Detecting hyperplane clusters with adaptive possibilistic clustering

K.D. Koutroumbas
Inst. for Astronomy, Astrophysics, Space Applications and Remote Sensing, National Observatory of Athens, Greece
koutroum@noa.gr

S.D. Xenaki
Inst. for Astronomy, Astrophysics, Space Applications and Remote Sensing, National Observatory of Athens, Greece and Dept. of Informatics and Telecommunications, National & Kapodistrian University of Athens, Greece
ixenaki@noa.gr

A.A. Rontogiannis
Inst. for Astronomy, Astrophysics, Space Applications and Remote Sensing, National Observatory of Athens, Greece
tronto@noa.gr

ABSTRACT

In this paper the problem of detecting clusters whose points are spread along an $(l-1)$-dimensional hyperplane in an l-dimensional space is considered. More specifically, the recently proposed *adaptive possibilistic c-means* algorithm is modified in order to cope with this type of clusters. The main advantage of the proposed method is that it does not require a priori knowledge of the exact number of clusters. Instead, it begins with an overestimated number of them and (potentially) ends up with the true number of them. Preliminary results of the proposed algorithm on both synthetic and real data verify its validity.

CCS Concepts

•Information systems → Clustering; •Theory of computation → Unsupervised learning and clustering; •Computing methodologies → Cluster analysis;

Keywords

possibilistic clustering, adaptivity, hyperplane clusters

1. INTRODUCTION

Clustering of a set of entities is its partition to a number of groups (clusters) each one containing "similar" objects, while "less similar" entities are placed in different clusters. In practice, each entity is represented by a set of l measurements. Thus, each entity can be viewed as a point in the l-dimensional space formed by the adopted measurements (*feature space*). Under this perspective, clustering can be considered as the identification of aggregates of points (clusters) in the feature space. According to the shape of these aggregates several types of clusters result. The most well studied case, for which most of the existing clustering algorithms have been invented (e.g. k-means [5], fuzzy c-means [2], possibilistic c-means [8] and their numerous variants, see e.g. in [12], [6] and the references there in) is the case where the points are aggregated around a central point (that is, they form compact and hyperellipsoidally shaped clusters). Another interesting case, which has been studied less intensively in the last three decades, is the one where clusters are formed by points that are arranged around a surface in the feature space (e.g. [1], [2]). A significant special case of the latter is the one where the surface is a linear variety (i.e., a hyperplane in the feature space).

In the sequel, we focus on cases where the data points are spread around $(l-1)$-dimensional hyperplanes in the $\mathcal{R}^l$ feature space such as those shown in Fig. 1. This problem may arise in several applications, such as low level image processing and computer vision systems, where after the detection of the edge pixels in an image, there is the need to identify lines around which they lie (in this case it is $l=2$). This may be a first step in identifying the objects encountered in the scene depicted in the image under study. Another application, where identification of hyperplane clusters is required, is blind source separation (see [10] for details).

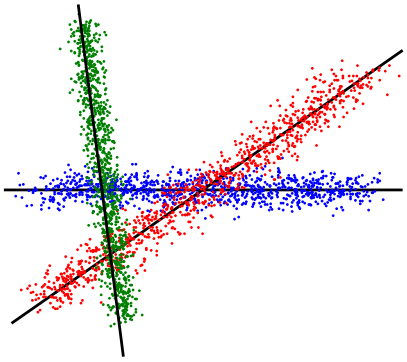
The first attempts to deal with linear variety-shaped clusters[1] go back to the works of Bezdek *et. al.* [1], where the fuzzy c-means algorithm is extended to deal with linear variety clusters. In that approach, each linear variety H_j of dimension $0 < r < l$ is characterized by a point $\boldsymbol{a}_j$ in it plus a set of r vectors $\boldsymbol{d}_j$'s that span it. The resulting algorithm, called *fuzzy c-varieties* (*FCV*) algorithm, is an iterative one and tries to adjust the parameters $\boldsymbol{a}_j$ and $\boldsymbol{d}_j$ of each H_j, so as to match the linear variety-shaped clusters underlying the data set. An issue that may arise with this formulation is that a single variety can model two or more different clus-

This research has been partially financed by the PHySIS project (http://www.physis-project.eu/), contract no. 640174, within the H2020 Framework Program of the European Commission.

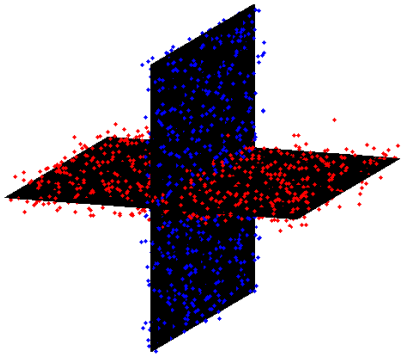
SETN '16, May 18-20, 2016, Thessaloniki, Greece

DOI: http://dx.doi.org/10.1145/2903220.2903236

[1]Note that a linear variety in the $\mathcal{R}^l$ space may be any subspace in it of dimension $r < l$. Clearly, a hyperplane is a special case of a linear variety with $r = l-1$.



