LINM2472 - Project 2

Student : Thonnard Julien - Vermeulen Corentin

1 Part 1: Graph Kernel

1.1 Computing the kernel

1.1.1 Weisfeiler-Lehman sub-tree complexity

We iterate over each node: $O(N)$.
We iterate H time again: $O(N)$ * $O(H)$.
Finally we find the complexity: $O(NH)$

1.1.2 Compute the kernels

To compute the kernels matrix we used the following pseudo code.

Algorithm 1 Weisfeiler-Lehman kernel computation

Initialize $wl\_framework$ the framework to compute Weisfeiler-Lehman sub-tree kernel
Initialize wl_kernel_matrix
for each graph g_i **do**
  fit the $wl\_framework$ on g_i and transform it
  for each graph g_j **do**
    transform g_j through $wl\_framework$ (this correspond to $k(g_i, g_j)$ add $k(g_i, g_j)$ to the wl_kernel_matrix
  end
end
Result: matrix K_{ij} where $k_{ij} = k(g_i, g_j)$

Later we found a method way faster to compute the pairwise matrix:
Directly applying the `wlm_kernel.fit_transform()` function on the whole data set. It gives the sames results. We arbitrary choose hyper-parameter H equal to 10 as it used for other part of the project.

1.1.3 Explicit features

Kernels have an implicit embedding space. For the Weisfeiler-Lehman sub-tree kernel, such an embedding can be made explicit. As explained in the Weisfeiler-Lehman Subtree Kernel *documentation*, it can be shown that the explicit embedding is equivalent to comparing the number of shared sub-trees between the two input graphs.

1.1.4 Explicit embedding versus kernel

The rank of the WL sub-tree kernel matrix for the three data sets with H=10 iterations are :

- 175 for the MUTAG data set
- 595 for the ENZYMES data set
- 4002 for the NCI1 data set

1.2 Visualization

1.2.1 Kernel centralization

To verify empirically that our matrix is well centered according to the formula, we applied it to our kernel matrix. Since the result is 0.0, the matrix is well centered.
As explained in the course, even if the data **xi** are centered in input space it is not always the case for the $\phi(xi)$ in implicit space. But if we define a centered data

$$\tilde{\phi}(\mathbf{x}_i) := \phi(\mathbf{x}_i) - \frac{1}{N}\sum_{i=1}^{N}\phi(\mathbf{x}_i)$$

We can find again a $\tilde{\lambda}\tilde{\alpha} = \tilde{K}\tilde{\alpha}$ based on the new $\tilde{\phi}(\mathbf{x}_i)$.
We have the same results as with non centered notation but with the following notation changes: α becomes $\tilde{\alpha}$, K becomes $\tilde{K}$ and λ becomes $\tilde{\lambda}$.

1.2.2 kernel-PCA implementation

Algorithm 2 kernel PCA implementation

Take K (see Algorithm 1)
K = centering(K)
Get λ and V the eigenvalues and eigenvectors of K.
Order V according to descending λ's values.
Select the two first components of V.
Scale V with: $V = \sqrt{\lambda} * V$
Plot results

1.2.3 kernel-PCA visualization

The kernel PCA plot below is quite similar to the results given by the `scikit_learn.KernelPCA()` function with some sign variation (plots were flipped along some axis) due to the sign of the eigenvectors.
It is noted that there are multiple communities on the graphs but it is difficult to identify them since they are all superimposed.

