Let's recall we are still using KMeans(n_clusters=24,n_init=50) and TSNE(n_components=2,verbose=1,perplexity=40)

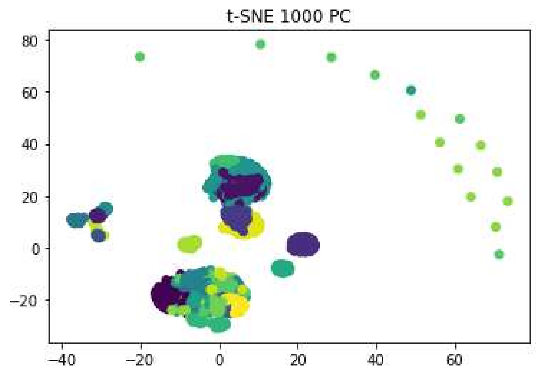












As a bonus track, and because it’s very graphical, t-SNE run on 1000PC was included. It’s easily observable how increasing the number of principal components used starts to “shrink” the clusters, in a similar fashion as it happens when wrong perplexity values are used.

Numerically, the KL divergence starts to increase as we add more and more components, meaning that we are getting worse distributions.

2. ***(13 points)** Pick three hyper-parameters below and analyze how changing the hyper-parameters affect the conclusions that can be drawn from the data. Please choose at least one hyper-parameter from each of the two categories (visualization and clustering/feature selection). At minimum, evaluate the hyper-parameters individually, but you may also evaluate how joint changes in the hyper-parameters affect the results. You may use any of the datasets we have given you in this project. For visualization hyper-parameters, you may find it productive to augment your analysis with experiments on synthetic data, though we request that you use real data in at least one demonstration.*

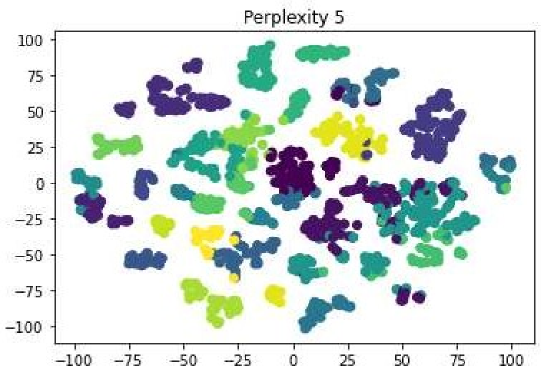
**Solution:**

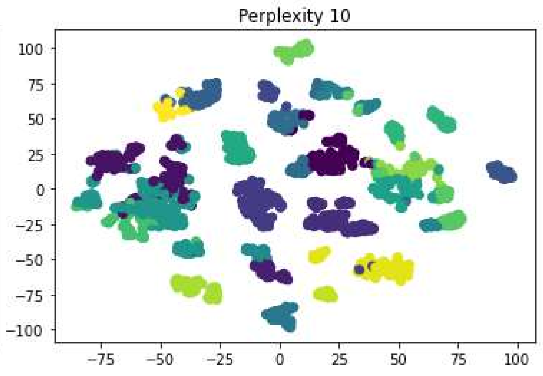
Category A: Perplexity

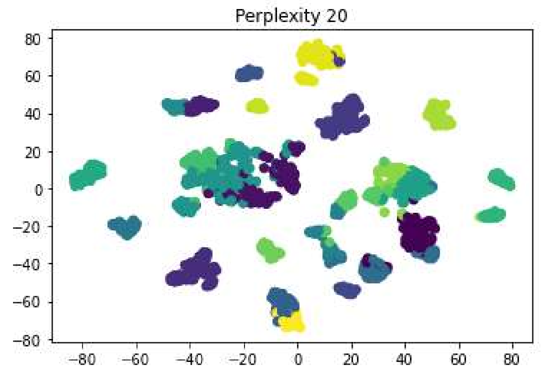
According to Sklearn documentation, perplexity values should be within 5 and 50. Below are 6 graphs between those numbers. It can be observed that the perplexity value spreads and contracts points in the plot. In this case, a value of 5 provides a loose clustering, while a value of 50 starts gathering clusters together.

