Feature-based Object Recognition Methods

In the last exercise, we represented objects as feature vectors $\mathbf{x} = (x_1, x_2, \dots, x_N)$. Each of this feature vectors can be represented as a point in N-dimensional space X^N.

As it was mentioned in the previous exercise, it is important that objects from one class have similar feature vectors, i.e. the distance between the feature vector $\mathbf{x}_O$ of object O and the feature vector $\mathbf{x}_Q$ of object Q is small. The goal is to assign a class to each of feature vector, the feature vectors of objects from one class should create clusters in the space X^N. It is illustrated in Fig. 1 that shows three clusters of points in a $2D$ space of features x_1 and x_2. Each cluster represents one class. The goal of the feature-based recognition is to correctly assign a class to the feature vectors.

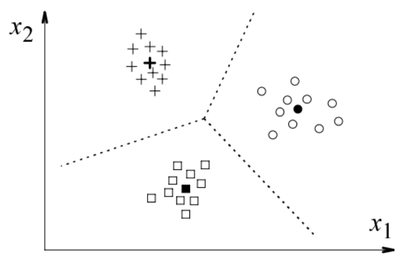

Figure 1: Illustration of objects in the feature space. The objects from the same class create clusters.

There exist many algorithms for clustering, they are called classifiers. Classifier is a function that assign a class to a feature vector. The algorithms can be divided into two classes - supervised and unsupervisied classifiers.

The supervised classifiers require a training data. These data contains feature vectors and the class of each vector. The goal of the algorithm is to find the parameters that divide the space of features vector such that the error of assigning the feature vectors from training data is minimal. Afterwards, it is possible to assign a class to a feature vector that is not in the training set according to the learned parameters. On the other hand, the unsupervised classifiers do not contain the training data and the clusters are found according to some common properties of the feature vectors (usually distance). The number of clusters should be defined in advance.

Computing Etalons

Today, we will introduce etalons that is a simple supervised classifier. It assumes that we have training feature vectors with the known class (label). Ethalon e_i is a vector that represents each class ω_i. The easiest way to compute the etalon is by computing the average value

$$\mathbf{e}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \mathbf{x}_{i,j}, \tag{1}$$

where N_i is number of training feature points of the class ω_i, $\mathbf{x}_{i,j}$ are values of the feature vector. In our case the ω_i consists of square, rectangle, and star. Values $x_{i,j}$ are the features. In Fig. 1, the etalons are denoted as the bold points. Let us assume that an unknown object (one that we obtain from our testing image) is represented using feature vector $\mathbf{x}$. Our classifier assigns a class to the unknown object based on the etalon that has the shortest distance to the point $\mathbf{x}$

$$\min_i = \left(\operatorname{dist}(\mathbf{e}_i, \mathbf{x})\right). \tag{2}$$

k-means

The k-means algorithm represents the branch of unsupervised clustering algorithms - we do not know the classes of the feature vectors. The clusters are found on the basis of distance between the points. The number of classes (clusters) is given a priori - it is the value of k. The algorithm consists of the following steps:

1. Initialize k centroids $(m_1, \dots, m_k)$ to randomly selected points from the input.

2. Compute Euclidean distance from each centroid to all input data points.
3. Assign each input data to the closest centroid.
4. Update position of the centroid by calculating the mean position of the assigned points.
5. Repeat from step 2 until centroids do not move very much (distance is less that give threshold, or no reassignment of data points).

Since the position of centroid are set randomly, it may happen that the algorithm does not always converge. So you have to run it once again to obtain a desired output. Also, you can experiment with the initial positions of centroids to see what happens.

Centroids provided by the k-means algorithm are then used in classification in the same way as were etalons. If the points are correctly assigned, the positions of etalon and centroid are identical. The only difference is that we do not know which class of object is represented by which centroid. Classification than works in the way that class names are simply numbers $(1, 2, \ldots, k)$. In addition, the numers may be different after next run of the algorithm.

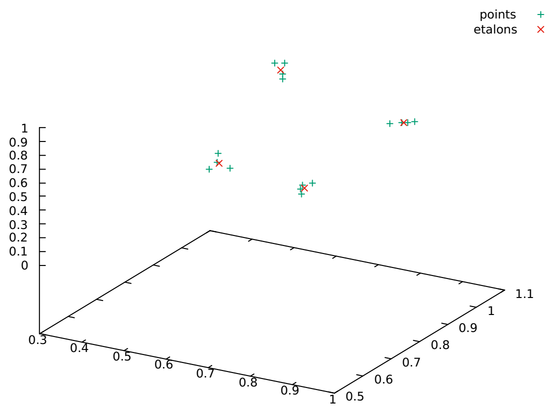

Figure 2: Illustration of features F1, F2 and F2 for four objects (green) and the etalons (red).

TASKS

Compute the etalons of classes from the training image. Afterwards, you will have 3 etalons. Open a testing image, compute features of the objects in this image and assign classes of them based on the distance to the etalons.

Do the same with k-means algorithm.