Practical session 1: Spectral Clustering

Gerard Castell and Victor Rubio

June 26, 2022

1 Mushroom clustering

1.1 Abstract

The goal of this experiment is to classify a set of mushrooms depending on if they are either edible or poisonous. With that purpose, we will use a dataset experiment which contains 8,124 samples taken from 23 species in the Agarius and Lepiota family. For each sample, there is a label indicating if the mushroom is edible or poisonous and there is also a feature vector detailing a number of its properties such as color, odor and size. In order to solve this classification problem, we will follow a spectral clustering approach to embed the graph nodes into a low-dimensional vector space. Then, k-means is performed on the embedded vectors in order to divide them into clusters.

1.2 Reading and cleaning up the data

After downloading the dataset from the UCI website we found out more information on how the dataset is built. We found that the labeling is classified into edible (e) or poisionous (p) and that there are 2480 missing values (denoted by "?") all in the 11th attribute.

In order to read and clean the dataset we code the following section to erase such feature from all the mushrooms, hence it is not considered for our experiment:

```
data = readtable('agaricus-lepiota.txt','ReadVariableNames',false);
data(:,12)=[];

data_label=data(:,1);
data_features=data(:,2:end);

data_label_c = categorical(data_label{:,:});
data_feature_c = categorical(data_features{:,:});
```

1.3 Feature vector encoding

With the data loaded and cleaned properly, the entries of the features vectors were converted to numerical values using an ordinal encoding:

```
[~,~, data_feature_vec]=unique(data_feature_c);
data_features_n=reshape(data_feature_vec , size(data_features));

[~,~, data_label_vec]=unique(data_label_c);
data_label_n=reshape(data_label_vec , size(data_label));
```

1.4 Forming a graph

Once the feature vector encoding was completed we used the *pdist2.m* method to build a weighted graph where each mushroom vector is a node, selecting at first the Hamming distance to calculate the weights:

```
A = pdist2(data_features_n ,data_features_n , 'hamming');

G=graph(A);
figure(1)
plot(G)
title('Mushrooms Graph')
```

With the previous code we were able to plot the following figure that is the representation of the nodes and weights of the graph. In such a plot we can see clearly that we have built an undirected graph which is fully connected. Each node represents a mushroom sample, so the edges between them are the weight calculated based on the similarity with the Hamming distance, so the intuition of that links is that the bigger the weight is the more similar the mushrooms are. Using this knowledge we have created the similarity matrix, which is the adjacency matrix (A), from that metric of the Hamming distance to end up with a symmetric graph of the samples and now we can apply the Spectral Clustering to classify them.

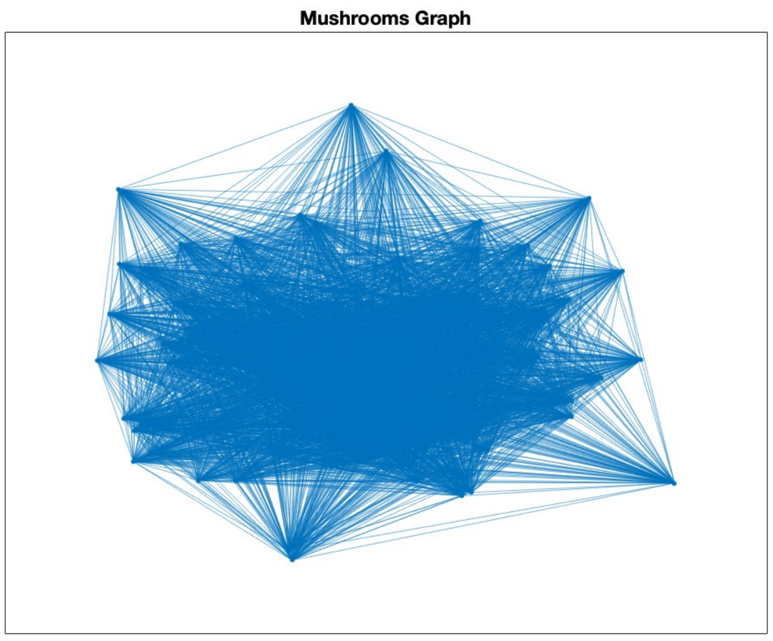

Figure 1: Plot of the nodes and edges of the graph built

It is important to note that spectral clustering performance largely depends on the similarity function used to build the graph. So the results obtained in this experiment are closely pegged to the Hamming metric.

