Fuzzy clustering

Let's suppose that we have a lot of object, and we've made some measurements of these objects. Can we now divide these objects into groups called **clusters**? And if so, how do we do this using fuzzy logic? That is what this chapter is about.

1 Types of clustering

1.1 The data set

Let's suppose that we have N objects (e.g. pieces of fruit). Of each of these objects, we make n **measurements** (e.g. size, weight, etcetera). These measurements are also called **features** or **attributes**. The set of measurements of one object, $\mathbf{z_k} = [z_{1k}, \ldots, z_{nk}]^T$, is called a **sample**, a **pattern** or simply an **object**. We can also put all measurements in a matrix. We then get the **data matrix** $\mathbf{Z} = [\mathbf{z_1} \ \ldots \ \mathbf{z_N}]$.

To divide objects into clusters, we often make use of **(dis)similarity measures**. One well-known example of a dissimilarity measure is the **Euclidian distance** $||\mathbf{z_j} - \mathbf{z_i}||$, but we'll consider more later. Based on the similarity measures, objects are divided into clusters. How exactly this can be done will be discussed later in this chapter.

1.2 Hard clustering

There is an important distinction between hard clustering and fuzzy clustering. In **hard clustering** we make a **hard partition** of the data set $\mathbf{Z}$. In other words, we divide them into $c \geq 2$ clusters (with c assumed known). With a partition, we mean that

$$\bigcup_{i=1}^{c} A_i = \mathbf{Z} \qquad \text{and} \qquad A_i \cap A_j = \emptyset \quad \text{for all } i \neq j. \tag{1.1}$$

Also, none of the sets A_i may be empty.

To indicate a partitioning, we make use of **membership functions** μ_{ik}. If $\mu_{ik} = 1$, then object i is in cluster k. Alternatively, if $\mu_{ik} = 0$, then object i is not in cluster k. Based on the membership functions, we can assemble the **partition matrix** $\mathbf{U}$, of which μ_{ik} are the elements. Finally, there is the rule that

$$\sum_{i=1}^{c} \mu_{ik} = 1. \tag{1.2}$$

In other words, every object is only part of one cluster. Thus, every column of $\mathbf{U}$ has only a single 1. The set of all hard clusterings $\mathbf{U}$ that can be obtained with hard clustering is now denoted as M_{hc}.

1.3 Fuzzy clustering

Hard clustering has a downside. When an object roughly falls between two clusters A_i and A_j, it has to be put into one of these clusters. Also, outliers have to be put in some cluster. This is undesirable. But it can be fixed by fuzzy clustering.

In **fuzzy clustering**, we make a **fuzzy partition** of the data. Now, the membership function μ_{ik} can be any value between 0 and 1. This means that an object $\mathbf{z_k}$ can be for 0.2 part in A_i and for 0.8 part in A_j. However, requirement (1.2) still applies. So, the sum of the membership functions still has to be 1. The set of all fuzzy partitions that can be formed in this way is denoted by M_{fc}.

Fuzzy partitioning again has a downside. When we have an **outlier** in the data (being an object that doesn't really belong to any cluster), we still have to assign it to clusters. That is, the sum of its membership functions still must equal one. In **possibilistic partitioning**, this requirement (1.2) is relaxed. Instead, it is only required that for every object we have $\mu_{ik} > 0$ for some cluster A_k. The set of all possibilistic partitions that can be formed in this way is denoted by M_{pc}.

