Fast And Robust
General Purpose Clustering Algorithms

V. Estivill-Castro
School of Computing and Information Technology,
Griffith University, Nathan, QLD 4111, Australia.

J. Yang
School of Electrical Engineering and Computer Science,
The University of Newcastle, Callaghan, NSW 2308, Australia.

Abstract. General purpose and highly applicable clustering methods are usually required during the early stages of knowledge discovery exercises. k-MEANS has been adopted as the prototype of iterative model-based clustering because of its speed, simplicity and capability to work within the format of very large databases. However, k-MEANS has several disadvantages derived from its statistical simplicity. We propose an algorithm that remains very efficient, generally applicable, multidimensional but is more robust to noise and outliers. We achieve this by using medians rather than means as estimators for the centers of clusters. Comparison with k-MEANS, EXPECTATION MAXIMIZATION and GIBBS sampling demonstrates the advantages of our algorithm.

Keywords: Clustering, k-MEANS, medoids, 1-Median problem, combinatorial optimization, EXPECTATION MAXIMIZATION.

1. Introduction

Making sense of complex issues is naturally approached by breaking the subject into smaller segments that can be each explained more simply. Clustering aims at finding smaller, more homogeneous groups from a large heterogeneous collection of items [7]. Computer-assisted analysis must partition objects into groups, and must provide an interpretation of this partition [7]. Efficient clustering is a fundamental task in data mining where the goal is to discover patterns with large data sets (thousands or millions of records) that are also high dimensional.

Many clustering methods exist to partition a data set by some natural measure of similarity [1, 18, 22]. While there is no widely accepted definition of a cluster, many algorithms have been recently developed to suit specific domains. However, a general purpose and highly applicable clustering method is usually required during early stages of knowledge discovery exercises to investigate potential for data mining. The k-MEANS algorithm [35] has been widely adopted as such general purpose algorithm because of its simplicity and speed. It offers practically no limitation on the size of data sets because it typically requires linear time to obtain an approximate solution to a hard optimization problem. It also does not explicitly restrict the dimensionality of the data and it is believed that it applies to a large variety of mixtures. General purpose clustering methods should be stoppable and resumable [9, 10, 23], with the capacity to obtain a clustering solution at any time, and to be able to improve on the quality of the solution given more computational resources. They should also work within the window or access methods of databases and data-warehouses.

However, a closer look at k-Means reveals that it is probably a poor choice for a clustering task unless very specific conditions on the data are met. For example, k-Means typically succeeds when the clusters are spherical, the data is free of noise and when its operation is properly initialized. These conditions hardly occur in practical knowledge discovery situations. The weaknesses of k-Means result in poor quality clustering, and thus, more statistically sophisticated alternatives have been proposed. Representatives of these alternatives are Expectation Maximization (and model-based clustering [6, 14, 24]), Data Augmentation [46] and Gibbs sampling Markov chain Monte Carlo algorithms [3, 26, 45]. While these alternatives offer more statistical accuracy, robustness and less bias, they trade this for substantially more computational requirements and more detailed prior knowledge [36].

This paper describes a fast and robust general purpose algorithm applicable to situations in which k-Means is applied. While just slightly slower than k-Means, it offers robustness to additive and multiplicative noise. Our method is faster that the next level of sophistication (namely, Expectation Maximization) and remains conceptually simple. For example, our method does not demand the selection of a family of models for a mixture, the provision of good estimates of variance, or the provision of prior probabilities. Thus, it is simple to use. We achieve this by minimizing a different loss function in the learning of representatives of clusters. Our algorithms derive from the basic structure of iterative methods, where subtle changes produce very different optimization problems. Thus, Section 2 reviews k-Means and Expectation Maximization. Section 3 presents our algorithm in detail. Section 4 describes a series of experiments that illustrate the efficiency of our algorithm and compare it with alternative methods like k-Harmonic Means, Expectation Maximization and Gibbs sampling. This section also includes experiments that demonstrate the robustness of our method and the quality of its results with large data sets. We conclude with some final remarks in Section 5.

2. General purpose clustering algorithms

A distinct characteristic of data mining applications is the huge size of the data files involved. Thus, besides k-MEANS program-code simplicity, the attractiveness of k-MEANS is due to its computational efficiency. It requires only $O(tDkn)$ time, where t is the number of iterations over the entire data set, D is the dimension, k is the number of clusters, and n is the number of data items. As $t, D, k \ll n$ for data mining applications, k-MEANS requires $O(n)$ time. The fascination with k-MEANS's speed has motivated its adaptation for efficient processing of large sets with both numeric and categorical attributes [28, 29].

2.1. THE OPTIMIZATION PROBLEM

By iteratively improving an initial clustering (perhaps a random clustering), the k-MEANS method produces an approximate solution to the following optimization problem:

$$\text{minimize} \quad M(C) = \sum_{i=1}^{n} w_i \, \text{EUCLID}^2(\vec{s}_i, \text{REP}[\vec{s}_i, C]), \tag{1}$$

where

1. $S = \{\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_n\}$ is a set of n data items in D-dimensional real space $\mathbb{R}^D$;
2. the weight $w_i > 0$ may reflect relevance of the observation $\vec{s}_i$, and $\text{EUCLID}(\vec{x}, \vec{y}) = (\sum_{j=1}^{D} |x_j - y_j|^2)^{1/2}$ is the Euclidean metric;
3. $C = \{\vec{c}_1, \ldots, \vec{c}_k\}$ is a set of k *centers*, or representative points of $\mathbb{R}^D$; and
4. $\text{REP}[\vec{s}_i, C]$ is the closest point in C to $\vec{s}_i$; that is,
$$\text{EUCLID}(\vec{s}_i, \text{REP}[\vec{s}_i, C]) = \min_{j \in \{1,\ldots,k\}} \text{EUCLID}(\vec{s}_i, \vec{c}_j).$$

The partition into clusters is defined by assigning each $\vec{s}_i$ to its representative $\text{REP}[\vec{s}_i, C]$. Those data items assigned to the same representative are deemed to be in the same cluster; thus, the k centers encode the partition $S = C_1 | \ldots | C_k$ of the data. That is, $C_j = \{\vec{s}_i \in S | \text{EUCLID}(\vec{s}_i, \vec{c}_j) \leq \text{EUCLID}(\vec{s}_i, \vec{c}_q) \forall \vec{c}_q \in C \setminus \{\vec{c}_j\}\}$.

