M-Estimator induced Fuzzy Clustering Algorithms

Roland Winkler[1] **Frank Klawonn**[2] **Rudolf Kruse**[3]

[1]German Aerospace Center (DLR) Braunschweig, roland.winkler@dlr.de
[2]Ostfalia University of Applied Sciences Wolfenbüttel, f.klawonn@ostfalia.de
[3]Otto-von-Guericke University Magdeburg, kruse@iws.cs.uni-magdeburg.de

Abstract

M-estimators can be seen as a special case of robust clustering algorithms. In this paper, we present the reversed direction and show that clustering algorithms can be constructed by using M-estimators. A clever normalization is used to link the values of several M-estimator prototypes together in one clustering algorithm. A variety of M-estimators and several normalization strategies are used in 4 data sets to present their differences and properties. The results are evaluated using 5 different clustering validation indices.

Keywords: Fuzzy c-means, M-estimators, Robust statistics, Noise clustering, Multiple prototypes

1. Introduction

Fuzzy clustering [1, 2] and M-estimators [3, 4] of location have roughly the same goal. For both, it is assumed that the data is somehow contaminated with noise and/or data objects can not be uniquely assigned. The difference however is, for fuzzy clustering, it is assumed that a set of different data generating processes are responsible for the observed data set. M-estimators of location just assume one, (contaminated) data generating process. The way from fuzzy clustering to M-estimators is possible, because fuzzy clustering algorithms contain the case of just one data generating process (one prototype) as a special case. The other direction however, to extend M-estimators to handle more than one prototype, is much more difficult. The start position is given by the update mechanisms of both algorithm classes since the update process of fuzzy clustering algorithms is very similar to M-estimators. For usual fuzzy clustering algorithms, this update process is derived from the definition of an objective function that is optimized by gradient descent. This gradient descent method can be altered to form a new clustering algorithm that must not necessarily be derived from an objective function. The usual process of fuzzy clustering algorithms applies weights on data objects. These weights are defined by characteristic functions for each fuzzy clustering algorithm. M-estimators also contain weights for data objects. In this paper, we demonstrate that a clever normalisation process can use these weights to form a clustering algorithm update process, that is similar to a fuzzy clustering algorithm and supports multiple prototypes.

1.1. Related Work

The famous noise clustering [5] algorithm is a very effective extension to Fuzzy c-Means (FCM) [6, 7]as it provides robustness properties to FCM. Davé and Krishnapuram present an M-estimator [8, 4] that behaves like noise clustering in combination with FCM if only a single “normal” cluster is applied. This connection is again emphasized in [9]. Davé and Sen introduced in [10] a noise clustering algorithm that uses a unique noise distance for each data object. This fuzzy clustering algorithm is very close to possibilistic clustering and also closer to M-Estimators than the original noise clustering algorithm. Keller has introduced a modified FCM algorithm in [11] that introduces a weight for each data object. The weight serves as a parameter that generates similar to the fuzzifier the importance of a data object. Noise data objects are identified by a low weight which increases the robustness of FCM with respect to outliers. A different robust FCM algorithm is introduced by Kim, Krishnapuram and Davé [12] which uses a least trimmed squares measure instead of the usual least squares approach for FCM. In their approach, they reformulate FCM to gain a representation based on a harmonic mean value that is trimmed to eliminate the influence of outliers.

Noise clustering is a good way to reduce the influence of noise or corrupted data on the ”good” cluster. Klawonn and Höppner introduced an alternative approach using a polynomial fuzzifier function instead of the usual exponential fuzzifier [13, 14] which effectively reduces the influence of noise points, no matter how far they are away from the “good” clusters. Their approach can be seen as a convex combination of hard c-means [15] and FCM with a fuzzifier of 2 and leads to fuzzy membership values as well as strict values of 1 and 0. More information about this concept is provided in [16, 14, 17]. A similar approach is discussed by J. Leski in [18]. His approach is to robustify FCM by applying an ϵ-norm where every distance below ϵ is set to 0. This norm was first introduced by Vapnik [19] to create an ϵ-insensitive estimator.

