Telecom Paris | **MDI343**
2019/2020 | **Pavlo Mozharovskyi**

TP 9 : Depth and depth-based classification

- Discovering Python -

To start with or find a reminder for the Python language you may use the following materials:

- http://www.python.org
- http://scipy.org
- http://www.numpy.org
- http://scikit-learn.org/stable/index.html
- http://www.loria.fr/~rougier/teaching/matplotlib/matplotlib.html
- http://jrjohansson.github.io/

- Questions -

1) Consider the following multivariate normal distributions:
   - $\mathcal{N}(\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right], \left[\begin{smallmatrix}1&1\\1&4\end{smallmatrix}\right])$ which we will call MVN1 in the sequel;
   - $\mathcal{N}(\left[\begin{smallmatrix}2\\2\end{smallmatrix}\right], \left[\begin{smallmatrix}1&1\\1&4\end{smallmatrix}\right])$ which we will call MVN2 in the sequel;
   - $\mathcal{N}(\left[\begin{smallmatrix}2\\2\end{smallmatrix}\right], \left[\begin{smallmatrix}4&4\\4&16\end{smallmatrix}\right])$ which we will call MVN3 in the sequel.

   From each of them, draw a data set containing 25 points and plot it, each on a separate plot.

2) Program a function `depthMah` that computes the **Mahalanobis** depth and takes as input arguments the points for which the depth should be computed and the data set with respect to which the depth is to be computed. Apply this function to each of the data sets from question 1, *i.e.*, calculate the depth of each point of the data set with respect to the data set itself; plot the data sets indicating on the plots depth value (rounded to two digits after the period) next to each point.

3) Program a function `depthTuk` that computes the **Tukey** depth and takes as input arguments the points for which the depth should be computed, the data set with respect to which the depth should to be computed, and the number of random directions. To compute the Tukey depth, exploit the (weak) projection property, *i.e.*, compute the Tukey depth as the minimal univariate depth of the data projection on directions uniformly distributed on the unit sphere. This approximation approach is called the **random Tukey depth**. Apply this function (with 100 random directions) to each of the data sets from question 1, *i.e.*, calculate the Tukey depth of each point of the data set with respect to the data set itself; plot the data sets indicating on the plots depth value (rounded to two digits after the period) next to each point.

4) Program the **maximum depth classifier** where the outsiders are classified using the 1NN classifier. In implementation, create a class `MaxDepthClassifier` that takes as arguments of the function `__init__` name of the depth notion `depthName` and the number of random directions used to approximate the depth (if this is the case) `ndirs` and contains functions `fit` and `predict`. To test it, create a data set that has two classes: one class contains 250 points from MVN1 and another class contains 250 points from MVN2 (one can call this configuration the **location alternative**). Split this data set randomly into two equal parts: a train set and a test set; plot the train set with classes in different colors. Train the maximum depth classifier with the Mahalanobis depth on the train set and report its error rate on the test set.

5) Plot the DD-plot:

   (a) For the data set generated in question 4, plot (again) the train set with classes in different colors. Then, plot the DD-plot of this train set using the Mahalanobis depth, again with classes in different colors.

   (b) Create a data set that has two classes: one class contains 250 points from MVN1 and another class contains 250 points from MVN3 (one can call this configuration the **location-scale alternative**). Split this data set randomly into two equal parts: a train set and a test set; plot the train set with classes in different colors. Then, plot the DD-plot of the train set using the Mahalanobis depth, again with classes in different colors.

6) Program the ***DD*-plot classifier** by employing kNN in the DD-plot (with fixed k, say equal to 11) where the outsiders are classified using the 1NN classifier. In implementation, create a class `DDkNNClassifier` that takes as arguments of the function `__init__` name of the depth notion `depthName`, the number of random directions used to approximate the depth (if this is the case) `ndirs`, the number of the k-nearest neighbors for the kNN employed in the DD-plot `k`, and contains functions `fit` and `predict`. Train the DD-plot classifier with the Mahalanobis depth on the train set and report its error rate on the test set (both generated in question 5b).

7) In the same way as in question 6, train (on the train set) and the report the error rate (on the test set) of the maximum depth classifier with the Mahalanobis depth for the data set from question 5b.

8) A random vector X in $\mathbb{R}^d$ is said to be generated from a **multivariate Student-*t* distribution** with ν degrees of freedom with center $\mu \in \mathbb{R}^d$ and scatter matrix $\Sigma \in \mathbb{R}^{d\times d}$ if it can be represented as:

$$X = \boldsymbol{\mu} + \sqrt{\frac{\nu}{W_\nu}} Z \,,$$

   where W_ν is a variable following chi-squared distribution with ν degrees of freedom and Z follows a d-variate normal distribution centered in the origin and with the covariance matrix Σ $\mathcal{N}(0_d, \Sigma)$, W_ν and Z being independent.

   Student-t distribution with 1 degree of freedom is called the Cauchy distribution.

   For more information on chi-squared distribution an a related Gamma distribution see: https://en.wikipedia.org/wiki/Chi-squared_distribution and https://en.wikipedia.org/wiki/Gamma_distribution.

   Program a function `rmvt` that generates a data set from a multivariate Student-t distribution and takes as arguments center μ, scatter matrix Σ, number of degrees of freedom ν, and number of points to generate.

   Draw a data set from the bivariate Cauchy distribution with the same parameters as MVN1 (let us call this distribution MVC1) containing 25 points and plot it. Draw another data set from the bivariate Cauchy distribution with $\boldsymbol{\mu} = (1, 1)^\top$ and $\Sigma = \left[\begin{smallmatrix} 1 & 1 \\ 1 & 4 \end{smallmatrix}\right]$ (let us call this distribution MVC2) containing 25 points and plot it well.

9) Create a data set that has two classes: one class contains 250 points from MVC1 and another class contains 250 points from MVC2 (also a location alternative). Split this data set randomly into two equal parts: a train set and a test set. For the train set, provide two DD-plots, one obtained using Mahalanobis depth and one obtained using the random Tukey depth approximated with 100 directions; plot the classes in different colors.

10) Compare the two DD-plot classifiers (one with Mahalanobis depth and one with the random Tukey depth approximated using 100 directions) on the data set form question 9 by training them on the train set and reporting their error rate on the test set. You may would like to run questions 9 and 10 several times to better examine the error on data generated with different seeds.