1.5 Spectral clustering

The key concept behind the Spectral Clustering is that **the multiplicity of the 0 eigenvalue of the Laplacian of a graph is equal to the number of connected clusters** with associated eigenvectors given by the indicator vectors of those clusters. In an ideal world, the graph has k connected or clusters and the Laplacian has an eigenvalue 0 with multiplicity k. The k eigenvectors of the null eigenvalue are linear combinations of the indicator vectors and allow us to identify the nodes of each cluster, which is our main target. However, in a real scenario like the one for this experiment, **we expect to obtain small eigenvalues of the Laplacian**, which are an approximation of the null eigenvalues of the Laplacian, so the corresponding eigenvectors are an approximation as well of the indicator vectors.

Then, we wanted to embed the graph nodes into a low-dimensional vector space where data points shall be organized according to the graph weights, performing k-means on the embedded vectors in order to divide them into clusters afterwards. We used the normalized graph Laplacian, calculated eigenvalues and eigenvectors and used the first 2 eigenvectors with the k-means function *kmeans.m*:

```
on=ones(size(A,1),1);
D_vec=(A*on);
D_mat=diag(D_vec);
D_inv2=diag(D_vec.^(-.5));
L=D_mat-A;
nL=D_inv2*L*D_inv2;
[veps,vaps]=eig(nL);
k=2;
idx=kmeans(veps(:,1:2),k);
```

1.6 Performance evaluation

In order to determine the usefulness of the results we compared them to the ground truth using the confusion matrix and calculating the failure probabilities. We also plotted the feature vectors in the embedded space differentiating using colors the different clusters.

```
C=confusionmat(data_label_n, idx);
figure(2)
h=gscatter(veps(:,1), veps(:,2), idx);
title('Scatter plot. Normalized Laplacian. 2 clusters')

figure(3)
h_r=gscatter(veps(:,1), veps(:,2), data_label_n);
title('Sparcification using the real labels')
```

In order to compare the results we plotted a scatter of the sparcification using the real labels in the figure bellow. This way we will be able to graphically determine if the model is working properly.

