Data Mining - Handin 1 - Clustering and outlier detection

Welcome to the handin on clustering algorithms and outlier detection. This handin corresponds to the topics in Week 5--9 in the course.

The handin IS

- mandatory
- done individually
- worth 10% of the grade

For the handin, you will prepare a report in PDF format, by exporting the Jupyter notebook. Please submit

1. The jupyter notebook file with your answers
2. The PDF obtained by exporting the jupyter notebook

Submit both files on Blackboard no later than **March 14th kl. 23.59**.

1. Theoretical Questions

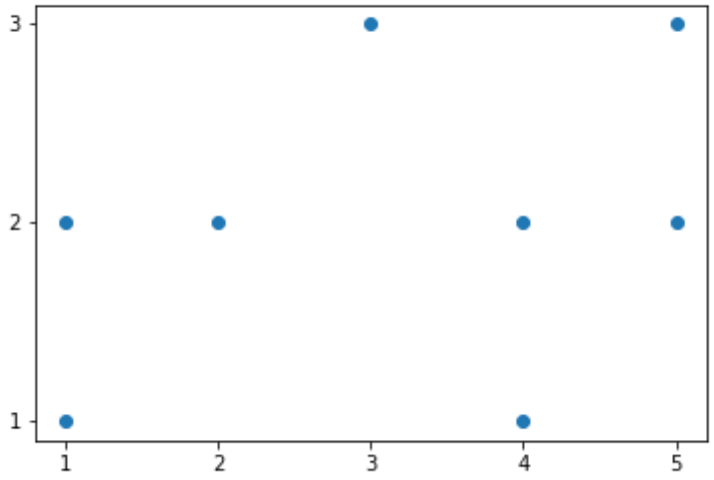
1. You are given a metric $d_{old}(x, y)$ where $x, y \in \mathbb{R}$, prove that $d_{new}(x, y) = \frac{d_{old}(x,y)}{1+d_{old}(x,y)}$ is also a metric
   - Suggestion: remember monotonicity of $f(t) = \frac{t}{1+t}$

In [ ]:
```
### YOUR ANSWER
```

1. Show that $\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu}^\top) \cdot (x_i - \hat{\mu}^\top)^\top = E[(X - \hat{\mu})(X - \hat{\mu})^\top]$

In [ ]:
```
### YOUR ANSWER
```

1. **The calculations in this exercise have to be done by hand**. Given the dataset below:
   - Calculate by hand the cluster assignments using k-means and $k = 2$ and report the results
   - Show two examples with two different initial cluster assignments that lead to a different result.
   - Compute the dendrogram
   - Run density-based clustering with $\epsilon = 1$ and $MinPts = 2$



In [ ]:
```
### YOUR ANSWER
```

1. After looking at the dataset below, you realize that the clusters are elliptic rather than spherical. You want to detect the red outlier point, assuming you know that is an outlier.
   - Would you use a distance-based outlier detector to detect the outlier?
   - Would you use a depth-based approach?
   - Motivate your answers!

In [ ]:
```
### YOUR ANSWER
```

1. Consider the following scenario: you have been asked to provide a subspace clustering of a data set. Since you already have an implementation of PCA for dimensionality reduction and a full space clustering (such as k-means or DBSCAN, the precise choice is not relevant for this question) available, you consider just using PCA and then full space clustering. Are the results expected to be similar? Why/why not?

In [ ]:
```
### YOUR ANSWER
```

2. Preliminary analyses and considerations

In this section, you will perform preliminary analyses on your data. These preliminary analysis are useful to understand how the data behaves, before running complex algorithms.

In [ ]:
```
import numpy as np
import pandas as pd
import io
import requests
import warnings
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.cluster import KMeans, DBSCAN
import time


warnings.filterwarnings('ignore')
%matplotlib inline
import matplotlib.pyplot as plt
```

In [ ]:
```
def read_data_online(url, headers, delimiter=","):
    s = requests.get(url).content
    data = pd.read_csv(io.StringIO(s.decode('utf-8')), sep=delimiter, names=head
    return data

headers = ['class', 'Alcohol', 'Malic_acid', 'Ash', 'Alcalinity_of_ash', 'Magnes
wines = read_data_online("https://archive.ics.uci.edu/ml/machine-learning-databa
```

In [ ]:
```
wines.head()
```

In [ ]:
```
data = wines.to_numpy()
X = data[:,1:]
y = data[:,0]
y = y.astype(int) - 1
rows, cols = np.shape(X)
```

1. Compute the covariance matrix among all the attributes

In [ ]:
```
### YOUR CODE HERE
```

1. Normalize the features using standard score normalization and range normalization. Plot the two covariance matrices.

In [ ]:
```
### YOUR CODE HERE
```

Discuss how the covariance matrix change and find a reason why such behaviour appears. What are the differences between the two normalizations?

