A new Fuzzy and Hard Clustering Software Toolkit

Mariana Soffer, M. Daniela López De Luise
marianasoffer@gmail.com
lopezdeluise@yahoo.com.ar

Abstract

This paper presents the FuzzyToolkit prototype which has a flexible software design for handling fuzzy and hard clustering algorithms. It provides modules that allow easy data input/output manipulation; multiple functionalities and clustering algorithms, which include calculation of the evaluation of the clustering performance; parameter selection via meta-heuristics; and 3D result visualizations. This toolkit is compared here against other open source clustering tools. A very short revision of some of its graphic options is included.

1. Introduction

Clustering methods [1], which belong to the unsupervised learning category, consist in dividing items from a dataset into a certain amount of groups; this amount can be determined a priori or not depending on the type of clustering algorithm and other factors. These groups could overlap or not. The methods where groups do not overlap are obtained using hard clustering algorithms [4]; in this case each instance belongs to only one of the groups. The methods where groups are able to overlap are generally referred to as fuzzy clustering algorithms; in this less constrained case [2] [3] items do not belong exclusively to one cluster, they are associated to a value which indicates the strength of belonging to each one.

This paper presents a functional prototype developed in Java programming language, called FuzzyToolkit. It is not part of the scope of this paper to analyze or justify every clustering algorithm, but to present an efficient tool implemented in order to concentrate most of them. This working prototype implements several well known fuzzy and non-fuzzy (also called hard) clustering algorithms. The following sections are organized as follows: section 2 describes the main functionality and structure of the Toolkit; section 3 presents a case study showing its usage, and a comparison of the FuzzyToolkit with other open source clustering software (xPloRe [9] and Weka [10]). Results are presented in section 4. Section 5 includes conclusions and future development plans for the toolkit.

2. FuzzyToolkit design

The overall architecture of the FuzzyToolkit was designed to be very flexible. Object Oriented Design patterns such as Observer, Singleton, Strategy, Factory, Delegator, etc were implemented. The pattern implementation objective is to allow adding new clustering algorithms, distance metrics and visualization procedures as natural extensions of the actual system.

The main components of the FuzzyToolkit architecture are:
- Distance function
- Clustering algorithm
- Meta-learning algorithm
- Dimensionality reduction method
- Visualization strategy
- Support function

Each one has been implemented as a class or interface hierarchy with a generic to specific fashion approach.

2.1. Architecture

The prototype implements a specific hierarchy from each of the main components. There is also a set of standard algorithms codified as extensions of the generic modules. Any specific alternate algorithm should be implemented as an extended class base from an original component.

In the next section, the main hierarchies will be described.
2.1.1. Distance function

This class defines the specific metrics that could be used to evaluate dot distances. Each dot represents a determined instance. This interface is inherited by two main abstract classes whose respective goals are:

- A generic function to calculate real number distances.
- A generic function to calculate nominal distances. Each one is further expanded by implementing a specific distance calculation according to the data type.

Dot distances such as Manhattan, Euclidean and Tchebyschev [4] are already implemented in the FuzzyToolkit. All of them are an extension of the generalized real distance. In the case of nominal distances, binary and correlation-based distances are already implemented as extensions of the generalized nominal distance.

Table 1

<table>
<thead>
<tr>
<th>FuzzyToolkit Structure</th>
</tr>
</thead>
</table>

2.1.2. Clustering algorithm

This class covers several approaches for clustering methods. It has three extended basic classes:

- Fuzzy clustering class for items composed by fields with real values.
- Hard clustering class for items composed by fields with real values.
- Hard clustering class for items composed by fields with nominal values.

Several specific algorithms are based on each of the previous approaches. Those algorithms where implemented according to the different base class they derive from.

- Among the implementation of the general Fuzzy clustering algorithms are CMeans [3], FuzzyKMeans [6] and GKCmeans [3].
- Among the concrete implementations of hard clustering algorithms are KMeans [4], FarthestFirst [4] and KMedoids [4].
- Some of the implemented nominal hard clustering algorithms are KMeans and KMedoids.

2.1.3. Meta-learning

The Meta-learning looks for finding out the best parameter for an algorithm upon certain restrictions. Therefore it needs a clustering method. It also requires the specification of the domain value of the parameter to be optimized, in some cases the step increment value should also be specified; with this information it executes the child algorithm several times, modifying the parameter value each time it executes it. In order to end the search for the optimum parameter the meta algorithm has several stopping conditions that could be selected, like a maximum number of iterations. Finally it finds the best solution and returns it to the caller. To do this it uses an objective function either to be minimized or maximized according to the specified parameter.

This meta-learning was expanded into the following more specific classes:

- A general meta-learning approach for hard clustering with nominal values algorithms.
- A general meta-learning generic algorithm for fuzzy clustering with real values algorithm.

It is important to note that using this algorithmic there is no guarantee of an optimal result. In general, the related parameters require a careful research performed either manually or with the meta-heuristic functionality.