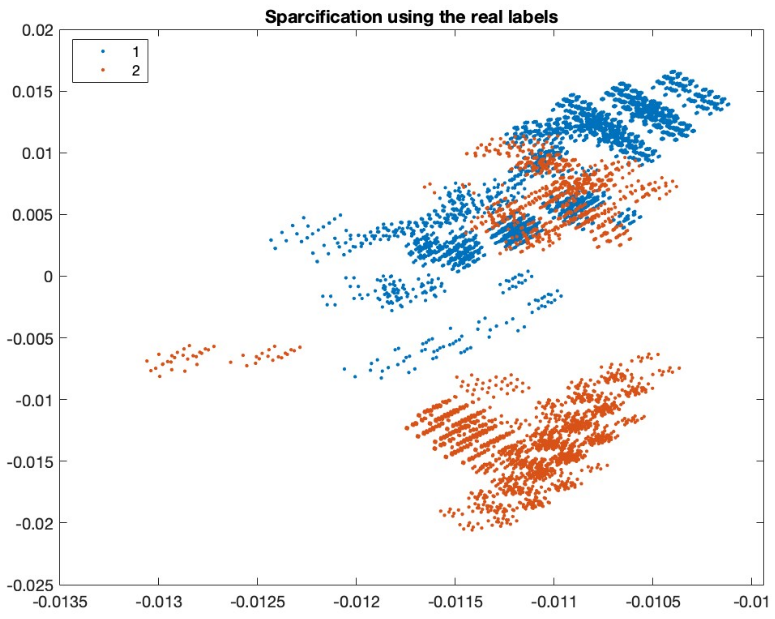

Figure 2: Scatter of the data segregated using the real labels

Due to our problem, we want to classify mushrooms into edibles or poisonous, we decided to use *k=2* in the k-means to have our samples into two clusters. We used a Normalized Laplacian as discussed previously. In the image bellow we can see how these two elements were separated by k-means.

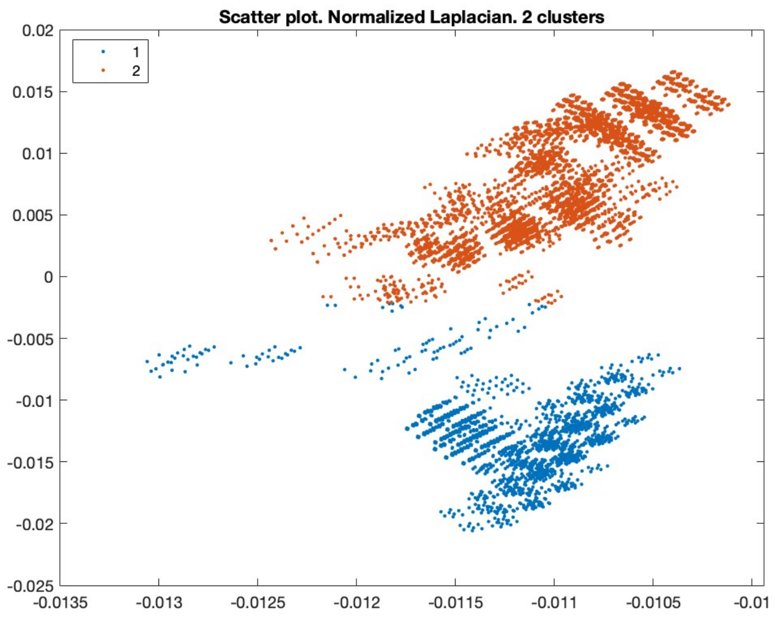

Figure 3: Cluster visualization with k=2 and Normalized Laplacian (L_{sn})

As you can observe comparing the both images above, there are some samples of one class quite mixed with the other cluster. This confuses the discriminator and makes the task more complicated. The classifier that we obtained clearly has separated all the samples from one class from the other, so we cannot find a mixed samples closely, they are separated almost linearly. As we see, with that approach we are misclassifying that mixed in the other cluster.

We also obtained the confusion matrix for the previous configuration:

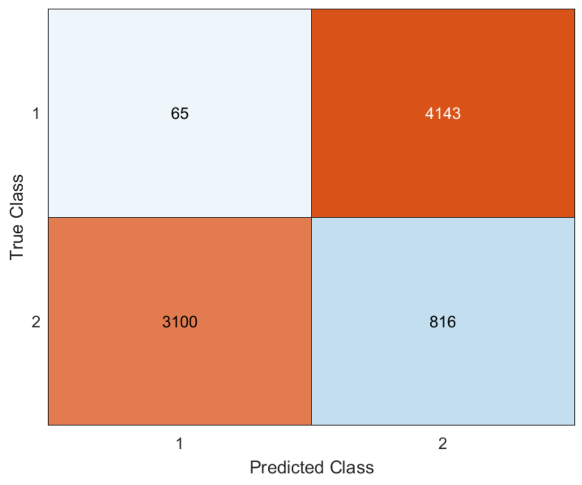

Figure 4: Confusion matrix of configuration k=2 and Normalized Laplacian (L_{sn})