k-MEANS iteratively refines a partition alternating a minimization step and a classification step. In the minimization step, for each cluster in the partition, a new representative is computed. In k-MEANS, the *weighted arithmetic mean* of the cluster's points is a "center" that

k-Means type

```
(1) Construct initial set of representatives.
(2) Iterate.
    a) Classification: (Find new clusters) Assign
       each observation to its nearest representative.
    b) Minimization: (Find new representatives)
       For each cluster, find a new "center".
```

Expectation Maximization type

```
(1) Construct initial set of representatives.
(2) Iterate.
    a) Expectation: (Find new 'complete' data $\vec{Y}$)
       Evaluate $E[l(\vec{Y};\vec{\theta}^{(t)})]$.
    b) Maximization: (Find new parameters
       for model) Find $\vec{\theta}^{(t+1)}$ to maximize $E[l(\vec{Y};\vec{\theta}^{(t)})]$.
```

Figure 1. Generic iterative model-based clustering.

minimizes the sum of squared errors between the center and the points in the cluster. Next, using the new representatives, a classification step obtains new clusters. These steps are repeated until an iteration occurs in which the clustering does not change; refer to Fig. 1. This conceptual iteration of k-Means is illustrated in Fig. 1 to highlight its similarity with Expectation Maximization.

We highlight that the conceptual pseudo code of Fig. 1 is not how k-Means or Expectation Maximization should actually be implemented. This is because this conceptual pseudo code implies two passes over the data. But in both cases, the two conceptual passes can be carried out per data item in an implementation that does only one pass on the data per iteration (and obtain exactly the same result as the two-pass version).

2.2. k-Means vs Expectation Maximization

k-Means is general but too simple, Expectation Maximization is robust but too specific.

Despite its efficiency, k-Means variants have other drawbacks well documented in the literature:

1. From an optimization point of view, it often converges to a local optimum of poor quality [11, 24].

2. k-Means favors hyper-spherical clusters and that it is sensitive to scaling or similar transformations [1, 18].

3. Because k central vectors are means of cluster points, they are commonly adopted as representative of the data points of the cluster. However, it is possible for the arithmetic mean to have no valid interpretation; for example, the average of the coordinates of a group of schools may indicate that the representative school lies in the middle of a lake.

4. k-Means is very sensitive to the presence of noise and outliers, as well as to the initial random clustering [31, page 277]. In particular, much effort has been focused on the sensitivity of k-Means on the set of representatives used to initialize the search [1, 10, 23].

5. The method is statistically biased. For parametric statisticians, this implies that even if provided with the exact number of distributions in a uniform family mixture (for example, all multivariate normal distributions), and large volumes of noiseless data, k-Means converges to the wrong parameter values. This has favored other statistical methods such as Expectation Maximization [16]. k-Means is also statistically inconsistent. This has favored Bayesian and Minimum Message Length (MML) methods [17, 48]. However, these alternative methods work better when the user is able to provide an accurate probabilistic model of the classes. Also, their high sensitivity to the initial random solution has prompted researchers to incorporate initialization mechanisms [23]. Some need to approximately solve NP-hard problems as well, or use dynamic programming algorithms that require $\Omega(n^2)$ time [1].

The most popular alternative to k-Means for learning with mixtures is Expectation Maximization. This approach adds to the k-Means a probabilistic assignment treating class labels as hidden variables. Its formal analysis is much more complex than k-Means, but Expectation Maximization is consistent (asymptotically unbiased). Convergence is slow near local maxima so some implementations switch to conjugate gradient methods or other methods near a solution [37]. The foundations of this approach were originally developed for the exponential family of distributions [13], although it applies more generally [30]. There is also concern in Expectation Maximization for its sensitivity to initialization [23]. While studies on its attempts to accelerate its performance on large data sets via summarization indicate that the tails of distributions are critical [42].

Expectation Maximization corresponds to a family of iterative methods for approximating the *maximum likelihood estimate* (MLE) $\hat{\vec{\theta}}$ in the likelihood function [16, 46, 47]. Some statisticians interpret mixture data as incomplete data. This interpretation allows to frame as Expectation Maximization a procedure that starts with an initial approximation $\vec{\theta}^{(0)}$ and produces a sequence of estimates $\langle \vec{\theta}^{(t)} \rangle$. Each iteration consists of a expectation step and maximization step.

[1] A function $f(n)$ is in $O(g(n))$ if there is n_0 such that $f(n) \leq g(n), \forall n > n_0$. A function $f(n)$ is in $\Omega(g(n))$ if $f(n) \geq g(n)$ for infinitely many n. A function is $f(n)$ is $\Theta(g(n))$ if it is both $O(g(n))$ and $\Omega(g(n))$ [27].

The expectation step estimates the complete data from the incomplete data. The maximization step takes the "estimated" complete data and estimates $\vec{\theta}$ by maximum likelihood [46, 47].

Titterington et al [47] show that often, (for example, if $k = 2$ or the components are assumed to be of the same type and belonging to an exponential density family), the maximization step is explicit, in the sense that the value that attains the maximum of $E[l(\vec{Y}; \vec{\theta}^{(t)})]$ can be found algebraically, without numerical approximation, or it is of no more difficulty as a Maximum Likelihood exercise on 'complete' data.

The Expectation Maximization procedure has solid theoretical results regarding the sequence $\langle \vec{\theta}^{(t)} \rangle$ of approximations. In particular, the estimated parameters produce a sequence of likelihood values that is non-decreasing.

Unfortunately, the maximization step for $\vec{\theta_j}$, depends on the form of the part f_j of the mixture $\sum_{j=1}^{k} \pi_j f_j(\vec{\theta_j})$. Thus, different Expectation Maximization methods update their parameters at each iteration slightly differently. In order to have analytical solutions, typically, it is assumed that each f_j is a multivariate Gaussian density $N(\vec{\mu}_j, \Sigma_j)$ (normal density) [6, 14, 24]. Further simplifications assume that all components have the same known covariance Σ, and thus, the only unknown parameter of each component f_j in the mixture is the mean $\vec{\mu_j}$. Delicate aspects of the iteration occur at different levels. For example, even in the simple case where the mixture is uni-dimensional and the components f_j are all normal densities, it is important to have $\sigma_j^{(t)} \neq 0$, otherwise, the process converges to the singularities created by data points placed on a class by themselves. Thus, only the simplest versions of Expectation Maximization can compete in speed with k-Means.