In [ ]:
```
### YOUR ANSWER
```

1. Sometimes it is convenient to know whether a variable is close to a normal distribution. Implement a method norm_dist that: 1) Inputs the number of buckets b and a vector x of values 2) Outputs the sum of the absolute differences of the buckets between the a histogram with b buckets of a normal variable with μ, σ deviation corresponding sample mean and standard deviation of the input vector and the histogram of the input vectors with b buckets. The sum of the differences is computed as

$$\sum_{i=1}^{b} |H_X(i) - H_{\mathcal{N}}(i)|$$

where $H_X(i)$ is the i-th bucket of the histogram of x and $H_{\mathcal{N}}(i)$ is the i-th bucket of the hisotgram obtained from the normal distribution $\mathcal{N}(\mu, \sigma^2)$.

Is the method a good method? Is it robust to outliers? Run your code on each columns of the dataset. What is the one with the largest distance? Compare the norm_dist of each attribute feature in the dataset.

In [ ]:
```
def norm_dist(x, b):
    dist = 0
    ### YOUR CODE HERE

    ### YOUR CODE HERE
    return dist
```

In [ ]:
```
### YOUR ANSWER
```

Now look at the method below. This is call a Quantile-Quantile Q-Q plot.

In [ ]:
```
from scipy import stats
from matplotlib import gridspec

_, n = data.shape

plt.tight_layout()

fig = plt.figure(constrained_layout=True, figsize=(8, 30))
spec = gridspec.GridSpec(ncols=2, nrows=(n-1), figure=fig)
for i in np.arange(1,n):
    x = data[:,i]
    r = i-1
    qq = fig.add_subplot(spec[r, 1])
    stats.probplot(x, plot=qq)
    h = fig.add_subplot(spec[r, 0])
    h.set_title(headers[i])
    h.hist(x, bins = 30)
```

1. Discuss why this method is more robust than the one we proposed.

In [ ]:
```
### YOUR ANSWER
```

1. Find the two attributes that are correlated the least among each other.

In [ ]:
```
### YOUR ANSWER
```

3. Cluster analysis

In this section, you will perform cluster analysis of the dataset above and modify clustering algorithms to achieve better results.

1. Implement and use the elbow method detect the number of clusters. For plotting you can use distortion, which is the sum of squared distances to the assigned cluster centroid.

In [ ]:
```
### YOUR CODE HERE
```

1. Run k-means on the dataset, with the number of clusters detected in the previous exercise.

Note: you can use the KMeans implementation from scikit-learn

```
kmeans = sklearn.cluster.KMeans(n_clusters).fit(X)
clusters = kmeans.labels_
```

In [ ]:
```
### YOUR CODE HERE
```

1. Implement Kernel K-means and the Gaussian Kernel.

The Gaussian kernel is defined as in the following equation:

$$K\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\sigma^2}\right)$$

In [ ]:
```
def gaussian_kernel(x, y, sigma=0.8):
    k = 0
    ### YOUR CODE HERE


    ### YOUR CODE HERE
    return k


def kernel_kmeans(X, n_clusters, kernel=gaussian_kernel, iters=100, error=.01):
    ### YOUR CODE HERE


    ### YOUR CODE HERE
    return clusters

clusters = kernel_kmeans(X_norm, 3)
```

We will now visualize our clusters with the method below

In [ ]:
```
## DO NOT MODIFY

def plot_clusters(X, clusters):

    time_start = time.time()
    tsne = TSNE(n_components=2, verbose=1, random_state=0)
    X_2d = tsne.fit_transform(X)
    print('t-SNE done! Time elapsed: {} seconds'.format(time.time()-time_start))

    from matplotlib import pyplot as plt
    plt.figure(figsize=(6, 5))
    colors = ['b','r','g', 'm', 'c', 'y', 'k', 'tan', 'gold','sienna', 'navy','d
    for i in np.arange(np.shape(data)[0]):
        plt.scatter(X_2d[i, 0], X_2d[i, 1], c=colors[clusters[i]])
    plt.show()
```

In [ ]:
```
# Assumes your normalized data has been saved as "X_norm"
plot_clusters(X_norm, y)
plot_clusters(X_norm, clusters)
```

The visualization above, use a method called t-SNE to plot the data into a 2D space. The method is not part of the course, but it is a good tool for data analysis that should be always considered.