The basis for the robust statistics is discussed in [3, 4]. Especially the introduction of the influence function by Hampel in [8] is very helpful. Further interesting work on robustness is done by Hennig, for example in [20, 21, 22]. He deals with measuring the robustness not only of the used method but also as a property of the cluster in question. His method is rather algorithm independent and comparable for several algorithms if the algorithm's result is a hard partitioning of the data. Also Choi and Krishnapuram have published a paper in [23] that links clustering and M-estimators. A more application oriented paper was published by Lozeron and Victoria-Feser that is related to M-estimators in [24].

An interesting alternative to noise clustering is presented by Frigui and Krisnapuram in [25] by using the robust prototype clustering (RPC) algorithm to handle noise data objects. RPC basically works the same way as for M-estimators: a robust loss function is wrapped around the distance measure in an FCM like objective function. However, the choice of this loss function is very limited since only functions can be used that allow for an closed-form solution for the prototype based on the derivative. Frigui and Krisnapuram also introduced an algorithm called (robust) competitive agglomeration clustering [26, 27]((R)CAFCM) which is able to identify noise data objects as well as other parameters of FCM like the number of clusters in a data set. The robust version of CAFCM again has a wrapped loss function around the distance value the same way it is done for RPC.

1.2. Overview

In the next section, the basics regarding fuzzy c-means clustering and M-estimators are introduced. Also the 4 in the experiments in Section 4 used M-estimators are presented. In Section 3, a fusion of the M-estimator robustness concept is extended to a multi-prototype clustering algorithm. Finally, the conclusions in Section 5 are followed by the used references.

2. Fuzzy c-Means, NC and M-Estimators

2.1. Fuzzy c-Means

Although fuzzy c-means (FCM) and noise clustering (NC) are very well known, some mathematical details are needed in the next section. Let $\boldsymbol{X} \subset V$ be a finite set of data objects of a vector space V with $|\boldsymbol{X}| = n$. The clusters are represented by a set of prototypes $\boldsymbol{Y} = \{y_1, \dots, y_c\} \subset V$ which can be initialized randomly, only the number of prototypes c must be known in advance. Let $1 < \omega \in \mathbb{R}$ be the fuzzifier and $\boldsymbol{U} \in \mathbb{R}^{c \times n}$ be the partition matrix with $u_{ij} \in [0,1]$ and $\forall j : \sum_{i=1}^{c} u_{ij} = 1$. And finally, let $d : V \times V \longrightarrow \mathbb{R}$ be a distance function with its abbreviation $d_{ij} = d(y_i, x_j)$ and f_{FCM} be the fuzzifier function $f_{\text{FCM}}(u) = u^{\omega}$.

Fuzzy c-means clustering is based on an objective function $\boldsymbol{J}$ that is to be minimized:

$$\boldsymbol{J}_{\text{FCM}} = \sum_{i=1}^{c} \sum_{j=1}^{n} f_{\text{FCM}}(u_{ij}) d_{ij}^2 \tag{1}$$

The minimization of J is done by iteratively updating the members of $\boldsymbol{U}$ and $\boldsymbol{Y}$ and is computed using a Lagrange extension to ensure the constraints $\sum_{i=1}^{c} u_{ij} = 1$. The iteration steps are denoted by a time variable $t \in \mathbb{N}$ denoting $t = 0$ as the initialization step:

$$\begin{aligned} u_{ij}^{t+1} &\overset{\text{FCM}}{=} \frac{(d_{ij}^t)^{\frac{2}{1-\omega}}}{\sum\limits_{k=1}^{c} \left((d_{kj}^t)^{\frac{2}{1-\omega}} \right)} \text{ and} \\ y_i^{t+1} &\overset{\text{FCM}}{=} \frac{\sum\limits_{j=1}^{n} f_{\text{FCM}}\left(u_{ij}^t\right) \cdot x_j}{\sum\limits_{j=1}^{n} f_{\text{FCM}}\left(u_{ij}^t\right)} \end{aligned} \tag{2}$$

For noise clustering, an additional cluster is specified which is represented by a virtual prototype y_0 which has no location in V. Instead, it has a constant distance $0 < d_{\text{noise}} \in \mathbb{R}$ to all data objects: $\forall j : d_{0j} = d(y_0, x_j) = d_{\text{noise}}$ which is called noise distance. y_0 is not represented as a member of V, which means, it is not updated during the iteration process. However, the membership values are updated as if y_0 was a normal prototype. The noise prototype is introduced to assign higher membership degrees to the noise cluster for all data objects whose distance to regular prototypes exceeds the noise distance. This favours regular prototypes to be better placed in the center of data clusters without being attracted by noise data. Also it suppresses the unwanted effect that membership values converge to $\frac{1}{c}$ for data objects far away from all normal prototypes.

