Laboratory 4

In this laboratory we will analyze K-Means and hierarchical clustering methods.

Clustering methods are unsupervised, i.e. they do not consider possible sample labels, but simply aim at grouping together samples that are similar according to a similarity criterion. In this laboratory we will employ K-Means and hierarchical clustering, with Euclidean distance as metric, to cluster the Iris dataset samples.

K-Means

We start considering the K-Means algorithm. First of all, we load the Iris dataset (you can reuse the same functions we developed in previous laboratories):

```
D, L = load_iris()
```

For this laboratory and for the project we **employ** the `scipy` **library implementation** of the K-Means algorithm. This requires that we first import the `scipy.cluster.vq` module:

```
import scipy.cluster.vq
```

The library provides different K-Means implementations, in the following we employ `kmeans2`.

Since the algorithm employs random initialization, to make the results reproducible we first seed the `numpy` random number generator (in the examples we use 0 as seed)

```
numpy.random.seed(0)
```

Note that we won't use the labels array `L` for training, since K-Means is unsupervised.

The `scipy` library assumes data matrices represent samples as stacked row vectors, whereas in the other laboratories and during lectures we assume data matrices contain stacked column vectors. Therefore, we need to transpose our data matrix before calling the `kmeans2` function. We can compute the cluster centroids and cluster labels as

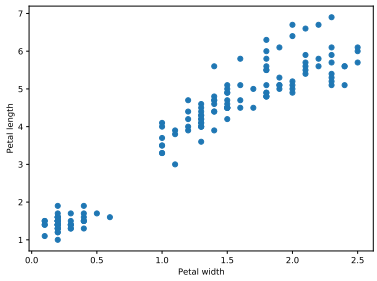
```
centroids, clusterLabels = scipy.cluster.vq.kmeans2(D.T, n, minit='points', iter=100)
```

where `n` is the desired number of clusters, and `minit` corresponds to the initialization algorithm (in our case we set `mini='points'`, i.e., we initialize centroids with randomly selected samples). Parameter `iter` controls the maximum number of iterations of the algorithm. The function returns a tuple that corresponds to the centroids (as stacked **row** vectors) and cluster labels. To be consistent with our samples representation, we transpose the centroid matrix so that centroids are represented as **column** vectors.

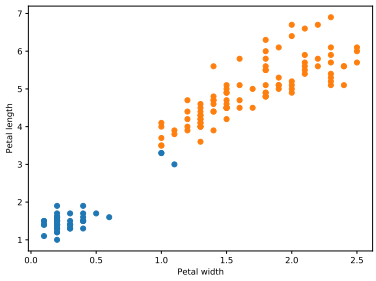
```
centroids = centroids.T
```

Try running the algorithm with different values for the cluster number `n`. Below you can find the scatter plots for the last two features, colored according to the cluster labels. You can obtain similar plots using the following function (note that you may obtain different cluster colors):

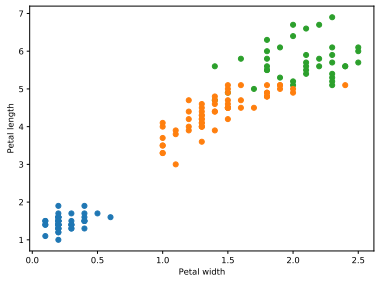
```
def plot_scatter(D, clusterLabels):
    plt.figure()
    for label in range(clusterLabels.max()+1):
        plt.plot(D[3, clusterLabels == label], D[2, clusterLabels == label], 'o')
    plt.show()
```