2.3. Interpretation of k-Means

k-Means can be considered a direct simplification of Expectation Maximization, in that it iteratively perform a simplified expectation step and a maximization step; refer to Fig. 1. The minimization step of k-Means corresponds to the maximization step of Expectation Maximization, in the case Expectation Maximization is dealing with a mixture of k multivariate normal distributions sharing a known common covariance matrix Σ and the only unknown parameters are the mean vectors $\vec{\mu}_j$ of the components. However, the expectation step is replaced by a classification step that also has some origins in maximum likelihood estimation. The arithmetic mean serves as an estimate for $\vec{\mu}_j$. The underlying intuition follows from the observation that the multivariate normal distribution $N_{\vec{\mu}_j,\Sigma}(\vec{x})$ is large when $\|\vec{x} - \vec{\mu}_j\|^2_{\Sigma^{-1}}$ is

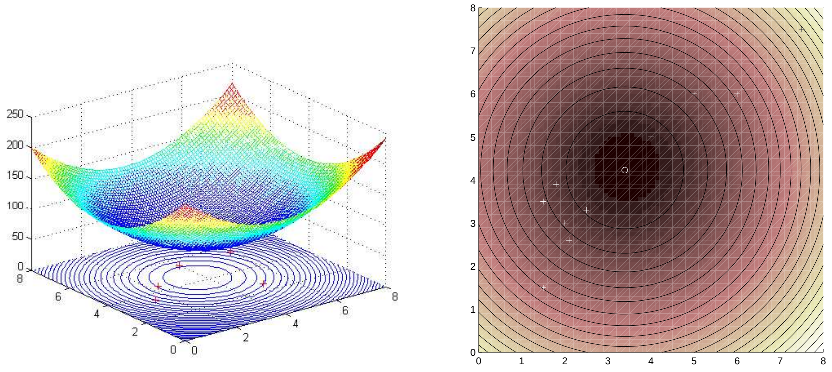
Figure 2. A function GRAVITY($\vec{x}$) and its level curves. The minimum is the gravity center (shown with $\circ$).

small, because the normal distribution has a peak at $\vec{\mu}_j$ with iso-lines (level contours) defined by the covariance matrix Σ. Thus, k-MEANS can be viewed as using the squared Euclidean distance $\|\vec{x} - \vec{\mu}_j\|_I^2 = (\vec{x} - \vec{\mu}_j)^{\mathrm{T}}(\vec{x} - \vec{\mu}_j)$ (which is easier to compute) to approximate the squared Mahalanobis distance $\|\vec{x}-\vec{\mu}_j\|_{\Sigma^{-1}}^2 = (\vec{x}-\vec{\mu}_j)^{\mathrm{T}}\Sigma^{-1}(\vec{x}-\vec{\mu}_j)$. The dependence on the squared Euclidean distance (or gravity model) [25] and not simply the Euclidean distance is a source of concern when using k-MEANS for non-Gaussian data [38].

It is very important to realize that k-MEANS is not only measuring dissimilarity by the Euclidean distance, but more importantly, it is a least squares approach. Recall that k-MEANS approximately solves the optimization problem of Equation (1) where the dissimilarity is squared. This aspect is usually unnoticed by practitioners because the pseudo-code or the descriptions of k-MEANS and its variants do not make this aspect explicit. In fact, nowhere are distances squared. However, when k-MEANS finds a representative for a cluster by computing the arithmetic mean, what has happened is that k-MEANS has found the minimum of a strictly convex function called the gravity function. More precisely, let $C_j = \{\vec{x}_1, \ldots, \vec{x}_{n_j}\}$ be the points in cluster C_j (in what follows we use $\vec{x}_i$ for a vector assigned to cluster C_j even when our description pertains to finding the representative of the one cluster C_j). Consider the objective function

$$\textsc{Gravity}(\vec{x}) = \sum_{i=1}^{n_j} w_i \textsc{Euclid}^2(\vec{x}, \vec{x}_i) = \sum_{i=1}^{n_j} w_i (\vec{x} - \vec{x}_i)^T \cdot (\vec{x} - \vec{x}_i). \quad (2)$$

An illustration of this function appears in Fig. 2. Because the function is strictly convex, it has a unique minimum. The *gradient* of $G(\vec{x})$ is a vector field whose components are the partial derivatives of $G(\vec{x})$ at $\vec{x}$. We denote by $\nabla G(\vec{x})$ the gradient of $G(\vec{x})$ and by definition $\nabla G(\vec{x}) = (\partial G/\partial x_1, \cdots, \partial G/\partial x_D)$. Then, the unique minimum can be found by solving $\nabla G(\vec{x}) = \vec{0}$. So, for the d-th coordinate,

$$0 = \partial G/\partial x_d \sum_{i=1}^{n_j} w_i(\vec{x} - \vec{x}_i)^T \cdot (\vec{x} - \vec{x}_i) = 2\sum_{i=1}^{n_j} w_i(x_j - x_{i_d}).$$

Letting $W_j = \sum_{i=1}^{n_j} w_i$, this implies $\hat{x}_d = \sum_{i=1}^{n_j} w_i x_{i_d}/W_j$. That is, the minimum is the arithmetic mean $\hat{\vec{x}}^T = (\hat{x}_1, \ldots, \hat{x}_D)$ of cluster C_j. It can be computed, as is typical in k-MEANS, in $O(n)$ time as $\hat{\vec{x}} = \sum_{i=1}^{n_j} w_i x_i/W_j$.