2.2. M-estimators

M-estimators (M-E) are based on the idea of maximum likelihood estimation (MLE). MLE [28] is a method to find the parameters of a statistical model in such a way that it becomes maximal likely to observe the given data set. This proposition already contains the greatest weakness of this method: the data generating process must be known in order to choose the correct model. If the model is chosen (even slightly wrong), no parameter optimization can generate a meaningful result. Often, the correct model is not known or the data set contains outliers that make it impossible to estimate the correct parameters even if the underlying model might be correct. M-estimators are designed in such a way that they still produce meaningful results, even if the model assumption is not correct or the data contains outliers. MLE is usually performed using the expectation maximization (EM) algorithm [29].

The EM algorithm is based on an objective function as FCM but with a statistical background. As for FCM, let there be c clusters and $\boldsymbol{X}$ be the data set. Consider the random variable $\tilde{X} : \Omega \to V$ that has the data space as domain and let $f_{\tilde{X}} : \boldsymbol{X} \times \boldsymbol{\Theta} \to \mathbb{R}$ be its probability density function that depends on some parameter set $\Theta \in \boldsymbol{\Theta}$. The underlying model assumption is done by choosing the kind of density function $f_{\tilde{X}}$. The likelihood of observing the data set is specified by the likelihood function

$$\boldsymbol{J}_{\text{MLE}} = \prod_{j=1}^{n} f_{\tilde{X}}(x_j, \Theta),$$

which plays the role of the objective function for FCM but in this case, must be maximized. For computational properties, the negative log-likelihood of the data set is optimized instead of the likelihood.

$$-\ln \boldsymbol{J}_{\text{MLE}} = \sum_{j=1}^{n} -\ln f_{\tilde{X}}(x_j, \Theta),$$

which in turn must be minimized. The new minimum is located at the same place as the maximum of the likelihood function because the negative logarithm is a strictly decreasing function. Normal statistical functions like normal distributions or even mixtures of normal distributions are very often used. Single outliers can cause significant damage to the parameter estimation, not only for normal distributions. Therefore, in robust statistics the model is replaced by a robust loss function ρ that is not necessarily based on a probability density function $\rho(x_j, \Theta) = -\ln f_{\tilde{X}}(x_j, \Theta)$. Therefore, $\Theta = y$ is interpreted as prototype, $y \in Y$ and the function $g(x, y) = \rho(\|x - y\|)$ is interpreted as a loss function $\rho : \mathbb{R} \to \mathbb{R}$ that takes the distance $\|x - y\|$ as an argument with $\|\cdot\|$ as the Euclidean distance. Finally, the M-estimator (maximum-likelihood-type estimator) is an optimization problem that is based on minimizing the following objective function:

$$\boldsymbol{J}_{\text{M-E}} = \sum_{j=1}^{n} \rho(d_j),$$

with $d_j = \|x_j - y\|$. For a local minimum, a necessary condition is, that the derivative is zero. Usually, it is not possible to calculate a global minimum for this objective function directly. Therefore, similar to FCM, an iterative update process is used. Let $\psi(d_j) = \psi(\|x_j - y\|) = \frac{\partial}{\partial y}\rho(\|x_j - y\|)$ be the derivative of ρ and $w_j = w(d_j) = \frac{\psi(d_j)}{d_j}$. The optimization criterion is:

$$0 = \sum_{j=1}^{n} \psi(d_j) \cdot \frac{\partial}{\partial y} d_j = 2\sum_{j=1}^{n} w_j \cdot (x_j - y). \tag{3}$$