So with the aim of find a better segregation results we try using k=3 and recompute the confusion matrix and plot again the results:

```
%% We repeat it with a different number of clusters
k=3;
idx3=kmeans(veps(:,1:3),k);
C3=confusionmat(data_label_n, idx3);
confusionchart(C3);
figure(4)
formats = [".r", ".b", ".g"];

for i = 1:3
    points = veps(idx3 == i, 1:3);
    scatter3(points(:, 1), points(:, 2), points(:, 3), formats(i), 'DisplayName'
    hold on
end

legend('Location', 'Best')
title 'Scatter plot. Normalized Laplacian. 3 clusters'
hold off
```

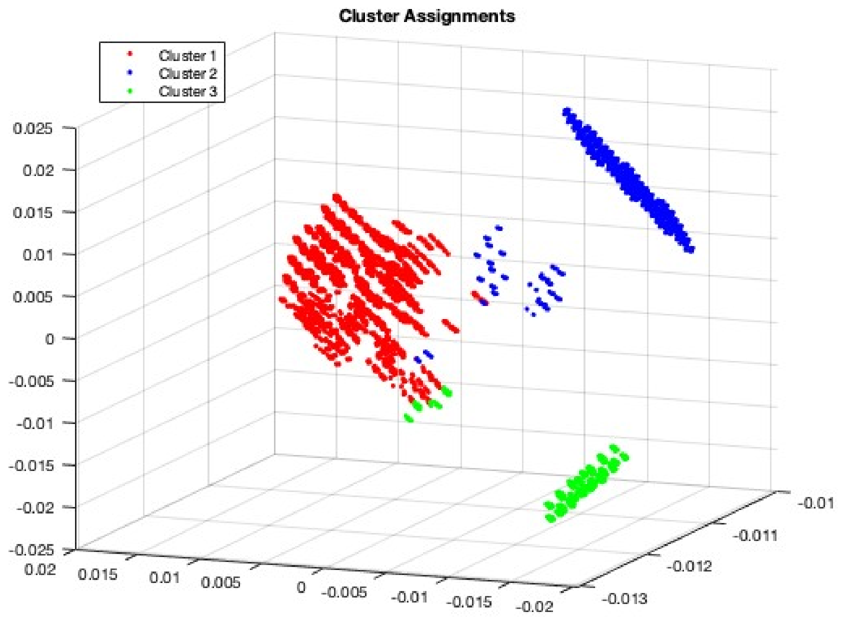

Figure 5: Cluster visualization with k=3 and Normalized Laplacian (L_{sn}) in a 3D view

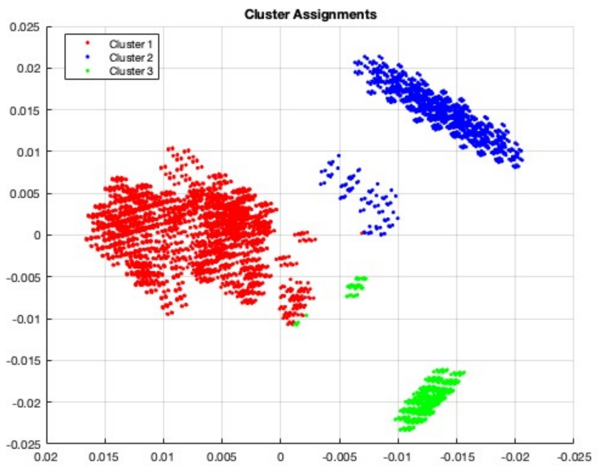

Figure 6: Cluster visualization with k=3 and Normalized Laplacian (L_{sn}) in a 2D view

As we may observe in the previous figure, three clusters are clearly visible, which have been projected onto the the first three eigenvectors as axis and also onto the second and third coordinates in the eigenvector basis. We compute the confusion matrix again for this configuration:

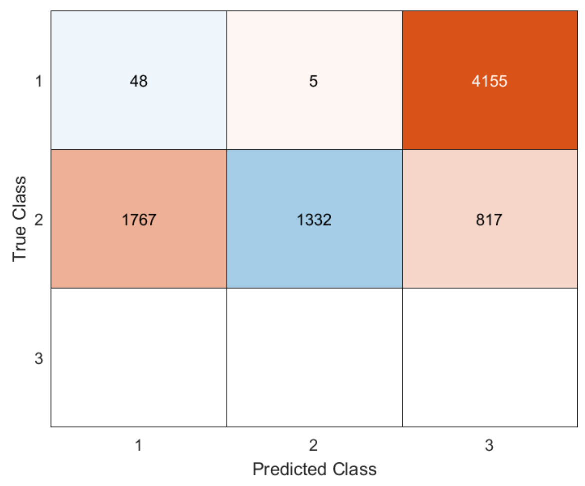

Figure 7: Confusion matrix of configuration k=3 and Normalized Laplacian (L_{sn})

With the previous confusion matrix we can see that the correct predictions have increased when increasing the number of eigenvectors used.

1.7 Alternative embeddings and graph sparsification

After adapting the k value in k-means we also wanted to try to classify the elements introducing different embeddings. We used the Unnormalized Laplacian at first, using the first two eigenvalues, although the result was worse than the one we obtained from the normalized laplacian:

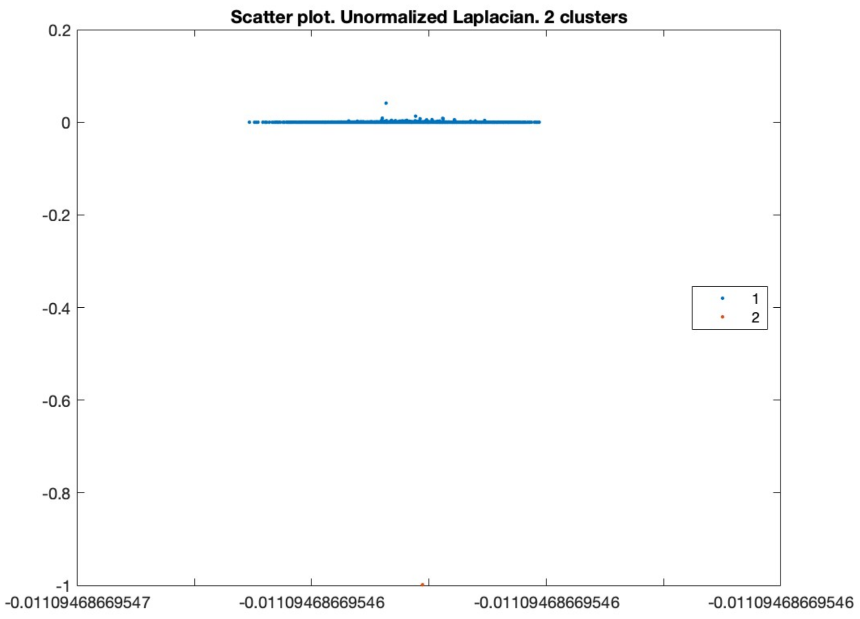

Figure 8: Cluster visualization with k=2 and Unnormalized Laplacian (L)

In this case we can see that is has only been able to classify one mushroom as poisonous. If we look at the confusion matrix it is evident that the results are worse:

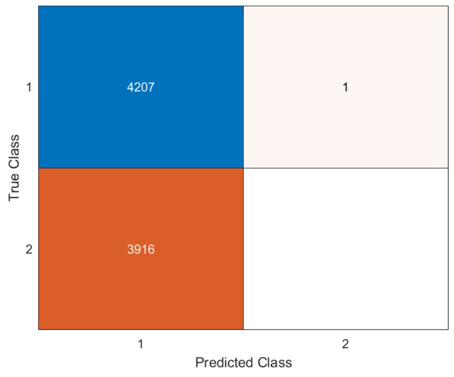

Figure 9: Confusion matrix of configuration k=2 and Unnormalized Laplacian (L)