2.1.4. Dimension reduction approach

The dimensionality for a model takes a big impact into the solution itself. But it could be also a problem when the number of dimensions becomes difficult to manage.

Henceforth, certain strategies must be used to reduce the dimension of the problem without losing the model accuracy.

The toolkit has the ability to reduce the dataset dimensionality using different methodologies, such as PCA and Gain Ratio. This is useful for high dimensionality datasets, due to the fact that these methodologies can transform the data in to a chosen number of dimensions or factors, loosing the minimum
amount of information in the process. This is also good for visualization of the results when reducing the large quantity of variables to either 2 or 3 factors.

There are two classes that extend the general dimension reduction class:
- Dimensionality reduction for graphical interface.
- Dimensionality reduction for clustering processing.

The first one was designed just for better visibility of the solution in the graphics that show the results, whereas the second is intended to reduce the amount of information handled to the clustering algorithms. Among the algorithms implemented for this later case are PCA [7], RedRelief [5] and Gain Ratio [5].

2.1.5. Visualization strategy

It covers all the classes that are related to the graphical interface involved in showing clustering results. The main classes defined in this category are:
- Algorithm to graph results in 2D.
- Algorithm to graph results in 2D showing also cluster radius as circles/ellipses.
- Algorithm to graph results in 3D.
- Algorithm to graph results in 3D with circular or ellipsoidal cluster limits depicted.

2.1.6. Support objects

In this part of the hierarchy there are several methods that implement minor functions of the data manipulation. Each one is a kind of interface between items and implemented algorithms. Here is a short list of them:
- Data normalization, useful to balance dimensions when needed.
- Result calculation upon applying certain clustering algorithm. It could be a cluster number (in hard strategies) or a belonging one (in fuzzy approaches). There is also other information that is part of the result: cluster’s maximum and average radius, SSE [4], number of clusters and other measures to help evaluate the goodness of the calculation.
- Data uploading from a CSV file.
- Results saving into a CSV file.
- Previous results along with its corresponding dataset retrieving from a CSV file.
- Learned clustering model saving to a file.
- Saved clustering model retrieving from file.
- New instance classification using a retrieved model.

- Confusion matrix calculation for comparison between both a saved clustering result and a new calculated clustering result.

3. Case Study

3.1. Data set Specification

The Iris Plants Database was created by R.A. Fisher and donated by Michael Marshall on the year 1988. It is perhaps the best-known database to be found in the pattern recognition literature. Fisher's paper [8] is a classic in the field and it is referenced frequently. The data set contains 150 instances of the iris plant. Each instance could belong to one of the 3 existing classes (Iris-Setosa, Iris-Versicolor and Iris-Virginica), in this particular dataset each class contains exactly 50 plans.

Every instance has four attributes:
- Sepal length ranging from 43 to 7.9 cm.
- Sepal width ranging from 2 to 4.4 cm.
- Petal length ranging from 1 to 6.9 cm.
- Petal width ranging from 0.1 to 2.5 cm.

3.2 Experiment description

The KMeans algorithm will be tested under the FuzzyToolkit and two open source programs in order to compare their efficiency and accuracy.

For this experiment the Euclidean distance is used, since all of the software packages are able to compute with it. The same parameter values for all the test software are assigned with equal numbers. Those parameters that do not exist in the rest of the tools are adjusted to provide the most similar possible outcome.

The resulting confusion matrices of each algorithm along with the percent of incorrectly classified instances are compared for the evaluation of the tests. To do that, the hand-made classification of the instance is taken. The resulting cluster-to-species association could be easily obtained by performing a simple exploratory analysis.

As part of the test, the meta-algorithm is used to search the best number of cluster for this dataset. This parameter (ranging between 2 and 10), is the result of the minimization of the resulting SSE value. It will be shown in the next section that 3 is the optimal number of clusters chosen by FuzzyToolkit meta-algorithm, which is the real number of species or classes in the dataset (Iris-Setosa, Iris-Versicolor and Iris-Virginica).

All the tests are compared with the WEKA software, an open source program developed by the Waikato University of New Zealand. Its distribution already comes with the iris dataset and the correctly
assigned class that each instance belongs to. The other software to be used is XploRe.

Whereas WEKA can only display results in 2D, the FuzzyToolkit graphic module can be used to display the solution in 2D and 3D (like other frameworks such as the SPSS one). The GUI can also rotate the graph, adjust the size and customize the graphic type among other functionalities in a similar fashion to SPSS. In WEKA there are an important number of features to allow the visualization of the graph in several ways. Finally, unlike the prototype, the type of data requires an extra module in xPloRe in order to be able to show the resulting clustering visualization graph.

4. Results

The dataset was processed for xPloRe. It was required a pre-processing of the iris dataset that comes with the standard distribution of WEKA in order to make it understandable for the xPloRe software. A small script was developed to load the data, execute the KMeans algorithm and extract the solution. Since this software does not have the ability to visualize clustering results graphically, it will not be possible to show it here either. The associated confusion matrix is depicted in table 2.