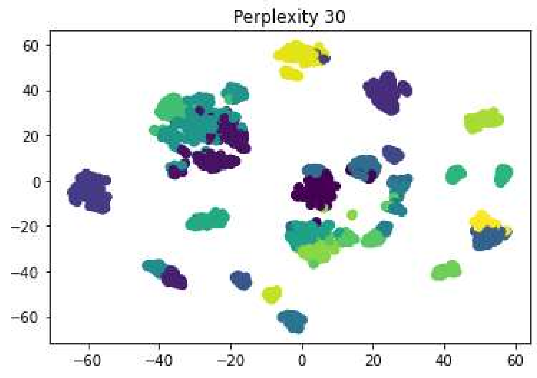
The perplexity measures how “attracted” points are to each other, and it is an estimate of the number of effective neighbors. Hence, smaller values favor global structure, producing “fluffy” projections, while larger values favor local structure generating dense clusters.

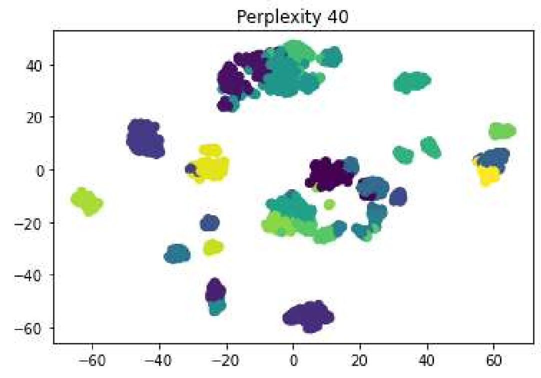
A balance, depending on the information we have at hand, is the key to attaining and adequate visualization.

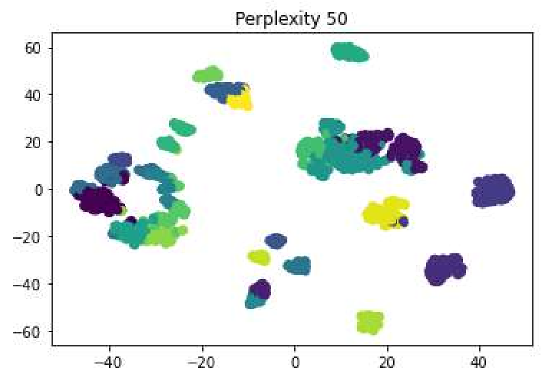








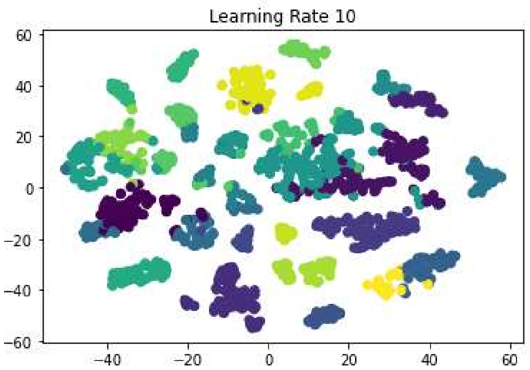


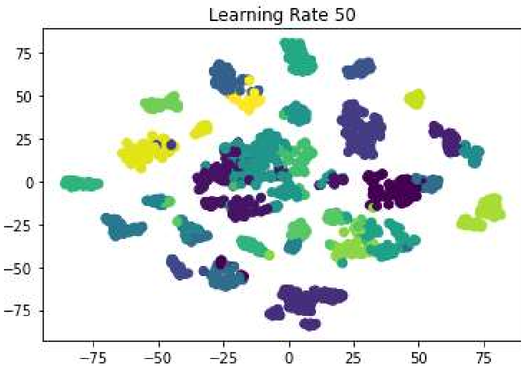


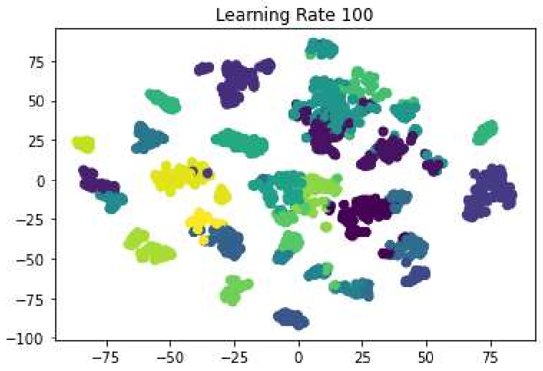
Category A: Learning Rate

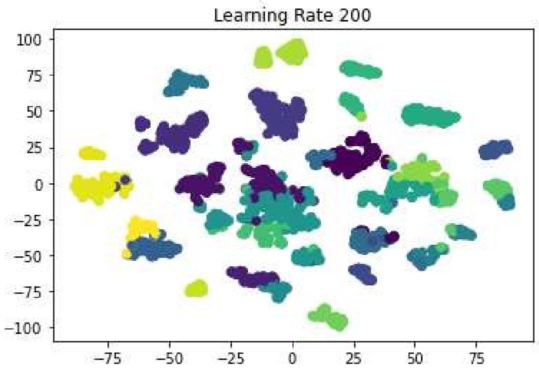
Set up as before KMeans(n_clusters=24,n_init=50) and TSNE(n_components=2,verbose=1,perplexity=30, learning_rate=**variable**)