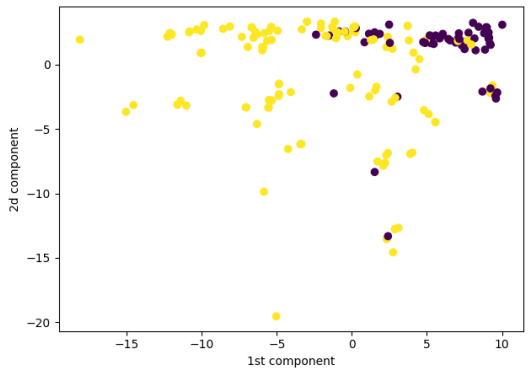

Figure 1: "MUTAG" kPCA

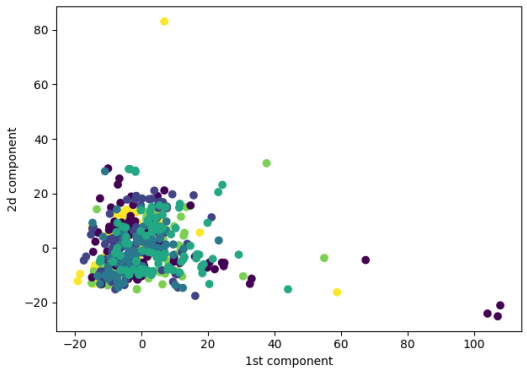

Figure 2: "ENYMES" kPCA

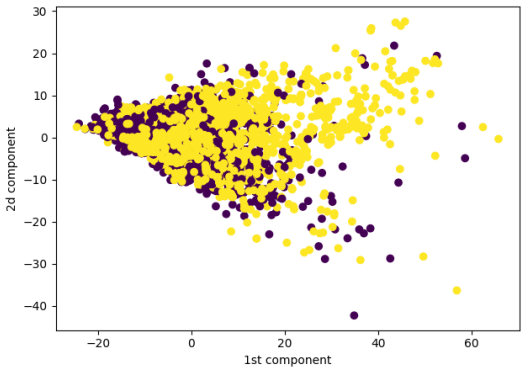

Figure 3: "NCI1" kPCA

It is noticed that with less values, the Kernel PCA is more easily readable than when the number of values is large. The different colors must be the different group but they are hard to see in these graphics. We don't have a very good segregation for the ENZYME and NCI1 data set.

1.2.4 Distance

Since the implicit embedding of $k(G_i, G_j)$ is $(\phi(G_i) \cdot \phi(G_j))$,
The pairwise distance:

$$\sqrt{k(G_1, G_1) + k(G_2, G_2) - 2k(G_1, G_2)}$$

Can be written as follows in the implicit space:

$$\sqrt{(\phi(G_1) \cdot \phi(G_1)) + (\phi(G_2) \cdot \phi(G_2)) - 2 * (\phi(G_1) \cdot \phi(G_2))} = \sqrt{||\phi(G_1) - \phi(G_2)||_2}$$

1.2.5 tSNE

The t-SNE stands for T-Distributed Stochastic Neighbor Embedding, which is a non linear data reduction algorithm that aims at visualizing high dimensional data in two dimension while keeping the original data structure.
In this algorithm the similarities between points is the joint probability and it tries to minimize the divergence between the joint probability between low and high dimensional data points. The main hyper-parameter are the perplexity and the steps where we can find a guide on *here*.
We used the default hyper-parameter because even after trying to tune them we did not get enough differentiated results.