2 The fuzzy c-means clustering method

2.1 The goal of the fuzzy c-means clustering method

Given a data set $\mathbf{Z}$, how do we find a good fuzzy clustering $\mathbf{U} \in M_{fc}$? For that, we have to use a clustering algorithm. One of the most-used algorithms is **fuzzy c-means clustering** which we will examine in this part. In the fuzzy c-means clustering method, we try to minimize the cost function called the **fuzzy c-means functional**, being

$$J(\mathbf{U}, \mathbf{V}|\mathbf{Z}) = \sum_{i=1}^{c} \sum_{k=1}^{N} (\mu_{ik})^m ||\mathbf{z_k} - \mathbf{v_i}||_{\mathbf{A}}^2. \tag{2.1}$$

In this equation, $\mathbf{V} = [\mathbf{v_1}, \ldots, \mathbf{v_c}]$ is a vector of **cluster prototypes** (centers) and m is a constant. Also, we have

$$D_{ik\mathbf{A}}^2 = ||\mathbf{z_k} - \mathbf{v_i}||_{\mathbf{A}}^2 = (\mathbf{z_k} - \mathbf{v_i})^T \mathbf{A} (\mathbf{z_k} - \mathbf{v_i}). \tag{2.2}$$

2.2 The fuzzy c-means clustering algorithm

So how do we minimize the cost function? Well, we start by taking a random partition matrix $\mathbf{U^{(0)}} \in M_{fc}$. We then continue doing the following steps.

1. We compute the weighted means of the clusters using

$$\mathbf{v_i}^{(l)} = \frac{\sum_{k=1}^{N} \left(\mu_{ik}^{(l-1)}\right)^m \mathbf{z_k}}{\sum_{k=1}^{N} \left(\mu_{ik}^{(l-1)}\right)^m}. \tag{2.3}$$

2. We compute the distances $D_{ik\mathbf{A}}^2$ using $D_{ik\mathbf{A}}^2 = ||\mathbf{z_k} - \mathbf{v_i}^{(l)}||_{\mathbf{A}}^2$.

3. We update the partition matrix $\mathbf{U}$. For all objects k, we define the new measurement functions $\mu_{ik}^{(l)}$ as

$$\mu_{ik}^{(l)} = \frac{1}{\sum_{j=1}^{c} \left(\frac{D_{ik\mathbf{A}}}{D_{jk\mathbf{A}}}\right)^{\frac{2}{m-1}}}. \tag{2.4}$$

   However, a problem occurs if $D_{ik\mathbf{A}}^2 = 0$. (This can occur if $\mathbf{z_k} = \mathbf{v_i^{(l)}}$ for some k, i or if $\mathbf{A}$ is a singular matrix.) Let's suppose that there are q clusters A_i for which $D_{ik\mathbf{A}}^2 = 0$. We then simply give all these clusters a membership degree of $\mu_{ik}^{(l)} = 1/q$. All the other clusters (with $D_{ik\mathbf{A}}^2 > 0$) get a membership function of $\mu_{ik}^{(l)} = 0$.

We repeat the above iteration until the partition matrix $\mathbf{U}$ doesn't really change anymore. That is, until $||\mathbf{U}^{(l)} - \mathbf{U}^{(l-1)}|| < \epsilon$ for some norm $||.||$ and for some defined ϵ. (Often $\epsilon = 0.01$ or $\epsilon = 0.001$ works well enough, depending on the trade-off between run-time and accuracy.)

2.3 Properties of the fuzzy c-means clustering method

There are several important things to know about fuzzy c-means clustering. First of all, it converges to a local minimum. (This depends on the initialization of $\mathbf{U}$.) So, to make sure that a good clustering is obtained, the algorithm needs to be run several times for different initializations $\mathbf{U}$.

It is also important to set the parameters of the algorithm right. The most important one is the number of clusters c. Sometimes, this is obvious. But often it is not. To test whether a clustering has the right number of clusters, you can look at a **validity measure** like the **Xie-Beni index**

$$\xi(\mathbf{U},\mathbf{V}|\mathbf{Z}) = \frac{\sum_{i=1}^{c}\sum_{k=1}^{N}(\mu_{ik})^m||\mathbf{z_k}-\mathbf{v_i}||^2}{c\cdot\min_{i\neq j}\left(||\mathbf{v_i}-\mathbf{v_j}||\right)}. \tag{2.5}$$

The upper side of the fraction can be seen as the 'average distance within the cluster', while the bottom side is an indication of the 'distance between clusters'. A small index is positive. So, if we simply run the algorithm for different numbers of clusters c, then we can select the solution with the smallest index.