All variants of k-MEANS minimize GRAVITY$(\vec{x})$ for each cluster in their minimization step (see Fig. 1). Thus, k-MEANS approximates the EXPECTATION MAXIMIZATION method where after each classification step, the maximization step maximizes the likelihood by minimizing GRAVITY$(\vec{x})$ within each cluster. Because of this, k-MEANS is a least squares error method where the sum of the squared discrepancies between the representative and each data point represented is minimized. Another view is that the error incurred by choosing the arithmetic mean $\hat{\vec{x}}$ as a representative for C_j is proportional to the total sum of squared discrepancies within cluster C_j. That is, arithmetic manipulation shows that

$$\begin{aligned} 2W\text{GRAVITY}(\hat{\vec{x}}) &= 2W\left[\sum_{i=1}^{n_j} w_i(\vec{x}_i - \hat{\vec{x}})^T \cdot (\vec{x}_i - \hat{\vec{x}})\right] \\ &= \sum_{i=1}^{n_j}\sum_{m=1}^{n_j} w_i w_m \text{EUCLID}^2(\vec{x}_i, \vec{x}_m). \end{aligned} \tag{3}$$

The arithmetic mean may have no valid interpretation, so one step towards robustness is to find the point in the data that minimizes GRAVITY$(\vec{x})$. This is called the discrete center optimization and it is simple to compute. For any convex function F, the set $L(c) = \{\vec{x} \in \mathbb{R}^d | F(\vec{x}) \leq c\}$ is a convex set, for all $c \geq 0$. However, for GRAVITY$(\vec{x})$, it is not hard to show that $L(c)$ is a solid sphere [25] (in the case $D = 2$ illustrated in Fig. 2 we see that the level curves are circles). Thus, the data point that minimizes GRAVITY$(\vec{x})$ can be found in $O(n)$ time simply by finding the center of all these spheres (the arithmetic mean $\hat{\vec{x}}$) in $O(n)$ time as before, and then, finding the nearest data point to the arithmetic mean in $O(n)$ time. Unfortunately, this variant to compute an estimator of location is only slightly more robust. However, it does point out that the finding of new centers can be achieved by other methods.

3. Our algorithms

The problem with means is that they are not robust estimators of central tendency [43]. Means are very sensitive to noise and outliers. Medians represent better a typical value in skew distributions and are invariant under monotonic transformations of the random variable. Means are invariant only under linear transformations. The median of a distribution is much less tractable from the mathematical point of view than the mean. This is the main reason why traditional statistics usually chooses the mean rather than then median to describe the "center" of a distribution [12]. In clustering, as in vector quantization, the mean is to be a representative of the data points $\vec{x}_i$ that are nearest to it. The mean and the median are both measures of location [12]. Equation (1) represents what statisticians call a L_2 loss functional [43]. Thus, an immediate alternative is to use an error evaluation that measures the sum of absolute errors rather than the sum of squared of errors. This L_1 criterion results in the Fermat-Weber clustering criterion [32],

$$\text{minimize} \quad FW(C) = \sum_{i=1}^{n} w_i \, \textsc{Euclid}(s_i, \textsc{rep}[s_i, C]). \tag{4}$$

Note that the squared Euclidean metric in Equation (1) is replaced by simply the Euclidean metric.

This implies that when we find the representative for cluster $C_j = \{\vec{x}_1, \ldots, \vec{x}_{n_j}\}$ we find the minimum for the following function.

$$\text{FW}(\vec{x}) = \sum_{i=1}^{n_j} w_i \textsc{Euclid}(\vec{x}, \vec{x}_i) = \sum_{i=1}^{n_j} w_i \sqrt{(\vec{x} - \vec{x}_i)^T \cdot (\vec{x} - \vec{x}_i)}. \tag{5}$$

An illustration of this function appears in Fig. 3. Because the function is strictly convex, it has a unique minimum that is the so called Fermat-Weber(FW) center.

However, some difficulties remain in order to use the Fermat-Weber center, for instance, the *gradient* of the objective function is discontinuous in the field (the *gradient* of $\text{FW}(\vec{x})$ is not defined for data points $\vec{x} = \vec{x}_i$). We emphasize that minimizing the function $FW(\vec{x})$ of Equation (5) for cluster $C_j = \{\vec{x}_1, \ldots, \vec{x}_{n_j}\}$ is a continuous optimization problem. There is no algorithm to compute the exact coordinates of the FW center on a digital computer [5]. Other advances have been mainly theoretical [4]. However, a practical approach can be developed. The use of the extended gradient [34] results in an iterative approximation algorithm [25]. Nevertheless, its convergence or divergence depends on initialization [33], and when converging, it is slow [41].

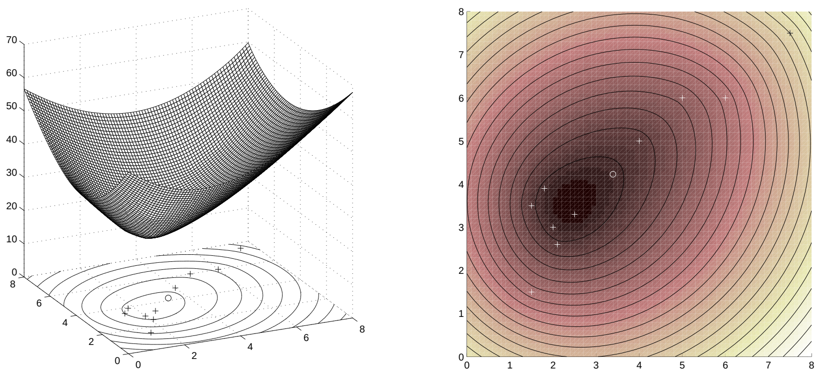

Figure 3. A Fermat-Weber function and its level curves. The arithmetic mean is shown with ○.

Solving Equation (4) with $\textsc{rep}[s_i, C] \in S$ has been named medoids clustering [39], but obtaining optimal solutions (that we call *Discrete FW Centers*) is an NP-complete problem because medoids clustering is equivalent to the p-medians problem (the optimization literature uses p rather than k for the number of representatives). However, several heuristics have been suggested to obtain medoids-based clustering [20, 21, 39]. Medoids based clustering is much more robust than k-MEANS with respect to multiplicative or additive noise, thus resulting in clustering of much better quality. However, the heuristics are still slower than k-MEANS and fundamentally applicable only for spatial data, in particular, the bi-dimensional case.

Our approach here uses FW-based clustering but in the maximization phase of an iterative algorithm of the EXPECTATION MAXIMIZATION or k-MEANS family. When k-MEANS recomputes the representatives of each cluster as the mean it is using Maximum likelihood as the inductive principle [15, 19]. However, it is also using least squares, or L_2 as the loss function. Our proposal, is to use the L_1 function. Thus, we are using a different loss function.