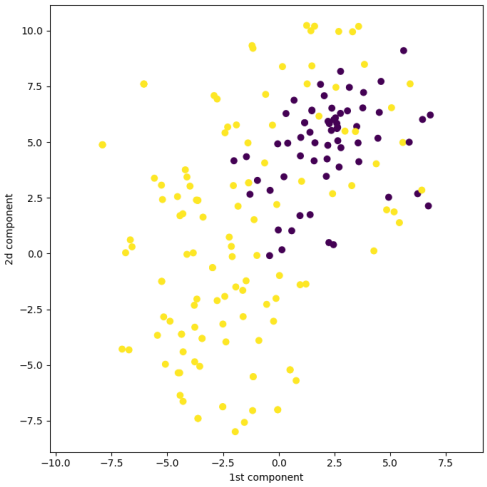

Figure 4: "MUTAG" t-SNE

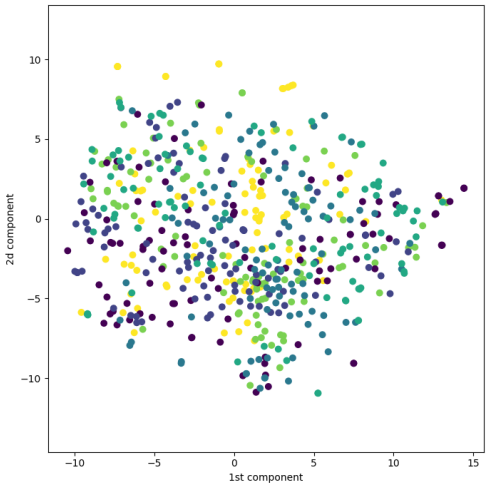

Figure 5: "ENYMES" t-SNE

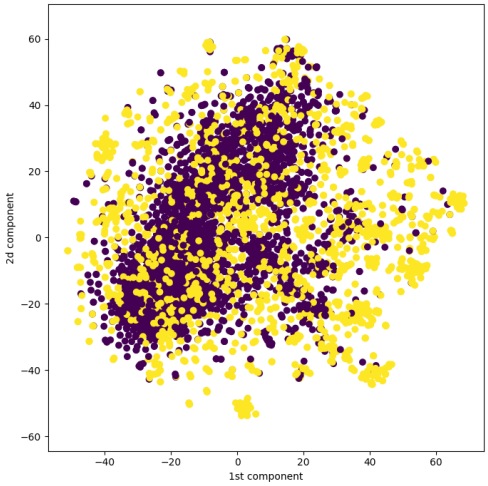

Figure 6: "NCI1" t-SNE

1.2.6 Comparison

The comparison between k-PCA and t-SNE for each data set can be found below.
It is observed that the t-SNE values are spatially more scattered, they don't overlap which allows a better readability on the graphs. Nevertheless, as the size of the data set increase, the readability of the t-SNE decrease and become as unreadable as the kPCA..
The mains differences between to two is that k-PCA is a linear dimensionality reduction technique in the implicit space that tries to preserve the global structure of the data and t-SNE is a non-linear dimensionality reduction technique that tries to preserve the local structure of the data. So that explains the variability between the two graphics for each database.