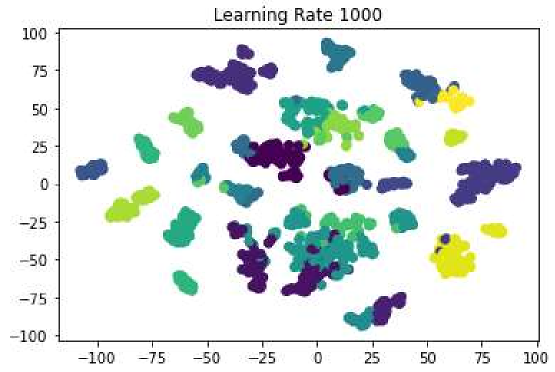
**[below plots were run with perplexity = 10]**



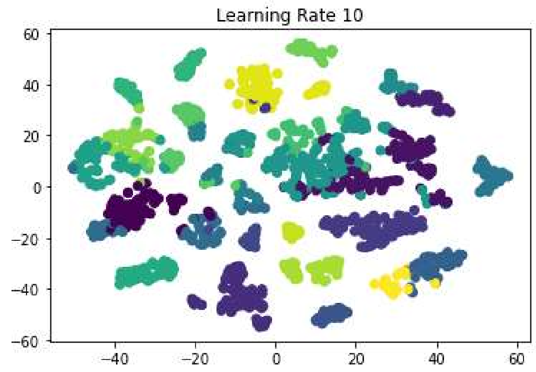


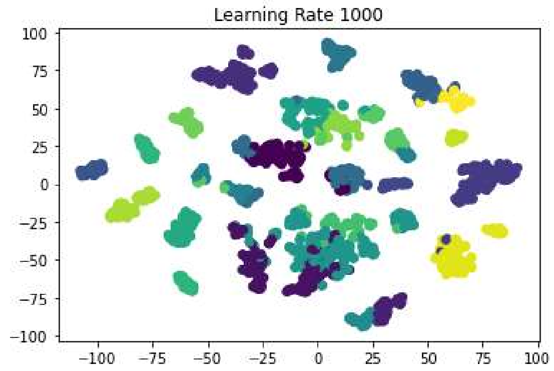




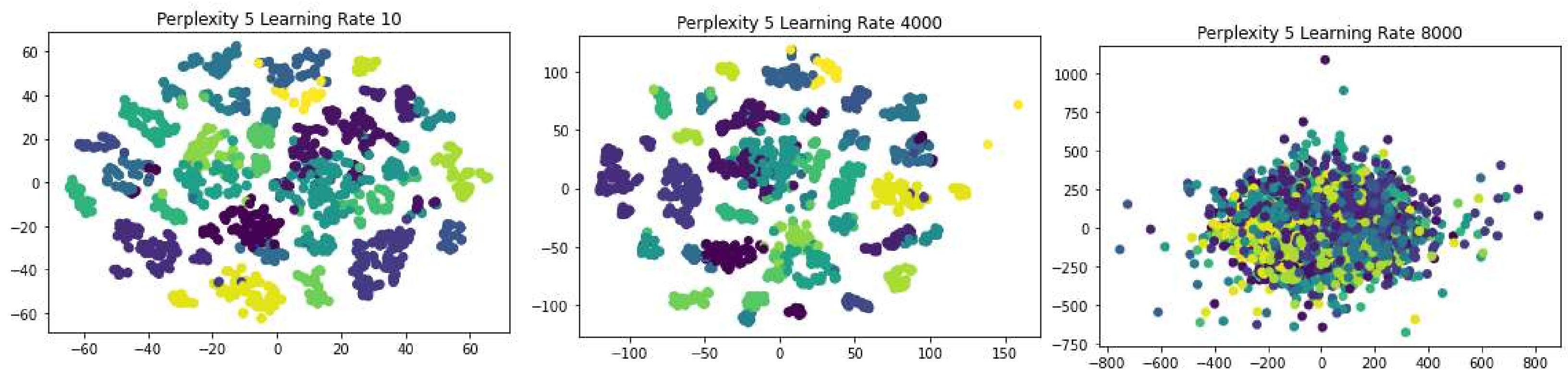


It’s actually very hard to see large differences among the plots. However, paying attention to the extreme cases it can be noticed that a lower learning rate has less dense clusters compared to a higher one.