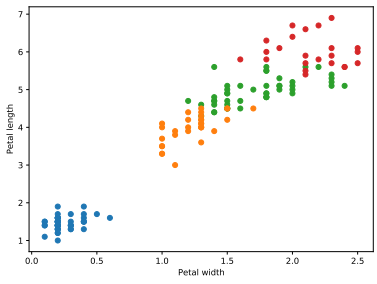
1 Cluster



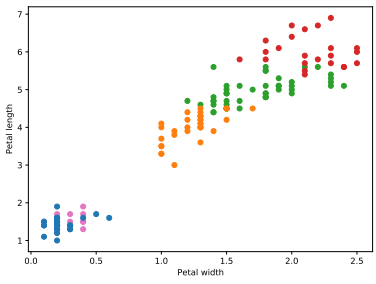
2 Clusters



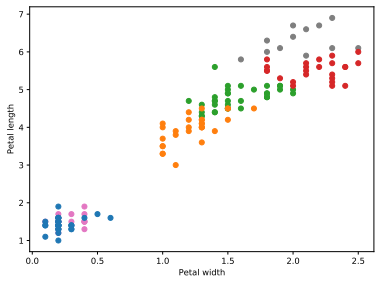
3 Clusters



4 Clusters



5 Clusters

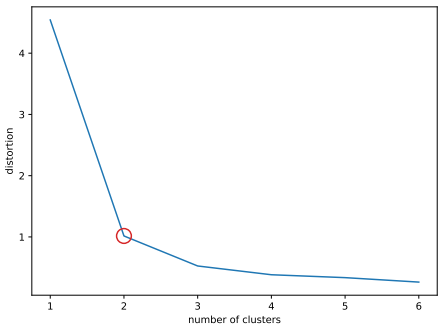


6 Clusters

To select the number of clusters we now turn our attention to the distortion plot. To compute the distortion, we need to compute the average square distance of each point from the centroids of its cluster. For a given solution `(centroids, clusterLabels)` we can compute the corresponding distortion with the following function:

```
def compute_distortion(D, centroids, clusterLabels):
    distortion = 0
    for i in range(D.shape[1]):
        distortion += numpy.linalg.norm(D[:, i] - centroids[:, clusterLabels[i]])**2
    return distortion / D.shape[1]
```

Plot the distortion as a function of the number of clusters. You should get



The red circle shows an "elbow", suggesting that there may be 2 clusters.

NOTE: remember that the "elbow" method is a heuristic, and does not necessarily lead to a good number of clusters. Furthermore, given that different runs of the algorithm with different number of clusters employ different random initialization, we cannot guarantee that the distortion will decrease when we increase the number of clusters. This could be guaranteed by further modifications of the algorithm, which, however, are beyond the scope of this laboratory.

Hierarchical clustering

We now turn our attention to hierarchical clustering. Also in this case we will rely on the `scipy` library. We need to import `scipy.cluster.hierarchy`

```
import scipy.cluster.hierarchy
```

Hierarchical clustering is implemented by the `scipy.cluster.hierarchy.linkage` function, which computes the *linkage* matrix that contains the information required to reconstruct the whole clustering tree (dendrogram). In this laboratory we won't analyze the linkage matrix structure, but we will simply extract "flat" clusters, i.e., clusters obtained by cutting the cluster dendrogram at different levels.

To compute the linkage matrix you can use

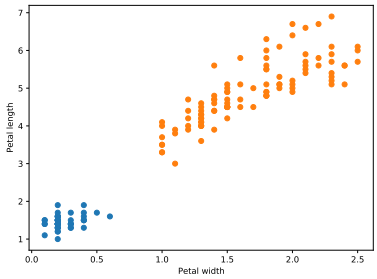
```
linkageMatrix = scipy.cluster.hierarchy.linkage(D.T, method = method, metric = 'euclidean')
```

where for `method` you can pass `'single'`, `'complete'` or `'average'` to obtain single, complete or average (UPGMA) linkage, respectively.

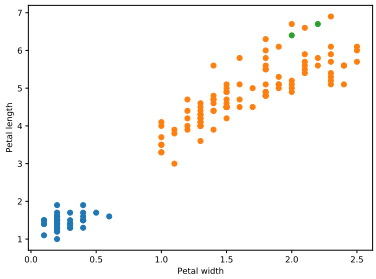
Given the linkage matrix, you can obtain `n` clusters by

```
clusterLabels = scipy.cluster.hierarchy.fcluster(linkageMatrix, n, criterion = 'maxclust')
```

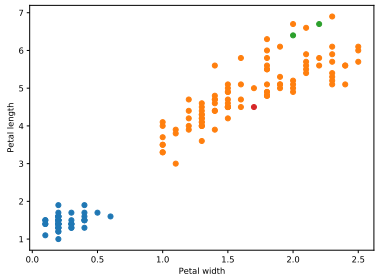
Below you can find the results for the different clustering methods with different number of clusters (2, 3 and 4):