1. Above you have seen how to measure cluster quality. Now we will implement, Normalized Mutual Information, another measure for cluster quality. Mutual information for two clusterings U and V is defined as $\mathrm{MI}(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|}\frac{|U_i\cap V_j|}{N}\log\frac{|U_i\cap V_j|}{|U_i||V_j|}$

In [ ]:
```
def NMI(C1, C2):
    mi = 0
    ### YOUR CODE HERE


    ### YOUR CODE HERE
    return mi
```

1. Plot the NMI among the class labels y and the clusters you found with k-means.
   - Reason about the measure, why is it a good measure?
   - What does the measure capture?
   - Plot purity and compare the two measures at increasing number of clusters.

In [ ]:
```
### YOUR CODE
```

In [ ]:
```
### YOUR ANSWER
```

1. Inspect again the clustering results above, both in terms of NMI and with t-SNE.

- What other clustering algorithm could be used instead of K-Means?
- Would Density-based clustering work better? Why?
- Run DB-Scan with parameters $\varepsilon = 0.5, minPts = 3$ from sklearn and compare the results with k-means. What do you notice?

In [ ]:
```
### YOUR ANSWER

dbscan = DBSCAN(eps=0.5, min_samples=3).fit(X_norm)
clusters_dbs = dbscan.labels_
plot_clusters(X_norm, clusters_dbs)
```

1. DBScan requires tuning of parameters $\varepsilon, MinPts$. Implement the heuristic strategy in the slides to find the best parameters. Run the code and see whether the results with DBScan improve.

In [ ]:
```
def tune_dbscan(X):
    eps = 0
    pi = 0
    ### YOUR CODE HERE


    ### YOUR CODE HERE
    return eps, pi
```

1. DBScan is an expensive algorithm. Assume you have been given e very big dataset. After thinking, you realise that you can create a number of samples of the points, run DBScan on the samples, assign the points to the closest cluster.
   - What is the disadvantage of such an approach?
   - What is wrong if you consider a uniform sample (every point equal probability) without replacement? What kind of sampling strategy would you devise to preserve the results of DBScan from the original dataset to the sampled one?

In [ ]:
```
### YOUR ANSWER
```

1. In this last point, we will try to implement a simple subspace clustering algorithm and compare the results with CLIQUE imeplemented in Week 8.
   - Take all subsets of 2,3 attributes.
   - Run dbscan on each subset.
   - Compute NMI for each subset.
   - Keep the k subsets with the largest NMI.

Note: You may have to experiment a lot with eps and MinPts to get reasonable clusterings

In [ ]:
```
### YOUR CODE HERE
```

Analyze:

- Advantages and disadvantages of the proposed algorithm
- Results with respect to CLIQUE

In [ ]:
```
### YOUR ANSWER HERE
```

4. Outlier detection

In this exercise we will work with outlier detection techniques and analyze their performance on the small dataset.

1. We will now compare two outlier detection techniques. We will first implement a simple distance-based outlier detector. This is the distance-based outlier detection from the lectures, where a point is considered an outlier if at most a fraction pi of the other points have a distance less of than eps to it.

In [ ]:
```
def DBOutliers(X, eps, pi):
    outliers = None
    ### YOUR CODE HERE


    ### YOUR CODE HERE
    return outliers
```

1. DBOutliers requires tuning the parameters eps, pi. Discuss how the results vary with those parameters. Plot the number of outliers as a function of eps and as a function of pi.

In [ ]:
```
### YOUR CODE
```

In [ ]:
```
### YOUR ANSWER
```

1. Propose a heuristic method to tune parameters eps, pi.

In [ ]:
```
def tune_dboutliers(X):
    eps = 0
    pi = 0
    ### YOUR CODE HERE


    ### YOUR CODE HERE
    return eps, pi
```

1. Using the parameters eps=0.9, pi=0.8 (and using the normalized data-set) compare the results of DBOutliers with those of LOF. What are the main differences? What outliers do you find?

In [ ]:
```
### YOUR ANSWER
```

1. In the last exercise we will try to detect outliers in different subspaces in a way similar to exercise 3.9, but we in this case we will use the Local Outlier Factor (LOF) measure with $k = 3$ to determine the quality of the clasters.

Take the subspace with the highest LOF, what outliers do you obtain? Looking at the dataset and the LOF measure, can you explain why these are considered outliers?

In [ ]:
```
### YOUR CODE
```