(a) $\ell = 2$



(b) $\ell = 3$

Figure 1: Examples of data sets where the data points are spread around $(\ell-1)$-dimensional hyperplanes in the $\Re^\ell$ feature space for (a) $\ell = 2$ and (b) $\ell = 3$.

ters (for example, for $l = 2$, a line may model two or more collinear clusters). Although in general this feature may be undesired, it may be proved useful in certain applications; for example, in the identification of linear elements, such as roads in images depicting urban areas, where parts of a certain road (linear cluster) may not be depicted although they should. In this case, the algorithm performs some kind of "reconstruction" of the road.

Different fuzzy based formulations that may identify linear clusters as special cases are employed in the Gustafson-Kessel [4] and Gath-Geva [3] algorithms. However, a major restriction that is inherent to all fuzzy clustering schemes, is the requirement for a priori knowledge of the true number of clusters, m. This problem has been addressed via two main alternatives. According to the first one, the adopted clustering algorithm is executed for several values of m and each resulting clustering is evaluated in terms of a prespecified criterion. The clustering that optimizes this criterion is the one that is finally selected (see e.g. [12]). According to the second one, the adopted algorithm is executed once for an overdetermined number of clusters and a post processing follows, which try to merge almost coincident clusters (see e.g. [7]).

A family of algorithms that (in principle) set us free from the requirement of knowing m a priori, is that of the possibilistic algorithms. In this case, provided that the algorithm starts with an overestimated number of clusters, $m_{ini}(> m)$, it has the ability to uncover the true clusters, although some of them will be duplicated (these may be easily removed after the termination of the algorithm)[2]. However, these algorithms require a careful adjustment of a set of parameters. This issue is addressed successfully by the recently proposed *Adaptive Possibilistic c-Means* (*APCM*) algorithm [13], where (a) the involved parameters are adapted as the algorithm evolves and (b) the number of initial clusters reduces as the algorithm evolves towards (hopefully) the true one.

In this paper, we extent the APCM algorithm to handle hyperplane clusters. The paper is organized as follows. In Section 2, a brief overview of the APCM algorithm is given, while in Section 3, the proposed *Adaptive Possibilistic c-Hyperplanes* (*APCH*) algorithm is derived. Section 4 contains experimental results that validate the algorithm. Finally, Section 5 concludes the paper.

2. RELATED WORK

In this section we give a brief description of the APCM algorithm [13], which is the ancestor and basis of the proposed APCH algorithm descibed in the next section. Let $X = \{\boldsymbol{x}_i \in \mathcal{R}^l,\ i = 1, \ldots, N\}$ be the set of data points to be clustered. Let also $\Theta = \{\boldsymbol{\theta}_j \in \mathcal{R}^l,\ j = 1, \ldots, m\}$ be a set of vectors in the feature space called *cluster representatives* or, simply, *representatives*, where each one of them corresponds to a cluster C_j. In addition, $U = [u_{ij}]$ is the $N \times m$ matrix, whose (i, j)-th element u_{ij} measures the "degree of compatibility" of $\boldsymbol{x}_i$ with cluster C_j. Note that (a) all entries of U lie in the range [0, 1] and (b) all the entries of each row of U (i.e. all u_{ij}'s that correspond to a given $\boldsymbol{x}_i$) are independent from each other. Finally, let m_{ini} denote the number of clusters (representatives) with which the algorithm is initialized.

The APCM algorithm tries to identify compact and hyperellipsiodally - shaped clusters, i.e. clusters where the points are aggregated around a certain point in the feature space. To this end, the algorithm tries to move the representative vectors towards the center of dense in data regions, *independently* from each other. After the algorithm terminates, m (out of m_{ini}) representatives $\boldsymbol{\theta}_j$ will have been survived and placed to the centers of dense in data regions (clusters). Under this perspective, we say that such a $\boldsymbol{\theta}_j$ *represents* its corresponding cluster.