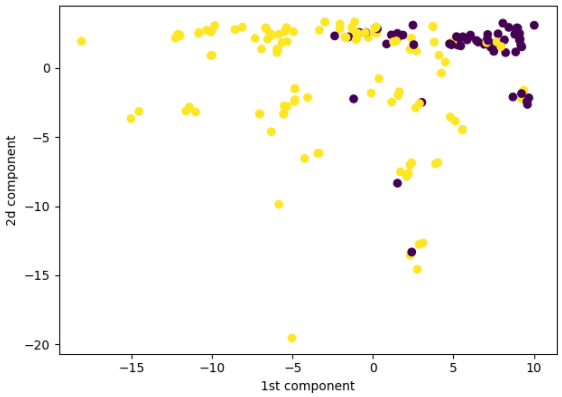

Figure 7: "MUTAG" kPCA

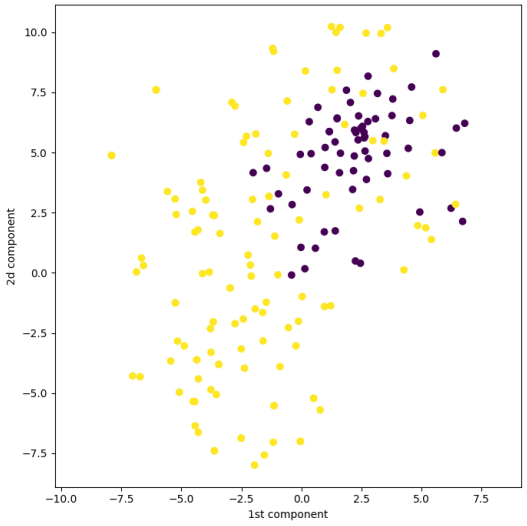

Figure 8: "MUTAG" t-SNE

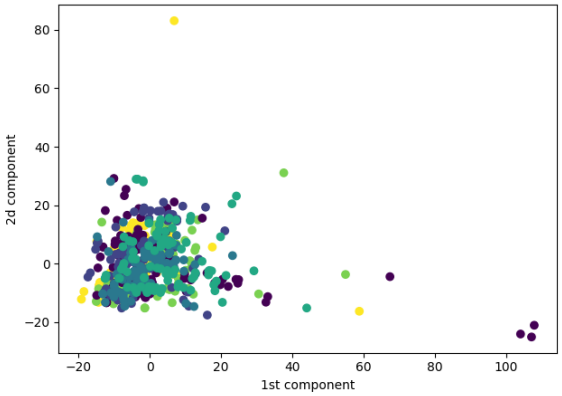

Figure 9: "ENZYMES" kPCA

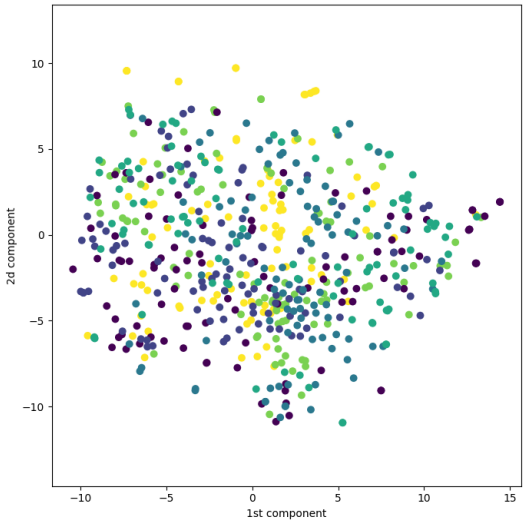

Figure 10: "ENZYMES" t-SNE

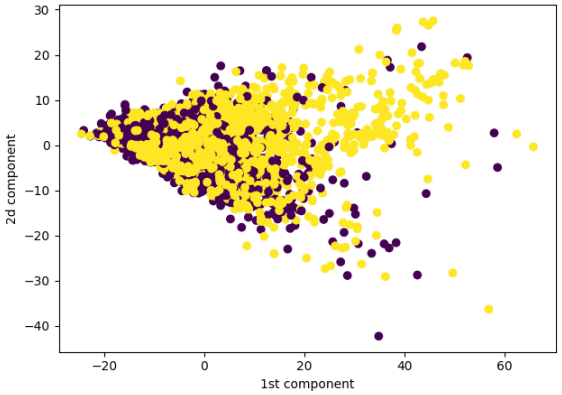

Figure 11: "NCI1" kPCA

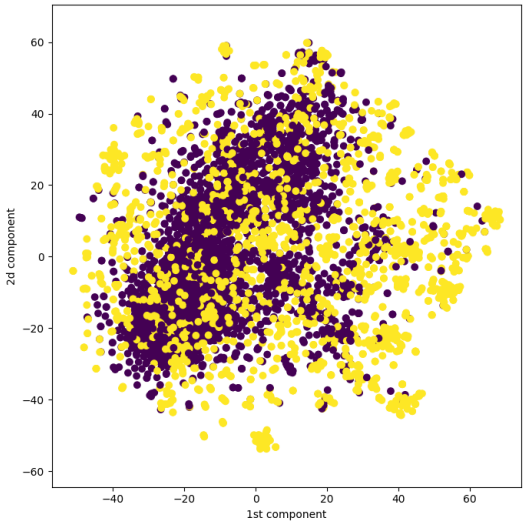

Figure 12: "NCI1" t-SNE

1.3 Classification

1.3.1 A simple baseline

To compute the simple baseline classification, we will always predict the mode class, meaning the class which is the most represented in the data set.

- For the MUTAG data set, we find 125 rows in class 1 and 63 in class -1 therefore we will always predict class 1. It will then lead to a $\frac{125}{188} = 0.665$ accuracy.
- For the ENZYME data set we have 6 classes with same number of rows (6 classes of 100 observation). It will then always predict the same class picked at random giving a $\frac{100}{600} = 0.167$ accuracy.
- For the NCI1 data set we have 2 classes of respectively 2057 and 2053 points. We will then predict the first class giving an 0.5 accuracy.

1.3.2 Support Vector Machines (SVM)

First we ran the SVM with a cross validation with following parameter: C= 0.01, H=3 and a test partition size of 0.2.
The accuracy of the model for each data set is :

- 0.816 for "MUTAG"
- 0.483 for "ENZYMES"
- 0.826 for "NCI1"

We observe that the accuracy is higher than the baseline classifier, our model is then working well. For the ENZYME data set the accuracy is only of 48% which is not seems a high score but when we look at the k-PCA plot and t-SNE plot, it's understandable that the prediction is not perfect.

1.3.3 Select hyperparameters

This part was very time consuming, we therefore modified the range of the parameter c. Instead of doing c-values in the range of 1e-5 to 10000, we choose 100 as the maximum value of c. Only "MUTAG" was done with the maximum range and the result matrix can be seen below. As we see with the maximum range of c=10000 the results do not vary but the computation time increase drastically so it was decided that the maximum value of c will be fixed at 100 for the two other data sets.

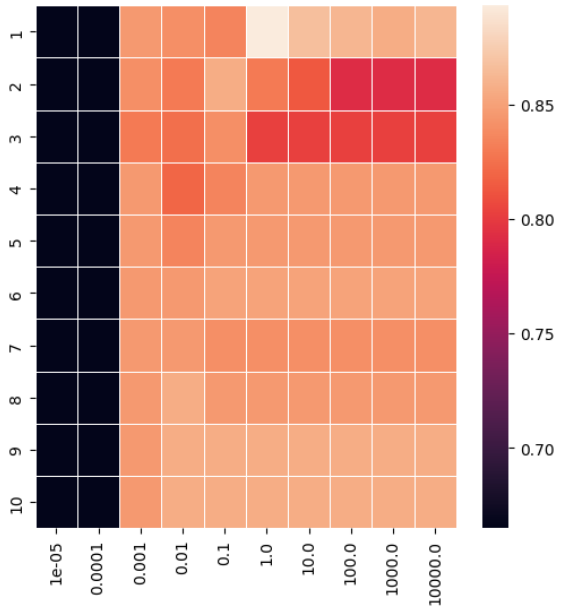

Figure 13: Heatmap "MUTAG" with max c=10000 (x=c, y=h)