This holds, because d_j has been defined as the Euclidean distance: $d_j \cdot \frac{\partial}{\partial y} d_j = \|x_j - y\| \cdot \frac{\partial}{\partial y}\|x_j - y\| = \frac{1}{2}\frac{\partial}{\partial y}\|x_j - y\|^2 = (y - x_j)$. From Equation (3) follows by rearranging:

$$y = \frac{\sum_{j=1}^{n} w_j x_j}{\sum_{j=1}^{n} w_j}. \tag{4}$$

Thus, y is the weighted mean of the data set where the weights in turn depend on y in relation to the data objects. This provides an iterative update process for M-estimators. The equation looks very much like the update equation for FCM in (2), if the term w_j is replaced by $f_{\text{FCM}}(u_{ij})$. Or more generaly,

$$y^{t+1} = \frac{\sum_{j=1}^{n} f(u_j^t) x_j}{\sum_{j=1}^{n} f(u_j^t)}. \tag{5}$$

with $f(u)$ is either a fuzzifier function $f_{\text{FCM}}(u)$ or a robust weight function w of an M-estimator.

Later, some M-estimators are applied into the new, multi-prototype algorithm. Table 1 the M-estimators that are used this paper are listed.

Table 1: Several M-Estimators

Function and its Parameters	Weight function $w(d)$	Parameter Range
Huber $d_{\text{noise}} > 0$	$\begin{cases} 1 \\ \frac{d_{\text{noise}}}{d} \end{cases}$	$0 \leq d < d_{\text{noise}}$ $d \geq d_{\text{noise}}$
Hampel $0 < a \leq b \leq c$ $a_1 = ab - \frac{1}{2}a^2$ $a_2 = \frac{a}{2}(c - b)$	$\begin{cases} 1 \\ \frac{a}{d} \\ \frac{a}{d}\left(\frac{c-d}{c-b}\right) \\ 0 \end{cases}$	$0 \leq d \leq a$ $a < d \leq b$ $b < d \leq c$ $c < d$
Cauchy $d_{\text{noise}} > 0$	$\left(1 + \frac{d^2}{d_{\text{noise}}^2}\right)^{-1}$	$0 \leq d$
Tukey $d_{\text{noise}} > 0$	$\begin{cases} \left(1 - \frac{d^2}{d_{\text{noise}}^2}\right)^2 \\ 0 \end{cases}$	$0 \leq d < d_{\text{noise}}$ $d_{\text{noise}} \leq d$

3. Fuzzy clustering approaches induces by M-estimators

The extension of M-estimators of location to the multi-prototype case, like in fuzzy clustering is not a trivial task. Frigui and Krishnapuram made a nice proposal in [25] by introducing RPC, a clustering algorithm with robust loss functions. However, the use of loss functions, wrapped around the distance is quite limited because the derivative of the objective function must be solvable for the prototypes positions y_i. Therefore, the objective function

$$\boldsymbol{J}_{\text{RPC}} = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^{\omega} \rho(d_{ij}^2)$$

can not be used with most M-estimators.

A different attempt how an M-estimator can be extended to multiple prototypes without loosing the characteristics of the original M-estimator is presented here. This attempt should not suffer the same problems as the RPC approach and is therefore not based on an objective function. The idea is basically to use the robust weights of prototypes independently as fuzzy membership values. To avoid a clustering algorithm with independent prototypes, the membership values are normalized to create a clustering algorithm, similar to FCM.

Let f be the fuzzifier function for a fuzzy clustering algorithm as in Equation (5). In a fuzzy algorithm, the sum of membership values is equal to 1. However, the membership values are implicitly determined, the update process just requires the fuzzified membership values (see Equation (5)). This fact is important since a membership value is usually not specified for M-estimators: $w_j = w(d_j) = f(u_j)$ for a data object x_j in a fuzzy clustering algorithm. The membership value u_j can also be considered to be a function of d_j, $u_j = u(d_j)$ while the function u depends on the fuzzy clustering algorithm: $f(u(d_j)) = w(d_j)$. The problem is, that there is no distinct equation for u_j, if considering an M-estimator. Therefore, the condition of $\sum_j u_j = 1$ is hard to specify.