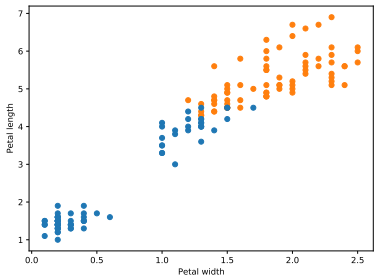
Single linkage, 2 Clusters



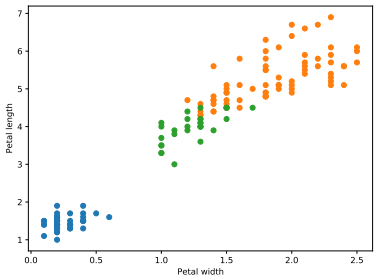
Single linkage, 3 Clusters



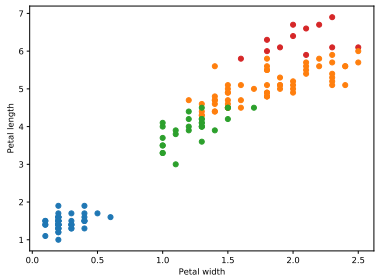
Single linkage, 4 Clusters



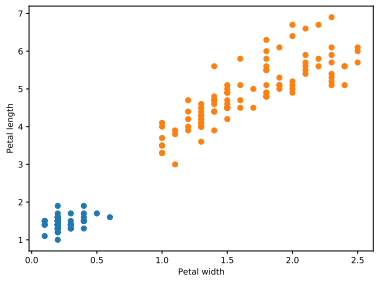
Complete linkage, 2 Clusters



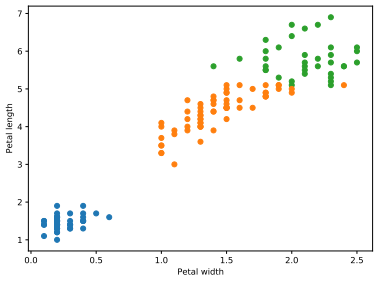
Complete linkage, 3 Clusters



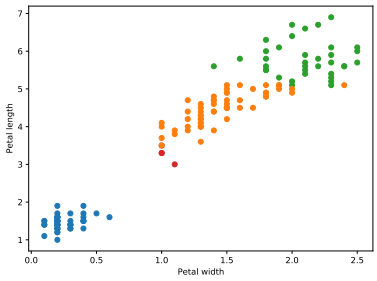
Complete linkage, 4 Clusters



Average linkage, 2 Clusters



Average linkage, 3 Clusters



Average linkage, 4 Clusters

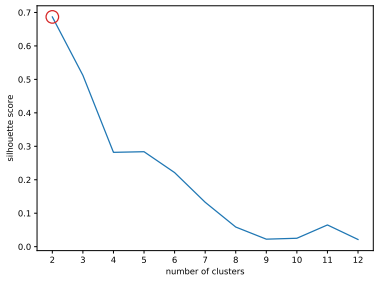
To choose the number of clusters in this case we use the silhouette score. It can be computed using the `sklearn.metrics.silhouette_score` function. We need to import `sklearn.metrics`

```
import sklearn.metrics
```

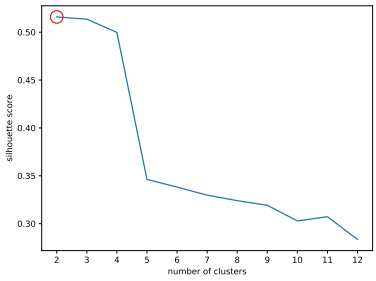
Given the data matrix `D` and the cluster labels `clusterLabels`, we can compute the silhouette score as

```
score = sklearn.metrics.silhouette_score(D.T, clusterLabels)
```

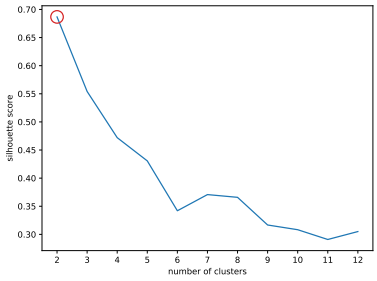
Compute the silhouette score for different number of clusters n (pay attention that the silhouette score is defined only for $n > 1$), and plot the results:



Single linkage, silhouette



Complete linkage, silhouette



Average linkage, silhouette

The silhouette maximum is obtained for $n = 2$, suggesting the presence of 2 clusters.

Remark: the Iris dataset contains 3 classes. However, our analysis suggests the presence of only 2 clusters. This should not be surprising: if two classes contain similar elements, clustering methods will tend to merge them. This is consistent with the idea behind clustering algorithms, i.e., merging together samples that are similar according to the chosen criterion (Euclidean distance in this laboratory).

Project

We have seen in the first laboratories that the project data scatter plots suggest the presence of clusters inside each class. In this project part we will analyze the clusters of class True (label 1). We will restrict the analysis to the last two features of the dataset, and treat the samples of class True as if they were an unlabeled dataset to analyze.

Extract from the project dataset the samples belonging to class True. For these samples, consider only the last two features (you can extract the required data as `D[4:, L == 1]`). Apply the K-Means algorithm with different number of clusters n, and visualize the solutions through scatter plots. What can you conclude on the quality of the solutions? Are the results consistent with a qualitative analysis of the features (i.e., scatter plots)? Compute the distortion, and apply the "elbow" heuristics. How many clusters would you select? Is this reasonable, given the data scatter plots?

Repeat the analysis with hierarchical clustering, considering the three different linkage methods we analyzed earlier. This time, employ the silhouette criterion to select the number of clusters. Again, plot the cluster solution and compare it with the K-Means one. How do the different methods work? How many clusters would you identify according to the silhouette criterion?