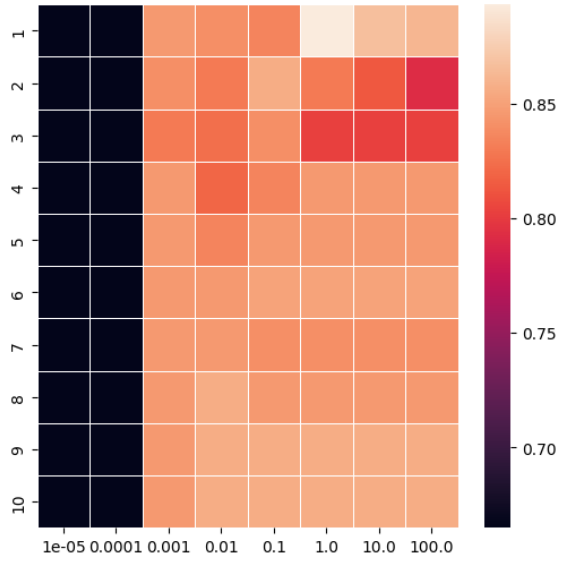

Figure 14: Heatmap "MUTAG" with max c=100 (x=c, y=h)

The results obtained for the hyperparameters are:

- h=1 and c=1 with an accuracy of 0.893 in 1.763 seconds for "MUTAG"
- h=3 and c=0.1 with an accuracy of 0.343 in 218.673 seconds for "ENZYMES"
- h=8 and c=0.01 with an accuracy of 0.816 in 5425.150 seconds for "NCI1"

The full results plot as heatmaps for two other data sets are just below.

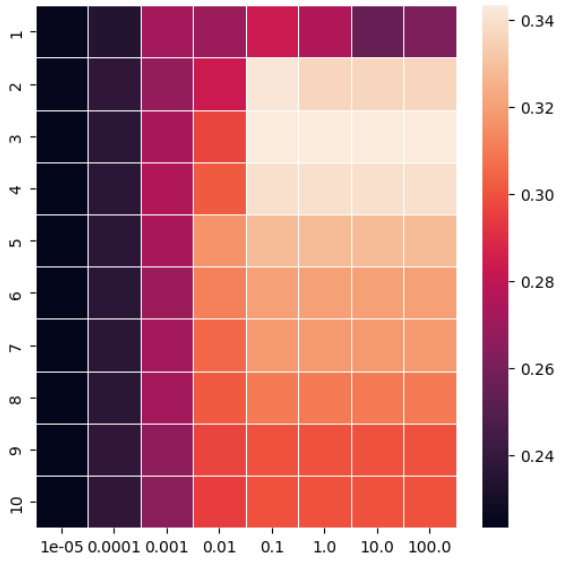

Figure 15: Heatmap "ENZYMES", (x=c, y=h)

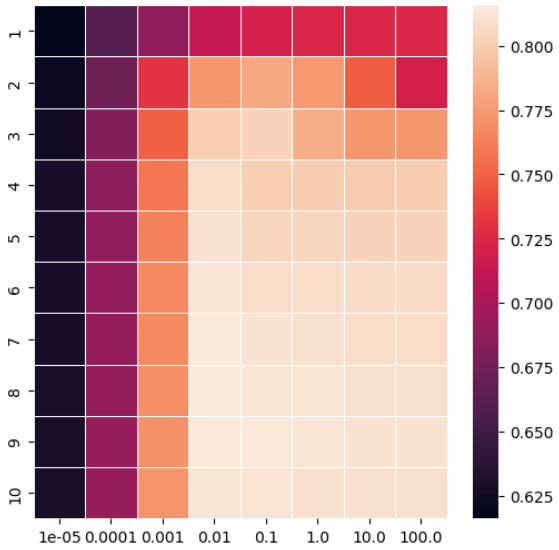

Figure 16: Heatmap "NCI1" (x=c, y=h)

The result for the first data set seems a bit strange because the maximum accuracy must be occur at a higher value of h just like the other results. It is also seen that the two lowest value of c are useless since their accuracy is very low compared to the other values.

1.4 Observation

We are quite impressed of the performance given by those techniques. Indeed the data points have a very high dimension, when we try to plot them using dimension reduction technique the plots are quite messy and we still get good accuracy.
We observe that using a simple kernel as the the Weisfeiler-Lehman sub-tree kernel on the data give impressive improvements in the results as a classic PCA won't give correct results. We are a bit disappointed on the plots of the t-SNE, we hoped to be able to plot it more clearly. It may come from the fact that we don't master this technique yet and that we did not tune parameter enough.