If going from one prototype to many prototypes considering, $f(u(d_{ij})) = w(d_{ij})$, the result is a clustering algorithm with c independent prototypes. This is not very convenient since it makes the clustering result extremely dependent on the initialization, much like PCM. Therefore, an interdependence among prototypes is necessary to create a reasonable well working clustering algorithm. One way of creating an interdependence is, to normalise the membership values. Let d_{ij} be the distance from prototype y_i to data object x_j and $w_{ij} = w(d_{ij})$ be a M-estimator weight function. Let the normalization $g : [0,1] \to [0,1]$ be a monotonously increasing function with $g(0) = 0$. Than the following equation describes the update mechanism of an M-estimator induced fuzzy clustering algorithm:

$$u_{ij}^{t+1} = g(w_{ij}) = g(w(d_{ij})) \tag{6}$$

with w_{ij} is an M-estimator weight function as in table 1. In the remaining part of this section, several normalisation methods are discussed for this scenario.

The first normalisation methods are very straight forward.

$$g_1(w_{ij}) = \frac{w_{ij}}{\sum_{k=1}^{c} w_{kj}} \text{ and} \tag{7}$$

$$g_2(w_{ij}) = \frac{w_{ij}}{\sum_{k=1}^{c} w_{kj}} \cdot w_{ij}. \tag{8}$$

g_1 maps the weights on the $[0,1]$ interval in such a way, that the sum of all normalised weights is 1. This is close to FCM, but it does not fit for the robust weight calculation because for some algorithms, the weights can become zero. If all weights but one are zero, the normalisation of a non-zero weight is set to 1, which is not a very robust feature. This problem is present for Hampel's and Tukeys M-estimator since both can produce 0 weights. g_2 overcome the problem of g_1 by multiplying the original weight on the normalised value of g_1. This takes the actual robustness of the weight into account.

In FCM as well as in PFCM, close prototypes are pulled apart from the surrounding data objects. This comes from the harmonic mean in FCM (and similar in PFCM) of the form

$$u_{ij} = \frac{\frac{1}{d_{ij}^2}}{\sum_{k=1}^{n} \frac{1}{d_{kj}^2}}.$$

which gives data objects close to an other prototype a membership value of almost 0. As this kind of mean is missing in the two presented normalisation methods, close prototypes do not have the tendency to move apart. Therefore, the third normalisation maps the weights from $[0,1]$ to $[0,\infty)$ in order to get a more sophisticated mean.

$$g_3'(w_{ij}) = \frac{\frac{1}{1-w_{ij}}}{\sum_{k=1}^{c} \frac{1}{1-w_{kj}}} \cdot w_{ij}. \tag{9}$$

g_3' reduces the problem, that two prototypes run into the same location. However, there are situations where this happens. If Huber's or Hampel's M-estimator is used, areas of weight 1 appear near their prototype. In this case, it is not clear how the data objects are partitioned. The situation can be solved by giving all membership values the same normalised weight. Let $\hat{c} = |\{w_{kj} : w_{kj} = 1, k = 1 \ldots c\}|$ be the number of weights of value 1.

$$g_3(w_{ij}) = \begin{cases} g_3'(w_{ij}) & \text{if } \hat{c} = 0 \\ 0 & \text{if } \hat{c} > 0 \text{ and } w_{ij} < 1 \\ \frac{1}{\hat{c}} & \text{if } \hat{c} > 0 \text{ and } w_{ij} = 1 \end{cases}$$

With normalisation g_3, the M-estimator induced clustering algorithms work quite well. However, the results are not very good if the areas of weight 1 are large and overlapping. Yet an other normalization avoids that problem by taking the relative distance of all prototypes with weight 1 into account:

$$g_4(w_{ij}) = \begin{cases} g_3'(w_{ij}) & \text{if } \hat{c} = 0 \\ 0 & \text{if } \hat{c} > 1 \text{ and } w_{ij} < 1 \\ \frac{d_{ij}}{\sum\limits_{\substack{k=1 \\ w_{kj}=1}}^{c} d_{kj}} & \text{if } \hat{c} > 1 \text{ and } w_{ij} = 1 \end{cases}$$

The only common problem for weight functions that allow strictly 0 values is, that their prototype is initialised to far away from any data object. This causes the prototype to be 'empty' which makes it impossible to move into a useful direction. But this problem is a tribute to a finite rejection point. See the next section for more detail of the usefulness of finite rejection points.

As of now, there is no noise cluster introduced. This is unnecessary during the clustering process as the robust properties of the weight functions do not require a noise cluster. However, for evaluation, the noise cluster is important. For all normalization functions holds, that the sum of membership values for one data object is ≤ 1. The noise membership value is simply determined by $1 - \sum_{i=1}^{c} w_i j$. Hence there are $c + 1$ membership values in consideration.