3.1. CONTINUOUS FW CENTER APPROACH

Our first proposal is to very closely locate the continuous FW center in each cluster during the maximization step. We call this algorithm k-CONTINUOUS-MEDIANS. For approximating the continuous FW center we use Kuhn's algorithm [34]. This algorithm is derived from the necessary and sufficient conditions for the optimum (equations that define it as a fixed point)[49]. Most applications of this algorithm for spatial

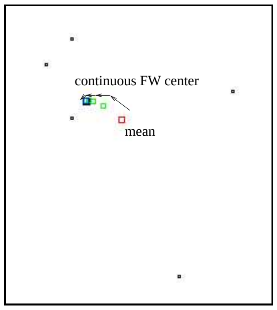

Figure 4. Trace of iteration towards the continuous FW center.

median location have been in the context of small location problems, although the convergence may be slow.

We conducted experiments to evaluate if the approximation using Kuhn's algorithm is fast and accurate enough for data mining purposes. We performed two kinds of experiments. In our first set of experiments, we took a configuration of points $(\vec{x}_1, \cdots, \vec{x}_m)$ where we knew the continuous FW center. We translated the points by a vector $\vec{x}$. We applied Kuhn's algorithm to the translated set. Then, we translated the solution obtained back by adding $-\vec{x}$. We found that the results from Kuhn's algorithm are accurate to the relative precision of the floating point system used. In our second set of experiments, we distributed points on a circle. We experimented with various radius of the circle and compared results; then we changed the density of points and compared results. This set of experiments gave similar results to the first experiment. Together these two experiments showed that despite the errors introduced by the floating-point system, the results from Kuhn's algorithm are satisfactory for data mining. Moreover, no divergent cases happened with our implementation.

Fig. 4 illustrates five points and the trace of iteration towards the CONTINUOUS FW CENTER. It also shows the location of the arithmetic mean.

3.2. Discrete FW center approach

Our second proposal is to find the data point in each cluster that minimizes FW$(\vec{x})$. That is, we solve a discrete 1-median problem for each cluster. Again, we minimize Equation (5), but now with the additional restriction that the estimator of location be in C_j. Our clustering algorithm (k-D-MEDIANS) has the same structure for k-MEANS presented in Fig. 1. However, the new center of each cluster C_j is the discrete 1-median of the points in C_j. This can trivially be solved in $O(n_j^2)$ time by evaluating $\mathrm{FW}(x_i)$ (for $i = 1, \ldots, n_j$) and returning the data point that resulted in the minimum value. However, this results in an overall clustering method requiring quadratic time. In what follows we present our strategy to obtain the discrete 1-median problem for each cluster in $O(n_j \log n_j)$ time. Our strategy does not impose any restrictions on the dimension of the data.

3.2.1. *The extended gradient as a filter*
We now show that the extended gradient is very useful as a filter.

LEMMA 3.1. *(Kuhn [33]) For $m = 1, \ldots, n_j$, let*

$$\mathrm{FW}_{\neg m}(\vec{x}) = \sum_{i=1\ i\neq m}^{n} \textsc{Euclid}(\vec{x}, \vec{x}_i)$$

be an objective function where the m-th point in C_j is excluded from consideration. Let $\nabla\mathrm{FW}(\vec{x})$ be the gradient of $\mathrm{FW}(\vec{x})$ defined in $I\!R^D \setminus C_j$. Let the extended gradient of $\mathrm{FW}(\vec{x})$ be denoted by $\nabla_E\mathrm{FW}(\vec{x})$ and defined by

$$\nabla_E\mathrm{FW}(\vec{x}) \\ = \begin{cases} \nabla\mathrm{FW}(\vec{x}) & \text{if } \vec{x} \notin C_j, \\ \max\left\{1 - \frac{1}{\|\nabla\mathrm{FW}_{\neg m}(\vec{x})\|}, 0\right\} \nabla\mathrm{FW}_{\neg m}(\vec{x}_m) & \text{if } \vec{x} = \vec{x}_m \in C_j. \end{cases}$$

Then, $\nabla_E\mathrm{FW}(\vec{x})$ is defined in $I\!R^D$ and the point f minimizes $\mathrm{FW}(\vec{x})$ if and only if $\nabla_E\mathrm{FW}(\mathsf{f}) = \vec{0}$.

The extended gradient has the properties of the gradient with respect to the level curves. For our purposes the following is most important [2].

PROPERTY 3.1. *The extended gradient vector $\nabla\mathrm{FW}_E(\vec{x})$ is normal to the tangent hyper-plane at $\vec{x}$ to $L(\mathrm{FW}(\vec{x})) = \{\vec{y} \in I\!R^D | \mathrm{FW}(\vec{y}) \leq \mathrm{FW}(\vec{x})\}$, for all $\vec{x} \in I\!R^D$.*

For an illustration in two dimensions see Fig. 5.

Thus, the extended gradient can be used as a filtering mechanism to eliminate data points than cannot possibly be the discrete 1-median.

Consider a point $\vec{x}$ on a level curve bounding $L(c)$; thus, $c = \mathrm{FW}(\vec{x})$. For illustration, refer to Fig. 5. The tangent hyper-plane of $L(c)$ at $\vec{x}$ will divide the space $I\!R^D$ into two parts. One half-space contains the set $L(c)$. We call this half space the *keeping zone.* For any point $\vec{y}$ in the other half space, $\mathrm{FW}(\vec{y}) > c = \mathrm{FW}(\vec{x})$. We call the other half space the *filtering zone.* Thus, any point in the filtering zone can be discarded