The updating equations for u_{ij}'s and $\boldsymbol{\theta}_j$'s, in APCM result from the minimization of the following cost function

$$J_{PCM}(\Theta, U) = \sum_{j=1}^{m} \left[\sum_{i=1}^{N} u_{ij} d(\boldsymbol{x}_i, C_j) + \right. \\ \left. + \gamma_j \sum_{i=1}^{N} (u_{ij} \ln u_{ij} - u_{ij}) \right] \quad (1)$$

as is the case with the (second) classical PCM algorithm [8], where $d(\boldsymbol{x}_i, C_j) \equiv \|\boldsymbol{x}_i - \boldsymbol{\theta}_j\|^2$. Setting $\gamma_j = \frac{\hat{\eta}}{\alpha}\eta_j$, where α is a user-defined parameter that usually takes values around 1 (the remaining notation is explained in the algorithmic form of APCM) and minimizing with respect to u_{ij} and $\boldsymbol{\theta}_j$, we

[2]Possibistic versions of the Gustafson-Kessel and Gath-Geva algorithms are discussed in [6]. Another relevant algorithm is discussed in [11].

end up with

$$u_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \boldsymbol{\theta}_j\|^2}{\gamma_j}\right) \quad (2)$$

$$\boldsymbol{\theta}_j = \frac{\sum_{i=1}^{N} u_{ij}\mathbf{x}_i}{\sum_{i=1}^{N} u_{ij}} \quad (3)$$

However, the difference between APCM and PCM is that in the former, γ_j's (η_j's) are no longer constant but they are adapted at each iteration of the algorithm. In addition, APCM provides also a mechanism for pruning clusters as it evolves, which is related to the adaptation of γ_j's. The steps of APCM are given below.

The APCM algorithm

- **Input:** X, m_{ini}, α
- $t = 0$
- *Initialization:*
  - **Initialize** $\boldsymbol{\theta}_j(t)$'s via Fuzzy c-means
  - **Set** $\eta_j(t) = \frac{\sum_{i=1}^{n} u_{ij}^{FCM}\|\boldsymbol{x}_i - \boldsymbol{\theta}_j(t)\|}{\sum_{i=1}^{n} u_{ij}^{FCM}}$, $j = 1, ..., m_{ini}$
  - **Set** $\hat{\eta} = \min_{j=1,...,m_{ini}} \eta_j(t)$
  - **Set** $m(t) = m_{ini}$
- *Main phase*
  - **Repeat**
    - **Set** $u_{ij}(t) = \exp\left(-\frac{\alpha}{\hat{\eta}}\frac{d(\boldsymbol{x}_i, C_j(t))}{\eta_j(t)}\right)$, $i = 1, ..., N$, $j = 1, ..., m(t)$
    - **Set** $\boldsymbol{\theta}_j(t+1) = \sum_{i=1}^{N} u_{ij}(t)\boldsymbol{x}_i \Big/ \sum_{i=1}^{N} u_{ij}(t)$, $j = 1, ..., m(t)$
    - **Remove** C_j if there is no $\boldsymbol{x}_i$ such that $u_{ij}(t) = \max_q u_{iq}(t)$, $j = 1, \ldots, m(t)$ and decrease $m(t)$ to $m(t+1)$ accordingly.
    - **Set** $\eta_j(t+1) = \frac{1}{n_j(t)} \sum_{\boldsymbol{x}_i : u_{ij}(t) = \max_{r=1,\ldots,m(t+1)} u_{ir}(t)} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j(t)\|$, $j = 1, ..., m(t+1)$
    - $t = t + 1$
  - **Until** the difference in $\boldsymbol{\theta}_j$'s between two successive iterations becomes suffciently small.
  - **Return** Θ, U

Note that $\boldsymbol{\mu}_j$ is the mean of the data points that are most compatible with cluster C_j (a more rigorous treatment can be found in [13]).

3. THE PROPOSED ADAPTIVE POSSIBILISTIC C-HYPERPLANES (APCH) ALGORITHM

The proposed APCH algorithm is an extension of APCM where now the cluster representatives are hyperplanes. Thus, in this case, each $\boldsymbol{\theta}_j$ is replaced by a pair of the form $(\boldsymbol{w}_j, w_{0j})$, which corresponds to the hyperplane

$$H_j : \quad \boldsymbol{w}_j^T \boldsymbol{x} + w_{0j} = 0$$

where $\boldsymbol{w}_j = [w_{j1}, \ldots, w_{jl}]^T$ and $\boldsymbol{x} = [x_1, \ldots, x_l]^T$. Now, the (squared) distance of a data point $\boldsymbol{x}_i$ from a cluster C_j is the distance of the point from its corresponding representative H_j, which is

$$d(\boldsymbol{x}_i, C_j) = (\boldsymbol{w}_j^T \boldsymbol{x}_i + w_{0j})^2 \quad (4)$$

subject to the constraint that ${\boldsymbol{w_j}}^T \boldsymbol{w}_j = 1$ (see Fig. 2). Under the above evidence, the cost function of J_{PCM} is modified to