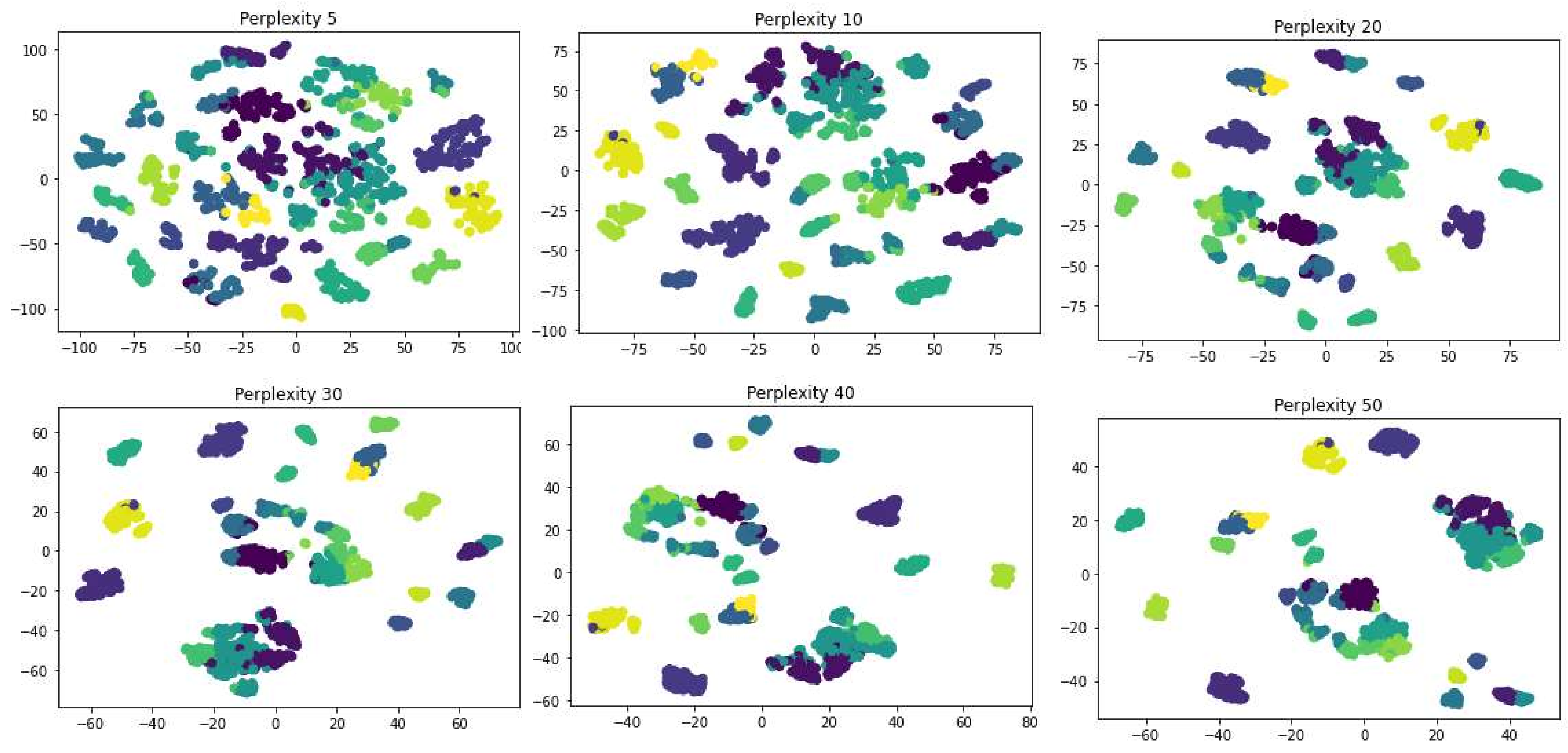




In order to get a better grip at the effect of the learning rate, let’s take extreme values of perplexity and learning rate



Below are plots with variable perplexities and a stationary **learning_rate=1000**



The learning rate is just as with any other algorithm, a way to control the gradient descent. With lower values we run the risk on finding a bad local minima and converge in the wrong spot. Larger values could “blow up” and increase the KL divergence. This can be observed in the figure with learning rate=8000. According to the documentation, typical values range from 10 to 1000.

Together with perplexity, these are the 2 most important values to tune.

I didn’t have time to complete category B. Thank you for reading this far.