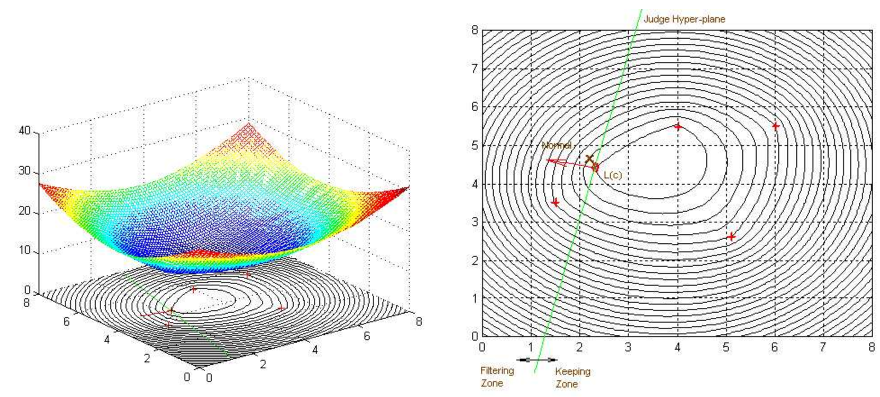

Figure 5. Fermat-Weber level curve and its normal and tangent.

from being the discrete 1-median if there is a data point $\vec{x}_i \in C_j$ in the keeping zone with $\mathrm{FW}(\vec{x}_i) \leq c = \mathrm{FW}(\vec{x})$. In addition, we call the dividing hyper-plane between the keeping zone and the filtering zone the *judge hyper-plane*. In particular, the point $\vec{x}$ is called the *judge point*.

Note also that after computation of the extended gradient we have the judge hyper-plane $L(c)$ encoded by its normal vector $\vec{n}_{\vec{x}}$. Moreover, testing if a point is in the filtering zone can be done in constant time (with respect to $n_j = \|C_j\|$). Namely, for any point $\vec{y}$ in $I\!R^D$, if the dot product between the normal $\vec{n}_{\vec{x}}$ and $\vec{y} - \vec{x}$ is not negative, $\vec{y}$ is in the filtering zone (this is because if we let $\alpha_{\vec{x}\vec{y}}$ denote the angle between $\vec{n}_{\vec{x}}$ and $\vec{y} - \vec{x}$, then $cos(\alpha_{\vec{x}\vec{y}})$ is non-negative if and only if $\vec{y}$ is in the filtering zone).

Our algorithm for 1-medoid repeatedly finds a hyper-plane and filters the points. In order for our algorithm to be effective, we need a data point $\vec{x}_i$ in the keeping zone with $\mathrm{FW}(\vec{x}_i) \leq c = \mathrm{FW}(\vec{x})$. This is easily achieved if we select $\vec{x}$ to be some data point $\vec{x}_i$. Note the importance of the extended gradient (selecting $\vec{x} = \vec{x}_i$ is impossible with the standard gradient).

However, we also need computational efficiency. Filtering passes must remove many points from further consideration because we can only afford $o(n)$ passes for a sub-quadratic clustering method.

3.2.2. *Each judge point halves the list of candidates*

We now describe our strategy for choosing hyper-planes that remove half of the candidates in each filtering step, thus reducing the n_j candidates in $\log n_j$ steps to a very small number where the discrete 1-median can be found in $O(n_j)$ time. The halving strategy will result in a total of $O(n_j \log n_j)$ time to compute the discrete 1-median.

As we already pointed out, the arithmetic mean corresponds to the center of mass. Also, by Equation (3) the mass is

$$\textsc{Gravity}(\hat{\vec{x}}) = \frac{1}{2W_j} \sum_{i=1}^{n_j} \sum_{m=1}^{n_j} \textsc{Euclid}^2(\vec{x}_i, \vec{x}_m),$$

where $W_j = \sum_{\vec{x}_i \in C_j} w_i$. Moreover, the level curves are spheres. Thus any hyper-plane on the arithmetic mean, independently of direction, will divide the mass in half. Thus, a procedure that repeatedly uses the center of mass as the judge point will require

$$O(n_j \log \sum_{i=1}^{n_j} \sum_{m=1}^{n_j} \textsc{Euclid}^2(\vec{x}_i, \vec{x}_m)/2W)$$

time in the worst case to reduce the number of candidates to a small constant. Since with $O(n_j)$ time we can make sure that

$$\sum_{i=1}^{n_j} \sum_{m=1}^{n_j} \textsc{Euclid}^2(\vec{x}_i, \vec{x}_m)/2W = O(n_j^l)$$

with l a small constant, we have a total of $O(n_j \log n_j)$ time to filter n_j items.

The problem with this procedure is that the arithmetic mean may be very close to the Fermat-Weber point and the discrete 1-median may actually be on the filtering zone. We need to select a point $\vec{x} = \vec{x}_i \in C_j$. We select the u nearest neighbors in C_j to the arithmetic mean of C_j as judge points ($u \geq 1$ a small integer constant). We also ensure that the hyper-planes cut a large proportion of the mass. There are very constrained configurations where this strategy will fail to remove a fraction of the candidates (for example, when the data points are bi-dimensional and constitute the vertices of a regular polygon). We monitor the number of candidates filtered, if the u judging hyper-planes filter less than a fourth of the candidates, we just pick the discrete center (the nearest neighbor to the arithmetic mean) as the center of this cluster. This maintains the $O(n_j \log n_j)$ time bound. Convergence of the overall clustering method results from its maximization/classification structure [44].

Thus our algorithm for finding the 'center' of cluster C_j is as follows.

Step 0. Set the list of candidates to the data points in C_j.

Step 1. Calculate the arithmetic mean $\hat{\vec{x}}$ of a list of candidates.

Step 2. Find u points $\vec{x}_m$ in the candidate set nearest to the arithmetic mean.

Step 3. If for one of the u-points $\mathrm{FW}(x_m) < \mathrm{FW}(\hat{\vec{x}})$, use $\hat{\vec{x}}$ as another judge point.

Step 4. Compute $\nabla_E \mathrm{FW}(x_m)$ and construct the normal vector with x_m as the judge point.

Step 5. Remove from the candidate list the points in u filtering zones.

Step 6. If at least a quarter of the candidates was filtered, repeat step 1 to step 5. Otherwise, return the point nearest to the arithmetic mean.

Step 7. When the list of candidates has 6 or less points, perform a brute force search for the discrete 1-median.