$$J_{APCH}(\Theta, U) = \sum_{j=1}^{m}\left[\sum_{i=1}^{N} u_{ij} d(\boldsymbol{x}_i, C_j) + \right.$$
$$\left. + \gamma_j \sum_{i=1}^{N}(u_{ij}\ln u_{ij} - u_{ij}) - \lambda_j(\boldsymbol{w}_j^T \boldsymbol{w}_j - 1)\right] \quad (5)$$

where $d(\boldsymbol{x}_i, C_j)$ is given by eq. (4). Clearly, minimizing J_{APCH} with respect to u_{ij}, we end up with eq. (2), where now the distance is defined as in eq. (4). Equating the gradient of J_{APCH} with respect to w_{0j} to zero, we obtain after some manipulations

$$w_{0j} = -\mathbf{w}_j^T \boldsymbol{\theta}_j, \quad (6)$$

where

$$\boldsymbol{\theta}_j = \frac{\sum_{i=1}^{N} u_{ij}\mathbf{x}_i}{\sum_{i=1}^{N} u_{ij}} \quad (7)$$

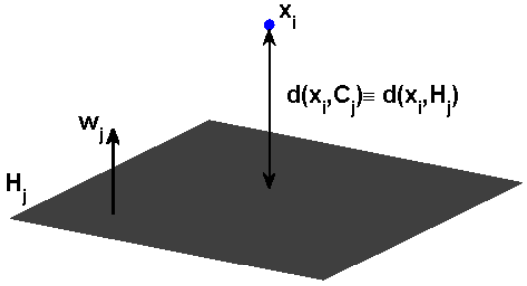


Figure 2: Graphical representation of the distance $d(\mathbf{x}_i, C_j)$ of a point $\mathbf{x}_i$ from the representative H_j of the cluster C_j.

Equating the gradient of J_{APCH} with respect to $\boldsymbol{w}_j$ to zero and utilizing eq. (6), we end up with

$$\sum_{i=1}^{N} u_{ij}(\boldsymbol{x}_i(\boldsymbol{x}_i - \boldsymbol{\theta}_j)^T)\boldsymbol{w}_j = \lambda_j \boldsymbol{w}_j \quad (8)$$

Thus, $\boldsymbol{w}_j$ is an eigenvector of the matrix

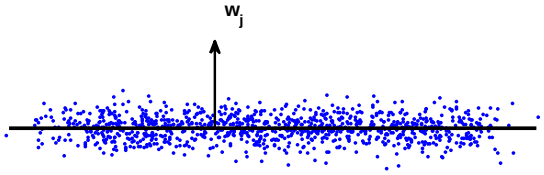
$$A_j = \sum_{i=1}^{N} u_{ij}\boldsymbol{x}_i(\boldsymbol{x}_i - \boldsymbol{\theta}_j)^T \quad (9)$$

that corresponds to the j-th cluster. Since $\boldsymbol{w}_j$ is the direction vector of H_j, that is, it is perpendicular to it, we choose $\boldsymbol{w}_j$ to be equal to the (normalized) eigenvector that corresponds to the minimum eigenvalue of A_j (that is, we choose the direction of the minimum spread around H_j, see Fig. 3). This choice is further justified by the fact that, as it

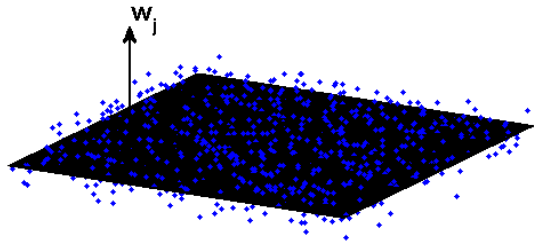
can be proved, A_j equals to the sample possibilistic covariance matrix of C_j, $\Sigma_j = \frac{\sum_{i=1}^{N} u_{ij}(\boldsymbol{x}_i - \boldsymbol{\theta}_j)(\boldsymbol{x}_i - \boldsymbol{\theta}_j)^T}{\sum_{i=1}^{N} u_{ij}}$, scaled by $\sum_{i=1}^{N} u_{ij}$, i.e.

$$A_j = \left(\sum_{i=1}^{N} u_{ij} \right) \Sigma_j \tag{10}$$

As a consequence, A_j is, in practice, positive definite and thus invertible.



(a) $\ell = 2$



(b) $\ell = 3$

Figure 3: The direction vector $\mathbf{w}_j$ of H_j is the eigenvector that corresponds to the minimum eigenvalue of A_j. Examples for (a) $\ell = 2$ and (b) $\ell = 3$.

