Grouping the dataset based on similarity

http://en.wikipedia.org/wiki/K-medoids

K –medoids

For the application that need to be performed in this lab, it will have as input the file with all the points that was generated in the first lab (called later "points file"). This file will be considered the training data set for the learning algorithm discussed in this lab. We consider in this file the samples are the form $(\vec{X},\vec{Y})$ where $\vec{X}$ is considered the input vector for the algorithm (in our case the coordinates of the point) and $\vec{Y}$ is considered the desired output vector for the algorithm (in our case the group of the example belongs to).
The *k*-Medoids algorithm is considered an unsupervised learning algorithm because it will take in consideration from the training file only the input data $\vec{X}$ not the categories in which the input data is. The *k*-Medoids algorithm is a classical partitioning technique of clustering that splits the data set of *n* objects into *k* clusters, where the number *k* of clusters needs to be specified a priori (which implies that the programmer must specify *k* before the execution of a *k*-Medoids algorithm). The medoid of a cluster is defined as the sample in the cluster whose average dissimilarity to all the samples in the cluster is minimal, that is, it is a most centrally located point in the cluster.
For the application that need to be performed I recommend randomly choosing the value for number of clusters (value of *k*), a randomly value between 2 and 10.

Steps of the *k*-Medoids algorithm

1. Each medoid must be initialized in the working space of the input vector. Therefore, I recommend that for each medoid we randomly select a point from the "point file" and initialize its coordinates with the coordinates of the point. Thus, the medoid will be in the same representation space as the training sample. At this stage we randomly set the starting position for the learning algorithm. Please analyze, after finishing the implementation, if the initial position of the medoids have influences regarding the quality of the learning or not. For drawing on the screen the result of the learning algorithm I recommend to assigning for each medoid a different color and draw on the screen as a circle with radius grater that of the point. The color of the medoids have no connections with the output vector $\vec{Y}$ from the points file.
2. Take one point at a time from the "point file" and compute the similarity between it and all the medoids initialized in step 1. The point will be assigned to the medoid to which it is most similar. In other words, the point will be grouped to the closest medoid. There are several methods of computing the similarity. In the next section two such methods are presented. You can also suggest and used other methods.
3. Repeat the step 3 for all the points from the “points file” and for each point we will keep the information related to the medoid (class or category) to which it was grouped.
4. After grouping all of points from the “points file” (after assigned a medoid to each point from the file), for each medoid separately will do:
   a. compute the center of gravity of all samples (points) grouped at a medoid (into same class). Thus, for each class will compute a new center of gravity. All points will be draw on the screen with the medoid color to which they were assigned (only for a good view of the algorithm operation).
   b. Find the point (from the points that was assigned to medoid) that is the nearest point of the center of gravity computed in the previous step and modify the medoid

coordinate (class coordinate) with the found point coordinate (move the medoid in the new point position – we can say that the medoid "learn" a better position).

5. After compute the position for all medoids we can evaluate the algorithm by computing the error (convergence) of the algorithm position. The convergence function, denoted by E, is calculated as the sum of all similarity from each sample in a cluster to its medoid. If the convergence function does not change, we can say that the algorithm is finished.

The equation of the convergence function is:

$$E = \sum_{i=1}^{k} \sum_{\vec{X} \in k_i} d(\vec{X}, \overrightarrow{k_i})$$

6. After all medoids will "learned", resume the algorithm from the step 2 as long as there are a difference between previous convergence function and the current convergence function.
7. The algorithm will stop when no difference occurs in the convergence function, and we will say that the algorithm reaches an equilibrium position.

Some suggestions:

1. In the literature we say that a "epoch" has passed when the learning algorithm goes through all the training samples once.
2. For a good view of the algorithm, recommend to draw on the screen, after each epoch, all the points (colored with the color of the medoid to which it was assigned) and the new position of each medoid.
3. The center of gravity of a class may be, for example, the arithmetic means of all elements that are contained in that class.

Methods for compute the similarity

In the mathematics there are a lot of methods for compute the similarity, each method is selected according to the applicability domain. Among of the most common methods are: computing the Euclidean distance between two vectors or the cosine angle between two vectors. Those methods compute the distance between two vectors and need to respect all the distance proprieties. I present in this laboratory two methods Euclidean distance and Manhattan distance.

1. Compute the similarity using the Euclidean distance:

$$d(\vec{X}, \vec{X}') = \sqrt{\sum_{i=0}^{n} (x_i - x'_i)^2},$$

where *n* represent the number of coordinates of the sample, x and x'represents elements of the input vectors (samples, points).

2. Compute the similarity using the Manhattan distance:

$$d(\vec{X}, \vec{X}') = \sum_{i=0}^{n} (|x_i - x'_i|)$$

Implement the *k*-Medoid algorithm to automatically group (create cluster) the training samples for the dataset generated in the first lab.