3.3. Other implementations

To show the difference among a group of representative-based clustering methods, we also implemented k-Harmonic Means [50] and k-c-L1 medians [11]. Fig. 6 demonstrates five points and centers of different clustering methods. k-c-L1 medians is a similar method [11] to our algorithms. k-c-L1 medians uses medians as centers but switches similarity measure to to the 1-norm to avoid the Fermat-Weber problem. Thus, it confirms that medians offer more robust iterative model-based clustering than arithmetic means. This change of similarity metric has problems because beyond 2 dimensions 1-norm median can be outside the convex hull of the cluster of points. Moreover, this alternative was proposed [11] with the use of a search for 1-norm medians that formulates a bilinear program. This bilinear program is solved iteratively in closed form. However, a much simpler implementation is possible since the 1-norm median can be obtained as the median of the projections on each coordinate [25]. Our implementation of k-c-L1 medians uses this much simpler and faster algorithm.

4. Experimental validation

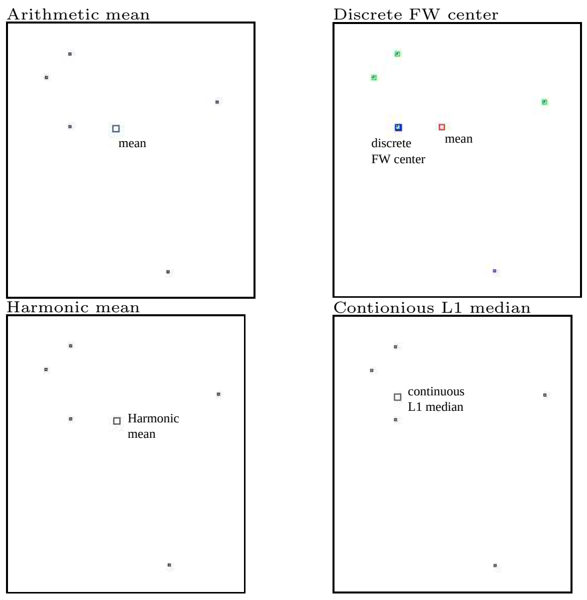

Figure 6. Four types of centers (estimators of location) on the same data set.

4.1. Performance

We first present results that evaluate the efficiency of our algorithm in Section 3.2, referred as k-d-medians. Our C implementation of k-d-medians is compared with our C implementation of k-Means and Expectation Maximization. For Expectation Maximization we used the assumption that the covariance matrix Σ_j of each component, although unknown, is diagonal [40]. The assumption that Σ_j is diagonal implies that the component densities are aligned with the axes. In a sense, the clouds of points that are the clusters are ellipsoids with axes parallel to the coordinate axes. If the assumption on diagonal from is removed, then other assumptions are required to manage the maximization step explicitly. So this is the most flexible Expectation Maximization with efficiency comparable with k-Means. We also compared with Gibbs sampling. Thus we used the 1999 release (also in C) of the *fbm-software* by R. N. Neal[2] for the Gibbs sampling and the generation of data in two examples for Bayesian mixture models.

The first example is a bivariate density estimation problem. We used the *fbm-software* to generate datasets from 500 to 100,000 points and measured the CPU time of the algorithms. The Bayesian Gibbs sampling (with two components in the mixture) remains linear because of

[2] `http://www.cs.toronto.ca/~radford/`

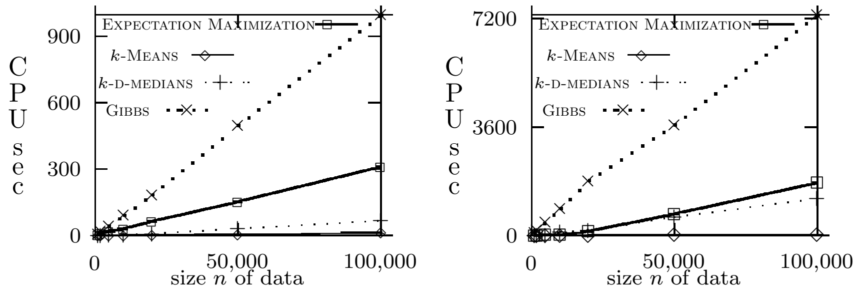

Figure 7. Comparison of CPU times for R. N. Neal examples for Bayesian mixture models.

a bound in the number of iterations but requires more than 15 minutes of CPU for 100,000 records. It requires several meta-parameters and prior probabilities but provides much more information on termination. Expectation Maximization requires 5 minutes of CPU for the same 100,000 points while k-d-medians only 1 minute and k-Means only 15 s. Thus, our algorithm is significantly faster than Expectation Maximization. In fact, time requirements are a fifth of Expectation Maximization at 5 times more than k-Means for this Bayesian model. All methods achieved equivalent quality classification with respect to Gibbs sampling with $k = 2$. Thus, the only difference for these datasets was speed. In fact, for most numerical data, specially from mixtures of Gaussians with noise, our methods was significantly faster than Expectation Maximization and just slightly slower than k-Means. The slowest remained Gibbs sampling.

The second example by R. N. Neal consist of 10 dimensional data where all attributes are categorical (and boolean). This dataset is the most difficult for our type of method, because, when data items are regarded as vectors in $\mathbb{R}^{10}$, they all are placed in the vertices of the Hilbert cube. Thus, they are all in the vertices of a convex body and clustering really is a result of repetitions. When using the *fbm-software* to generate 100,000 records, only 1024 are different. The only method of those tested designed for this type of data is Gibbs sampling which took just over 2 hrs of CPU time (using the default stopping criteria in the *fbm-software*). However, all methods obtained the 4 patterns corresponding to the 4 centers of the clusters. The only exception was k-Means, which resulted in poorer clusters for files of 20,000 records of more, while for 500 records or less k-d-medians occasionally missed one center. Thus, for these datasets quality of clustering was not an issue. However, our method becomes slower, about the same of Expectation Maximization, while k-Means remains extremely fast (only half a minute of CPU time for 100,000 items). After extensive search, we believe that this type of dataset constitutes the worst case for our method, and we are pleased to see that it remains comparable to Expectation Maximization. We should remark that we experimented with enlarging the categorical domain of attributes (from two to 5 or 10 values) and found that k-Means performance deteriorates and ours improves.