In light of the above analysis, the APCH algorithm differs from its ancestor APCM, in two points, namely (a) in the initialization of its parameters and (b) in the updating of the representatives, where now the parameter representatives for each C_j are $\boldsymbol{w}_j$ and w_{0j}. More specifically, in the initialization phase, we generate m_{ini} clusters, C_j, $j = 1, \ldots, m_{ini}$, based on the m_{ini} "most distant" points in the data set, denoted by $\boldsymbol{\theta}_j$, $j = 1, \ldots, m_{ini}$,[3] and we cluster each of the rest points $\boldsymbol{x}_i$ of the data set X to the cluster C_j whose $\boldsymbol{\theta}_j$ lies closest to $\boldsymbol{x}_i$, among all $\boldsymbol{\theta}_q$'s, $q = 1, \ldots, m_{ini}$ (hard clustering). Denoting by X_j the points of X that have been assigned to cluster C_j, we compute the "hard version" of A_j in (9), i.e. $A'_j = \sum_{\boldsymbol{x}_i \in C_j} \boldsymbol{x}_i (\boldsymbol{x}_i - \boldsymbol{\theta}_j)^T$, $j = 1, \ldots, m_{ini}$ and we set $\boldsymbol{w}_j$ equal to the (normalized) eigenvector that corresponds to the smallest eigevalue of A'_j. Next, we compute w_{0j} via (6) and, finally, we set η_j equal to the minimum eigenvalue of A'_j bounded above by a multiple of the expected variance that exhibit the (elongated) clusters around their corresponding hyperplane representative[4]. Next, during the iterative phase of the algorithm, the updating of $\boldsymbol{w}_j$'s, w_{0j}'s and η_j's, is carried out in the same way as in the initialization phase, with the only difference that in the updating of $\boldsymbol{w}_j$'s, A_j is taken into account (eq. 9), instead of A'_j. The main steps of the proposed algorithm are summarized below.

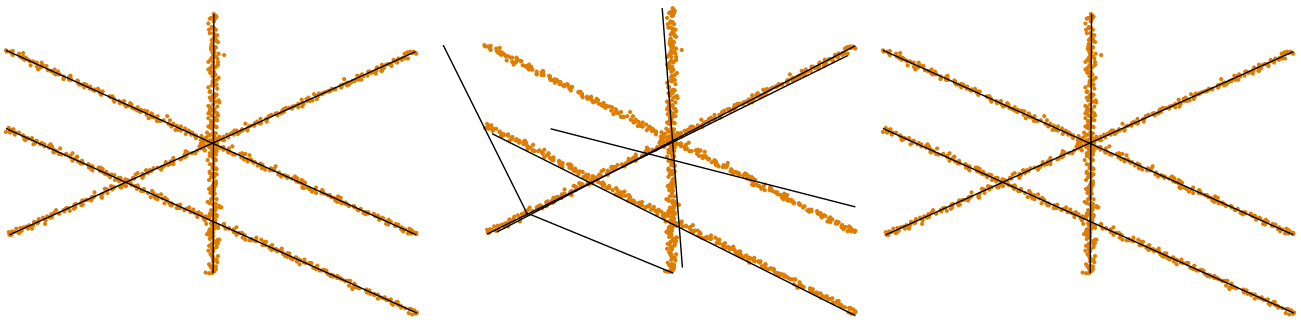
The APCH algorithm

- **Input:** X, m_{ini}, α
- $t = 0$
- *Initialization:*
  - **Find** the m_{ini} "most distant" points $\boldsymbol{\theta}_j(t)$'s, $j = 1, \ldots, m_{ini}$ in X.
  - For each cluster C_j **determine** the set of points X_j that are closer to its corresponding $\boldsymbol{\theta}_j(t)$, $j = 1, \ldots, m_{ini}$, in terms of Euclidean distance
  - **Form** $A'_j = \sum_{\boldsymbol{x}_i \in X_j} \boldsymbol{x}_i (\boldsymbol{x}_i - \boldsymbol{\theta}_j(t))^T$, $j = 1, \ldots, m_{ini}$.
  - **Set** $\boldsymbol{w}_j(t)$ equal to the eigenvector that corresponds to the smallest eigenvalue of A'_j.
  - **Set** $w_{0j}(t)$ according to eq. (6).
  - **Set** $\eta_j(t)$ equal to the smallest eigenvalue of A'_j, bounded above (see text), $j = 1, \ldots, m_{ini}$.
  - **Set** $\hat{\eta} = \min_{j=1,\ldots,m_{ini}} \eta_j(t)$
  - **Set** $m(t) = m_{ini}$
- *Main phase*
  - **Repeat**
    - **Set** $u_{ij}(t) = \exp\left(-\frac{\alpha}{\hat{\eta}} \frac{d(\boldsymbol{x}_i, C_j(t))}{\eta_j(t)} \right)$, $i = 1, ..., N$, $j = 1, ..., m(t)$
    - **Set** $\boldsymbol{\theta}_j(t+1) = \frac{\sum_{i=1}^{N} u_{ij}(t) \boldsymbol{x}_i}{\sum_{i=1}^{N} u_{ij}(t)}$, $j = 1, \ldots, m(t)$
    - **Form** $A_j(t+1) = \sum_{i=1}^{N} u_{ij}(t) (\boldsymbol{x}_i (\boldsymbol{x}_i - \boldsymbol{\theta}_j(t+1))^T)$, $j = 1, \ldots, m(t)$.
    - **Set** $\boldsymbol{w}_j(t+1)$ equal to the eigenvector that corresponds to the smallest eigenvalue of $A_j(t+1)$, $j = 1, \ldots, m(t)$.
    - **Set** $w_{0j}(t+1)$ according to eq. (6), $j = 1, \ldots, m(t)$.
    - **Set** $\eta_j(t+1)$ equal to the smallest eigenvalue of A_j, bounded above (see text), $j = 1, \ldots, m(t)$.
    - **Remove** C_j if there is no $\boldsymbol{x}_i$ such that $u_{ij}(t) = \max_q u_{iq}(t)$, $j = 1, \ldots, m(t)$ and decrease $m(t)$ to $m(t+1)$ accordingly.
    - $t = t + 1$
  - **Until** the difference in $[\boldsymbol{w}_j^T \; w_{0j}]$'s between two successive iterations becomes sufficiently small.
  - **Return** $\boldsymbol{w}_j$'s, w_{0j}'s, U