Finally we also computed the Random-Walk Laplacian Matrix and using the first two eigenvalues with k-means we plot the following figure:

```
D_inv=diag(1./D_vec);
rwL=D_inv*L;
[veps_rw,vaps_rw]=eig(rwL);
k=2;
idx_rw=kmeans(veps_rw(:,1:2),k);
figure(6)
h_rw=gscatter(veps_rw(:,1), veps_rw(:,2), idx_rw);
title('Scatter plot. Random-walk Laplacian. 2 clusters')
```

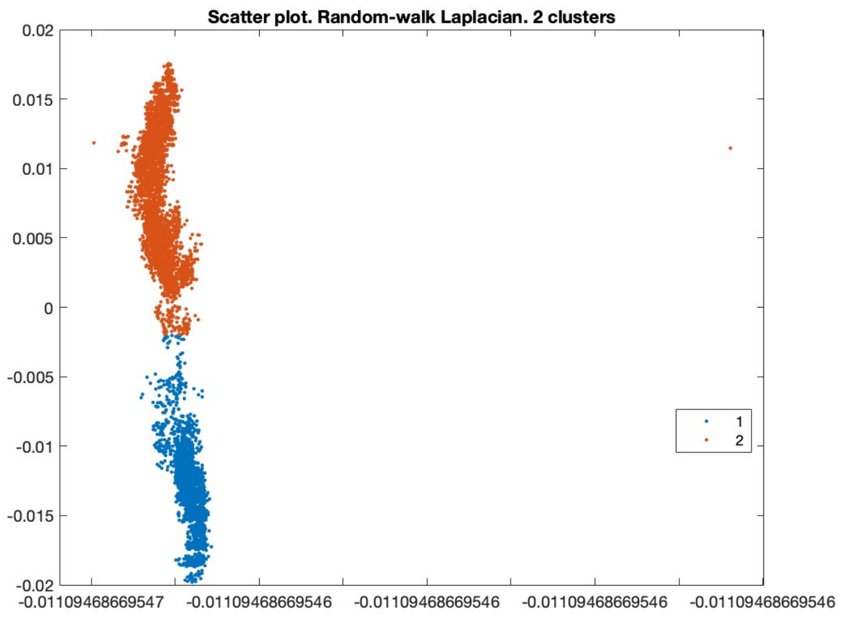

Figure 10: Cluster visualization with k=2 and Random-Walk Laplacian (L_{RW})

If we look at the confusion matrix for this configuration we can observe that the result has greatly improved as the amount of correct predictions outweighs the amount of false predictions. This model still can be improved because if we want to detect poisonous mushrooms the false positives should be significantly lower as to avoid risking lives.

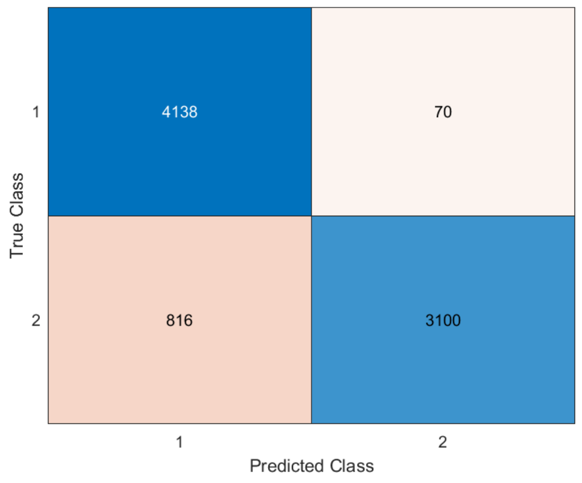

Figure 11: Confusion matrix for configuration k=2 and Random-Walk Laplacian (L_{RW})