Table 2
Confusion Matrix for xPloRe with KMeans

<table>
<thead>
<tr>
<th>Clusters</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>Iris Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>50</td>
<td>0</td>
<td>0</td>
<td>Virginica</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>40</td>
<td>15</td>
<td>Versicolor</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>4</td>
<td>40</td>
<td>Setosa</td>
</tr>
</tbody>
</table>

Incorrectly clustered instances: 13.33%

The same processing was repeated with WEKA. The corresponding confusion matrix is showed in table 3; graphical display of the tool, with the cluster assignments is included in table 4.

Table 3
Confusion Matrix for WEKA with KMeans

<table>
<thead>
<tr>
<th>Clusters</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>Iris Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>Virginica</td>
</tr>
<tr>
<td>1</td>
<td>47</td>
<td>0</td>
<td>3</td>
<td>Versicolor</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>0</td>
<td>36</td>
<td>Setosa</td>
</tr>
</tbody>
</table>

Incorrectly clustered instances: 11.33%

Table 4
Clustering visualization results in WEKA

Table 5
Confusion Matrix for FuzzyToolkit with KMeans

<table>
<thead>
<tr>
<th>Clusters</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>Iris Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>50</td>
<td>0</td>
<td>Virginica</td>
</tr>
<tr>
<td>1</td>
<td>46</td>
<td>4</td>
<td>0</td>
<td>Versicolor</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>1</td>
<td>37</td>
<td>Setosa</td>
</tr>
</tbody>
</table>

Incorrectly clustered instances: 11.33%

The meta-heuristic module was tested first in order to demonstrate that it could select the best number of clusters for this dataset. The selection criterion of clusters was the minimization of the SSE (Sum of Square Errors). The parameter to be tuned was the number of clusters. The domain specified for it ranged between 2 to 10 clusters. The result indicated that the best number of groups was 3.

The next test performed was KMeans in order to compare results with the same data in 3 clusters, sticking to the Euclidean distance, setting the epsilon parameter to 0.01 and the maximum number of iterations to 999. Results are in table 5.

As can be seen from the previous figures, all software package outputs include the cluster assignment of each instance and the centroid of each cluster. WEKA also yields the confusion matrix of the clustered dataset when the real results are provided, including the percentage of misclassified instances and the percentage of instances that belong to each cluster.

FuzzyToolkit provides the previous mentioned information and some other evaluation measures on top of them. They are used to know how good the results are and also the descriptive information about each cluster. These measures include (among others) the hard clustering method ones:
- SSE: sum of the square error from the items of each cluster.
- Inter cluster distance: sum of the square distance between each cluster centroid.
- Intra cluster distance for each cluster: sum of the square distance from the items of each cluster to its centroid.
- Maximum Radius: largest distance from an instance to its cluster centroid.
- Average Radius: sum of the largest distance from an instance to its cluster centroid divided by the number of clusters.

The FuzzyToolkit can present a large variety of graphics. The system applies the PCA dimensionality reduction algorithm, in order to show the same data in two alternative 3D visualizations that are in table 6 and table 7 respectively.

### Table 6
Visualization I: FuzzyToolkit clustering results

![Visualization I](image1)

### Table 7
Visualization II: FuzzyToolkit clustering results

![Visualization II](image2)

## 5. Conclusions

As it can be seen from the confusion matrixes that resulted from each software calculation, the accuracy of the FuzzyToolkit is better than the one from xPlorE and similar to the one from WEKA. This higher accuracy is provided in part by the possibility of adjusting more parameters, such as epsilon (that indicates the minimum variance of the objective function in order not to stop the algorithm) and the maximum amount of iterations allowed. These parameters can be used either one at a time or both of them together.

The meta algorithm method can provide a good approximation of the required parameter to be chosen. It is useful in cases where the exact amount of clusters is unknown and is not obvious either.

The FuzzyToolkit is a more flexible tool allowing to perform cluster selection automatically based on desired parameters; enabling to tune more parameters than the other programs; giving more evaluation measures that enable to better judge cluster results; letting select a dimensionality reduction algorithm to graph the clustering assignment results; and finally providing much better visualizations of the obtained results.

Other advantages not shown in this paper are the number of options this software provides which can be combined as desired, such as:
- 20 different clustering algorithms.
- 10 different distance functions.
- 5 different dimensionality reduction algorithms.
- 4 kinds of elaborate and interactive visualizations.
- Retrieve different kinds of datasets.
• Store the data along with the clustering results.
• Compare different algorithms.
• Store and retrieve clustering models.
• Easily extend the toolkit.

The prototype clustering methods yielded results as good as the ones provided by the open source software packages, and in some cases better than them. This happens for each algorithm tested.

In the future we’ll test if fuzzy algorithms can outperform its analogous non-fuzzy ones; implement fuzzy clustering with nominal data; implement new algorithms; and finally develop clustering algorithms in order to be able to evaluate datasets containing instances with nominal and real numbers.

6. References