Remark 1: After the algorithm termination, there may exist hyperplanes (representatives) that do not correspond to physical clusters (usually, they intersect several physical clusters and they consist of relatively few points, that lie in the physical clusters they intersect). In order to identify and remove such clusters we work as follows: we identify the clusters of relatively small size (e.g. less than half of the mean size of clusters resulted by the algorithm) and we check whether they consist of more than one connected components (sets of points that lie away from each other). If this is the case, the cluster under study is removed. Also, there

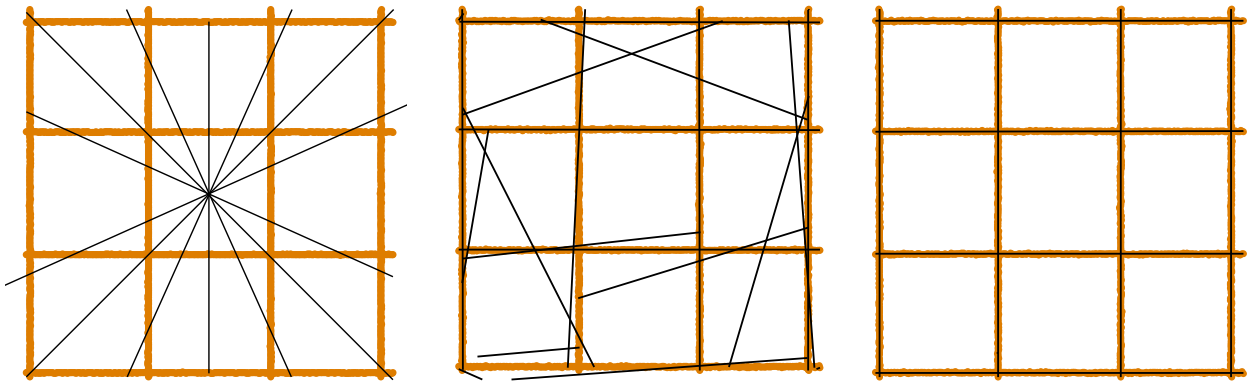
[3] They are determined by applying the *MaxMin* procedure given in [9], p.88.

[4] Actually, the algorithm turns out to be very robust to the choice of this bound.



(a) FCV (b) Initial stage of APCH (c) APCH

Figure 4: The clustering results of (a) FCV ($m = 4$) and (c) APCH ($m_{ini} = 7, \alpha = 1$) for experiment 1. The initial stage of APCH is shown in (b).



(a) FCV (b) APCH initial stage (c) APCH

Figure 5: The clustering results of (a) FCV ($m = 8$) and (c) APCH ($m_{ini} = 20, \alpha = 1$) for experiment 2. The initial stage of APCH is shown in (b).

may be duplicate clusters that have not removed. The criterion here is to identify all sets of near coincident hyperplanes and to keep only one of them.

Remark 2: As it will also be verified by the experimental results, the algorithm is capable of identifying very good initializations, i.e., initializations where most of the hyperplane clusters have been (at least crudely) identified. Of course, this make things easier during the main phase of the algorithm.