4. Experiments

In this section, the four M-estimator weight functions from table 1 are applied on four different data sets. The well known iris data set [30] as well as the wine data set [31] provide a relative easy clustering task. The third data set is the shuttle data set, also from the UCI data base. The shuttle data set is very imbalanced and provides a complex, multidimensional structure that is hard to cluster. We test the quality of the clusterings with several different cluster validity indices: The normalized partition coefficient (NPC), the Bezdek Seperation index (BSI), Davies-Bouldin index (DBI), Xie-Beni Index (XBI) and the value of a virtual objective function (OFV) given by equation (1). All indices are implemented using the equations from [16], chapter 7.

Table 2: Validity values of the iris data set

Index	NPC	BSI	DBI	XBI	OFV
Quality	>	<	<	<	<
Huber	**2.90**	0.293	**2.44**	**0.147**	**160**
Hampel	2.75	0.227	5.54	0.33	182
Cauchy	1.70	0.187	4.68	0.248	614
Tukey	2.00	**0.173**	4.83	0.495	743

Table 3: Validity values of the iris data set wine data set

Index	NPC	BSI	DBI	XBI	OFV
Quality	>	<	<	<	<
Huber	2.75	0.416	3.95	0.163	**1275**
Hampel	**2.80**	0.412	4.20	0.209	1634
Cauchy	1.55	0.474	**3.92**	**0.15**	1430
Tukey	2.60	**0.245**	6.27	1.23	2432

For the iris and wine data set, all M-estimators are applied, using g_3 as normalization. See table 2 for details, the 'quality' entry shows which direction of values is better (< means smaller values are better). Only the NPC value is better if larger, all other are indices to be minimized.

The Hubers M-estimator works surprisingly well in these examples. However, it is very difficult to set up. Even though it knows only one parameter, it is very sensitive to it.

The Hampel M-estimator performed quite well with relative ease. However, it is hard to optimize as it has three parameter to be set and it is hard to find the optimal combination. A visual examination also showed that the Hampel M-estimator induced clusterer performed best.

On the wine dataset (table 3), the Cauchy M-estimator did not work well. The prototypes were not well separated and also were unable to aquire full membership value. Maybe, the parameter value was wrong, but we tried a large variety of values and none of them yielded good results. So in the wine data set the Cauchy clusterer shows a trivial solution. Due to the influence of the noise cluster, the validity values are still quite reasonable.

The third example set is a very hard clustering task compared to the other two. During its execution, the robust nature of the clustering algorithms clearly showed. For this test, we changed the original objective of the clustering data set. Visual examination showed that there are around 8 clusters that are spatially separated from each other. However, they are very different populated, almost 80% of all data objects are contained in one main cluster. For this data set, it is important that the clusterer find the small satellite clusters.

Table 4: Validity values of the shuttle data set

Index	NPC	BSI	DBI	XBI	OFV
Quality	>	<	<	<	<
Huber	**5.43**	0.118	5.38	0.278	18000
Hampel	4.41	**0.105**	5.28	0.949	37500
Cauchy	3.85	0.364	5.04	**0.247**	**16500**
Tukey	3.06	0.155	**4.24**	0.441	26000

The normalization method had huge influence on the clustering quality. Because it worked best that way, g_4 was used for Huber and Hampel. Tukey however, did not work well with g_3 or g_4, g_1 was used instead. In both cases (g_3 and g_4), all prototypes clumped together at the point of highest density. This is very robust but unnecessary for clustering algorithms as a good coverage of the data objects is important as well. g_1 worked much better in this regard. But even though, 2 of the 8 prototypes were positioned well while the other 6 were still placed at the location of highest density. Changing the $d_{\text{noise}}0$ parameter did not increase the quality. Cauchys M-estimator induced clusterer however did not work on this data set. Independently from the d_{noise} parameter and normalization method, all prototypes clumped up in one location. When using g_1, all data objects had an equal membership value while in any other case almost all data objects were placed into the noise cluster. Huber and Hampels M-estimator performed reasonably well. Even though, both did not find all small clusters, they spread out quite well and covered most of the data objects.