Another important parameter is the **fuzziness parameter** m. If $m = 1$, then we wind up with a hard clustering. However, if $m \to \infty$, we wind up with a very fuzzy clustering $\mu_{ik} = \frac{1}{c}$ for all i, k. Usually, $m = 2$ offers a good compromise, though this number can be varied during subsequent runs of the algorithm.

The **norm-inducing matrix** mainly determines the shapes of the clusters. If $\mathbf{A} = \mathbf{I}$, then we are using a **Euclidian norm**. The shape of the clusters will be circular. Alternatives are the **diagonal norm** and the **Mahalanobis norm**, which respectively use

$$\mathbf{A} = \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma_2^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\sigma_n^2} \end{bmatrix} \quad \text{and} \quad \mathbf{A} = \mathbf{R}^{-1} = \left(\frac{1}{N}\sum_{k=1}^{N}(\mathbf{z_k}-\bar{\mathbf{z}})(\mathbf{z_k}-\bar{\mathbf{z}})^T\right)^{-1}. \tag{2.6}$$

Here, the parameters σ_i^2 are the variances of the matrix $\mathbf{Z}$ in direction i. Both the diagonal norm and the Mahalanobis norm will result in clusters with ellipsoidal shapes. However, the fundamental difference is that, with the diagonal norm, the axes of the ellipses are aligned with the main axes. When using the Mahalanobis norm, the axes are arbitrary.

2.4 An extension of the fuzzy c-means method

The downside with using a single matrix $\mathbf{A}$ is that all clusters will have the same shape and orientation. When there are clusters with different shapes, this will be undesirable. The **Gustafson-Kessel algorithm** is an extension of the fuzzy c-means method which gets rid of this downside.

The main idea is that, instead of using the same matrix $\mathbf{A}$ all the time, we use a different matrix $\mathbf{A_i}$ to calculate the norm $D^2_{ik\mathbf{A_i}}$. Now, also an optimum for the matrix $\mathbf{A_i}$ needs to be found. This does give rise to a problem though. If we minimize the cost function J right away, then the matrices $\mathbf{A_i}$ simply go to zero. We thus need to constrain them in some way. Usually, $\det(A_i) = |A_i| = \rho_i$ is used as a constraint. Here the choice of values for ρ_i depends on earlier experience. If this experience is lacking, $\rho_i = 1$ is simply chosen.

So how can this be implemented in the algorithm? Well, we simply replace step 2 by the following. We first calculate the **fuzzy covariance matrix $\mathbf{F_i}$** for all clusters i using

$$\mathbf{F_i} = \frac{\sum_{k=1}^{N}\left(\mu_{ik}^{(l-1)}\right)^m(\mathbf{z_k}-\mathbf{v_i})(\mathbf{z_k}-\mathbf{v_i})^T}{\sum_{k=1}^{N}\left(\mu_{ik}^{(l-1)}\right)^m}. \tag{2.7}$$

Now, the improved value of $\mathbf{A_i}$ can be found using

$$\mathbf{A_i} = (\rho_i \det(\mathbf{F_i}))^{\frac{1}{n}} \mathbf{F_i}^{-1}. \tag{2.8}$$

This matrix $\mathbf{A_i}$ is then used to compute the distance norm $D^2_{ik\mathbf{A_i}}$ after which the algorithm proceeds as normal.

The matrices $\mathbf{F_i}$ are quite important. In fact, the shapes of the clusters depend on them. The main axes of the ellipses are denoted by the eigenvectors $\phi_{\mathbf{ij}}$ of $\mathbf{F_i}$. The sizes of the ellipses in these directions are proportional to $\sqrt{\lambda_{ij}}$, with λ_{ij} the eigenvalue corresponding to the eigenvector $\phi_{\mathbf{ij}}$.