4. EXPERIMENTAL RESULTS

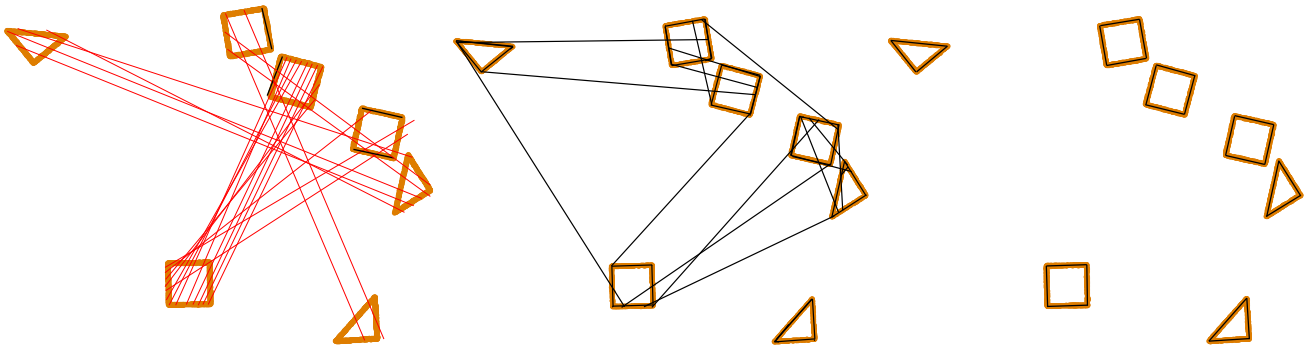
In this section we give some experimental results that verify the effectiveness of the APCH. We note again that the main advantage of the proposed algorithm compared to its fuzzy predecessors, is that it does not need the exact number of clusters a priori, but, rather, a crude overestimate of it. As the experimental results indicate, the algorithm is, in principle, capable to deal with demanding cases of intersecting linear clusters. In the sequel, we compare the clustering performance of APCH with that of FCV [1] for several synthetic data sets.

Experiment 1: Consider a two-dimensional data set consisting of $N = 1200$ points, which form four line clusters, each one having 300 data points[5]. The aim here is to explore the ability of APCH and FCV to determine the clusters. Fig. 4a shows the clustering result obtained using the FCV with $m = 4$. Figs. 4b, 4c depict the initialization stage and the final result of APCH, respectively, with $m_{ini} = 7$ and $\alpha = 1$. As it can be deduced, both clustering algorithms manage to detect the underlying clustering structure.

Experiment 2: Consider a two-dimensional data set with $N = 2400$ points, which form eight line clusters, each one consisting of 300 data points, arranged in a grid (see Fig. 5). As shown in Fig. 5a, FCV ($m = 8$) fails to distinguish any of the underlying line clusters. On the other hand, the APCH ($m_{ini} = 20, \alpha = 1$) identifies correctly all clusters (see Fig. 5c). It is noted again that the APCH algorithm achieves very good initializations with almost all lines being crudely approximated (see Fig. 5b).

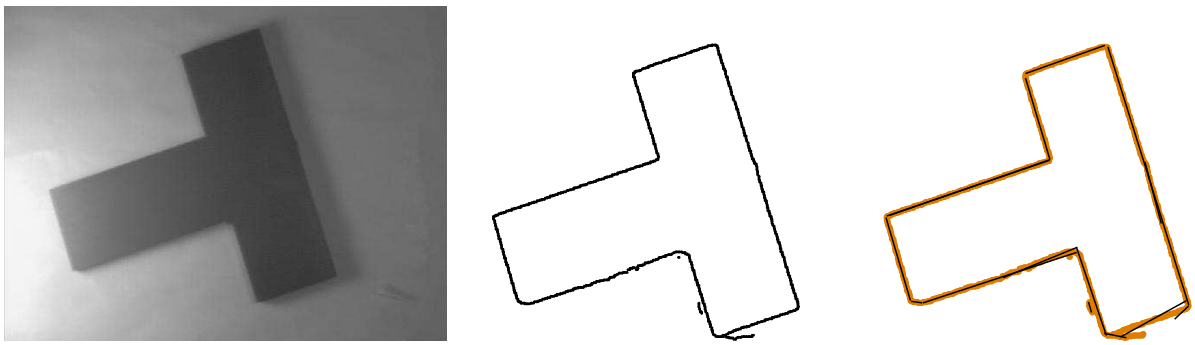
Experiment 3: Consider now a two-dimensional data set consisting of $N = 7500$ points which form 25 line clusters, each one having 300 data points. 16 of these line clusters form 4 squares and the rest 9 form 3 triangles arranged randomly in the $\Re^2$ space, as shown in Fig. 6. Fig. 6a shows

[5]In all the experiments the standard deviation of the points around the corresponding hyperplanes is 0.01.