The last data set we apply the algorithms on is an artificial data set. It contains 5 clusters Gaussian distributed clusters in a 3 dimensional environment. The clusters did not overlap which is why we used g_3 for Huber, Hampels and Tukeys M-estimator induced clusterer. Again, Cauchys M-estimator induced clusterer did not work well with g_3 which is why we used g_2. See table 5 for details.

Table 5: Validity values of the artificial data set

Index	NPC	BSI	DBI	XBI	OFV
Quality	>	<	<	<	<
Huber	4.48	0.243	4.48	0.21	4043
Hampel	**5.1**	**0.07**	**2.88**	1.37	**1586**
Cauchy	2.49	0.226	3.11	0.157	4892
Tukey	4.49	0.247	4.34	**0.1**	5883

One effect that should be mentioned is, that only Tukeys M-estimator induced clusterer found all clusters. The other M-estimator induced clusterers missed one, which seems to be a common problem with this approach. Although the test indicates that Hampels M-estimator induced clusterer might be performed best, Tukeys performed best. As this might be a problem, it could potentially be positive because prototypes that find the same cluster tend to become identical. Such an event is easy to detect and therefore, one of the identical prototypes could be removed. If initialized with an overestimated number of prototypes, the M-estimator induced clusterers could be able to estimate the number of clusters. Further study is needed at this point.

One additional remark is in order. The cluster validation methods were not designed to handle a noise cluster. But none of them requires a prototype to work. Therefore, it is unproblematic to apply them as if the noise cluster were a normal cluster. But that also influences the quality measurement as trivial solutions can be best in presence of a noise cluster. If the noise cluster is removed from the clustering process, the trivial solution to add all data objects to the noise cluster is best for minimizing one of the indices. But the same time, the values are not very well comparable when using different M-estimators as they require different noise distance set-ups. The noise distance is differently interpreted among the clusterers. Hubers M-estimator provides a membership value of 1 until the distance is larger than the noise distance. Tukeys M-estimator on the other hand provides membership values of 0 if the distance is larger than the noise distance. This alone requires different noise distance parameters for the same data set. This is also part of the reason why the validation values are in favour of Hubers M-estimator.

Performance and scalability of all M-estimator induced clusterers is similar to noise FCM (NFCM) as the basic structure is the same. However, it is likely that for a high number of data object clouds, the number of clusters entirely covered by the noise cluster is higher than for NFCM. Also the number of iterations might differ slightly.

5. Conclusions and future work

We showed in this paper, that it is possible to construct a clustering with robust properties that is based on M-estimators. In fact, the developed template works with any M-estimator weight function. However, the quality of the clustering process does not only depend on the choice of the M-estimator. The choice of the normalization process that links the prototypes together is important as well. Maybe there is one normalization process that provides a good clustering in most cases. Finding a good normalization that works in most of the cases is an interesting problem for future work.

Apart from the algorithm it self, we evaluated the clustering result using several validation indices. We presented 3 examples and the usage of the validation index for all 4 M-estimator induced clusterers. Even though the validation indices give an idea about the quality of the clustering result, the different properties of the M-estimators makes them hard to compare. A cluster validation index that takes a noise cluster into account would be welcome in this situation.

References

[1] James C. Bezdek, James M. Keller, Raghu Krishnapuram, and Nikhil R. Pal. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing.* Kluwer Academic Publishers, Boston, 1999.

[2] F. Höppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis.* John Wiley & Sons, Chichester, England, 1999.

[3] Colin Goodall. M-estimators of location: An outline of the theory. In David C. Hoaglin, Frederick Mosteller, and John W. Tukey, editors, *Understanding robust and exploratory data analysis*, Wiley series in probability and mathematical statistics, chapter 11, pages 339–400. Wiley-Interscience, 1983.

[4] Peter J. Huber. *Robust statistics.* Wiley, 1981.

[5] Rajesh N. Dave. Characterization and detection of noise in clustering. *Pattern Recogn. Lett.*, 12(11):657–664, 1991.

[6] J.C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Cybernetics and Systems: An International Journal*, 3(3):32–57, 1973.

[7] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, New York, 1981.