(a) FCV (b) Initial stage of APCH (c) APCH

Figure 6: The clustering results of (a) FCV ($m = 25$) and (c) APCH ($m_{ini} = 60, \alpha = 1$) for experiment 3. The initial stage of APCH is shown in (b).



(a) Image (b) Detected edges (c) APCH result

Figure 7: (a) The image of experiment 4, (b) The detected edge pixels using the Canny method with threshold 0.2 (note that some undesired edge pixels result at the bottom right) and (c) the clustering result of APCH ($m_{ini} = 30, \alpha = 0.3$).

the clustering result obtained using the FCV with $m = 25$. Figs. 6b, 6c depict the initialization stage and the final result of APCH with $m_{ini} = 60$ and $\alpha = 1$, respectively. As it can be seen, FCV manages to detect only 4 of the 25 line clusters, whereas APCH, exploiting its good initialization, correctly identifies all clusters.

The superior results of APCH to the last two experiments may be partially attributed to its good initialization, contrary to FCV that is randomly initialized.

In the next experiment, we consider only the new APCH algorithm, aiming to accentuate its ability to successfully detecting lines that correspond to edges in real two-dimensional images.

Experiment 4: Consider the real image shown in Fig. 7a that depicts a T shape. We first apply the *Canny* method [14] with threshold 0.2 for detecting the edge pixels of the image (Fig. 7b). The aim here is to detect the linearly shaped clusters formed by these edge pixels. The resulting image was considered as the input of the APCH algorithm. Fig. 7c shows the clustering result of APCH for $m_{ini} = 30$ and $\alpha = 0.3$. As it can be deduced, APCH succeeds in identifying all the line clusters of the data set, while, additionally, achieves very accurate detection for all but one of them.

5. CONCLUSIONS

In this paper a novel possibilistic clustering algorithm, called APCH, for identifying clusters that spread around $(\ell - 1)$-dimensional hyperplanes in the $\Re^\ell$ space is introduced. The algorithm needs only an overestimated number of the clusters lying in the data set and (in principle) has the ability to uncover complex arrangements of intersected clusters. Experimental results show the effectiveness of the algorithm compared with other related methods.

Acknowledgement: We would like to thank Dr. Y. Kopsinis for bringing to our attention [10].

6. REFERENCES

[1] Bezdek J.C., Coray C., Gunderson R, Watson J.: Detection and Characterization of Cluster Substructure II. Linear Structure: Fuzzy c-varieties and Convex Combinations thereof. SIAM Journal of Applied Mathematics 40(2), 358-372 (1981)

[2] Bezdek J.C., Keller J., Krisnapuram R., Pal N.R.: Fuzzy models and Algorithms for Pattern Recognition and Image Processing, Springer (2005)

[3] Gath I., Geva A.B.: Unsupervised Optimal Fuzzy Clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence, 11, 773-781 (1989)

[4] Gustafson E.E., Kessel W. C.: Fuzzy clustering with a fuzzy covariance matrix. Proc. IEEE CDC, San Diego, CA, 1979, 761-766 (1979)

[5] Hartigan J.A., Wong M.A.: Algorithm AS136: A k-means clustering algorithm. Journal of the Royal Statistical Society, 28, 100-108 (1979)

[6] Höppner F., Klawonn F., Kruse R., Runkler T.: Fuzzy Cluster Analysis - Methods for Classification, Data Analysis and Image Recognition, Wiley (1999)

[7] Krishnapuram R., Freg C.-P.: Fitting an unknown number of lines and planes to image data through compatible cluster merging. Pattern Recognition, 25(4), 385-400 (1992)

[8] Krishnapuram R., Keller J.M.: The possibilistic c-means algorithm: insights and recommendations. IEEE Transactions on Fuzzy Systems, 4, 385-393 (1996)

[9] Mirkin B.: Clustering for Data Mining: A Data Recovery Approach. CRC Press (2005)

[10] Plumbley M.D., Blumensath T., Daudet L., Gribonval R., Davies M.E.: Sparse representation in Audio and Music: from Coding to Source separation. Proceedings of the IEEE, 98(6), 995-1005 (2010)

[11] Škrjanc I., Dovžan D.: Evolving Gustafson-Kessel Possibilistic c-Means Clustering. Procedia Computer Science, 53, 191-198 (2015)

[12] Theodoridis S., Koutroumbas K.: Pattern Recognition, 4th edn. Academic Press (2009)

[13] Xenaki S.D., Koutroumbas K.K., Rontogiannis A.A.: A Novel Adaptive Possibilistic Clustering Algorithm. IEEE Transactions on Fuzzy Systems, to appear.

[14] Canny J.: A Computational Approach to Edge Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6), 679-698 (1986)