[8] Frank R. Hampel, Elvezio M. Ronchetti, Peter J. Rousseeuw, and Werner A. Stahel. *Robust Statistics: The Approach Based on Influ-*

ence Functions. Wiley-Interscience, New York, revised edition, April 2005.

[9] R.N. Dave and R. Krishnapuram. M-estimators and robust fuzzy clustering. In *Fuzzy Information Processing Society, 1996. NAFIPS. 1996 Biennial Conference of the North American*, pages 400–404, Jun 1996.

[10] R.N. Dave and S. Sen. Noise clustering algorithm revisited. In *Biennial Wrokshop of the North American Fuzzy Information Processing Society*, pages 199–204, Sep 1997.

[11] A. Keller. Fuzzy clustering with outliers. In *Fuzzy Information Processing Society, 2000. NAFIPS. 19th International Conference of the North American*, pages 143–147, 2000.

[12] Jongwoo Kim, Raghu Krishnapuram, and Rajesh Davé. Application of the least trimmed squares technique to prototype-based clustering. *Pattern Recognition Letters*, 17(6):633 – 641, 1996.

[13] Frank Klawonn and Frank Höppner. What is fuzzy about fuzzy clustering? understanding and improving the concept of the fuzzifier. In *Cryptographic Hardware and Embedded Systems - CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 254–264. Springer Berlin / Heidelberg, 2003.

[14] Frank Klawonn and Frank Höppner. Fuzzy cluster analysis from the viewpoint of robust statistics. In *Views on Fuzzy Sets and Systems from Different Perspectives*, volume 243 of *Studies in Fuzziness and Soft Computing*, pages 439–455. Springer Berlin / Heidelberg, 2009.

[15] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[16] Christian Borgelt. *Prototype-based Classification and Clustering (Habilitationsschrift)*. PhD thesis, Otto-von-Guericke-University of Magdeburg, Germany, 2005. http://www.borgelt.net/habil.html.

[17] Frank Klawonn and Frank Höppner. An alternative approach to the fuzzifier in fuzzy clustering to obtain better clustering. In *EUSFLAT Conf.*, pages 730–734, 2003.

[18] Jacek Leski. Towards a robust fuzzy clustering. *Fuzzy Sets and Systems*, 137:215–233(19), 16 July 2003.

[19] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[20] C. Hennig. Dissolution and isolation robustness of fixed point clusters. In *Cooperation in Classification and Data Analysis*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 27–39. Springer Berlin Heidelberg, 2009.

[21] Christian Hennig. Dissolution point and isolation robustness: Robustness criteria for general cluster analysis methods. *Journal of Multivariate Analysis*, 99(6):1154 – 1176, 2008.

[22] Christian Hennig. Cluster-wise assessment of cluster stability. *Computational Statistics & Data Analysis*, 52(1):258 – 271, 2007.

[23] YoungSik Choi and R. Krishnapuram. Fuzzy and robust formulations of maximum-likelihood-based gaussian mixture decomposition. In *Fuzzy Systems, 1996., Proceedings of the Fifth IEEE International Conference on*, volume 3, pages 1899–1905 vol.3, Sep 1996.

[24] E. Dupuis Lozeron and M.P. Victoria-Feser. Robust estimation of constrained covariance matrices for confirmatory factor analysis. *Computational Statistics & Data Analysis*, 54(12):3020 – 3032, 2010.

[25] Hichem Frigui and Raghu Krishnapuram. A robust algorithm for automatic extraction of an unknown number of clusters from noisy data. *Pattern Recognition Letters*, 17(12):1223 – 1232, 1996.

[26] Hichem Frigui and Raghu Krishnapuram. Clustering by competitive agglomeration. *Pattern Recognition*, 30(7):1109 – 1119, 1997.

[27] Hichem Frigui and Raghu Krishnapuram. A robust competitive clustering algorithm with applications in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:450–465, 1998.

[28] Fisher. Theory of statistical estimation. In *Proc. Cambridge Philosophical Society*, volume 22, pages 700–725. Cambridge University Press, Cambridge, 1925.

[29] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Rozal Statistical Society, Ser. B*, 39(1):1–38, 1977.

[30] E. Anderson. The irises of the gaspe peninsula. *Bulletin of the American Iris Society*, 59:2–5, 1935.

[31] A. Frank and A. Asuncion. UCI machine learning repository, 2010.