4.2. Resistance to noise

To contrast the quality of clustering in the presence of noise, we present an experimental illustration contrasting our algorithm k-d-medians with k-Means and Expectation Maximization. We first used a simple generator [20] of bi-dimensional data sets and mechanisms to regulate noise. The mixture of generator [20] produces a random vector $\vec{s_i}^T = (x_{i1}, x_{i2}) \in I\!R^2$ with a probability density function given by

$$p(\vec{x}) = \frac{1-\phi}{k} P(\vec{c_1} + r) + \ldots + \frac{1-\phi}{k} P(\vec{c_k} + r) + \phi\, U([0,1] \times [0,1]), \quad (6)$$

where $U([0,1] \times [0,1])$ denotes the uniform distribution over the unit square and $P(\vec{c}+r)$ denotes a peak distribution over the circle centered at $\vec{c}$ and radius r.

In this generator, noise is modeled as the additive term in the finite mixture model corresponding to uniform distribution on the unit square. Additive noise are points that do not belong to any cluster. Our experiments compare clustering algorithms with different levels of additive noise – increasing values of ϕ. For this purpose, the generator has an option modified to initially produce a data set with no noise. A later option can be used to perform a pass over the first dataset and the generator repeatedly generates a random number ρ uniformly in $[0,1]$. If $\rho > \phi$, it copies a data point from its input to the output. If $\rho \leq \phi$, then a point in $U([0,1] \times [0,1])$ is produced. As performed in [20], for the same data set and level of additive noise we compare with different levels of multiplicative noise – the term for it would appear multiplying in the finite mixture model. The levels of multiplicative noise are regulated by the parameter ψ.

The quality of a partition is the percentage of non-noise points that are labeled correctly by the clustering algorithm. The fewer the miss labeled points, the higher the quality of the partition. The top part of Table I shows comparisons of the algorithms for one data set of 300 data items generated by Equation (6). The algorithms compared are k-Means k-Means initialized by the results of single linkage clustering (found in $O(n \log n)$ time by Minimum Spanning Tree computation),

Table I. Misclassification with 95% confidence intervals. The top part of this table is one data set produced with GENERATOR and for each combination of ϕ, ψ each clustering algorithm is executed 10 times (with different initialization). The bottom part combines 10 different data sets produced with GENERATOR.

$n = 300$		One data set (10 runs per set)			
$k = 10$		Algorithm			
$u = 1$		k-MEANS		k-D-MEDIANS	EM
Noise		Random	MST	Random	Random
ψ	ϕ	start	start	start	start
0	0	39% ±7	8%	16%±4	25%±5 %
	0.1	30% ±4	27%	16%±4	30%±4
	0.2	30% ±5	30%	22%±5	39%±3
0.5	0	29% ±5	12%	14%±4	24%±4
	0.1	30% ±4	30%	14%±4	38%±4
	0.2	30% ±5	31%	18%±5	38%±4
1.0	0	33% ±5	19%	18%±4	22%±3
	0.1	26% ±6	36%	17%±4	39%±4
	0.2	35% ±6	30%	15%±5	38%±4
1.5	0	34% ±4	22%	18%±4	30%±5
	0.1	30% ±4	31%	14%±5	37%±10
	0.2	34% ±4	30%	20%±5	34%±4

$n = 300$		10 data sets (10 runs per set)			
$k = 10$		Algorithm			
$u = 1$		k-MEANS		k-D-MEDIANS	EM
Noise		Random	MST	Random	Random
ψ	ϕ	start	start	start	start
0	0	31% ±4	7%±2	16%±5	24%±6
	0.1	30% ±4	23%±5	18%±5	42%±6
	0.2	30% ±5	30%±3	20%±4	40 %±6
0.5	0	30% ±4	12%±4	19%±7	24%±4
	0.1	30% ±4	30%±4	19%±7	37%±6
	0.2	30% ±5	30%±5	18%±6	39%±7
1.0	0	31% ±5	20%±4	22%±6	22%±4
	0.1	30% ±6	32%±4	20%±4	38%±7
	0.2	31% ±6	30%±5	19%±5	40%±8
1.5	0	33% ±4	22%±4	18%±5	31%±6
	0.1	32% ±4	31%±5	20%±4	38%±9
	0.2	32% ±4	31%±5	20%±5	36%±8

our k-D-MEDIANS and EXPECTATION MAXIMIZATION (with diagonal covariance matrices). The bottom part of the table combines the results from 10 different data sets, each generated by the model in Equation (6). The results clearly show the robustness of our algorithm to both types of (additive and multiplicative) noise.

4.3. 3D MIXTURE DATA TEST

This experiment consisted of generating data with respect to a mixture of 3-dimensional (multivariate) normal distributions with noise. Thus, data was generated with the form $p(\vec{x}) = \pi_1 N_{\vec{\mu}_1,\Sigma_1}(\vec{x}) + \ldots + \pi_k N_{\vec{\mu}_k,\Sigma_k}(\vec{x}) + \pi_{k+1} U(\vec{x})$ where each component $N_{\vec{\mu}_j,\Sigma_j}(\vec{x})$ is a multivariate normal distributions with mean $\vec{\mu}_j$ and the covariance matrix Σ_j and $U(\vec{x})$ is the uniform distribution in a box that bounds all $\vec{\mu}_j$. Again we compared k-MEANS, EXPECTATION MAXIMIZATION, and our algorithms: our implementation of k-C-L1 MEDIANS and k-D-MEDIANS. We used 20% noise, $k = 3$ and $\pi_j = .8/k$ for $j = 1, \ldots, k = 3$. Moreover, the three covariance matrices Σ_j were set to the identity in order to create data sets as favorable as possible to k-MEANS and EXPECTATION MAXIMIZATION.

We evaluated the quality of the clustering results by the sum of the norms between the original $\vec{\mu}_j$ and the approximations $\hat{\vec{\mu}}_j$ obtained for the algorithms. Data sets with $n = 2,000$ were generated by selecting three points $\vec{\mu}_j$ at random in $[0, 20.0] \times [0, 20.0] \times [0, 20.0]$. For example, a typical data set had $\vec{\mu}_1^T = (13.1, 7.6, 6.9)$, $\vec{\mu}_2^T = (2.6, 7.1, 14.5)$ and $\vec{\mu}_3^T = (16.6, 9.3, 14.9)$. Table II shows the results for one data set. Typically, the sum of discrepancies between norms was twice as large for k-MEANS than for k-D-MEDIANS. k-D-MEDIANS consistently outperformed the others with respect to quality. On average, EXPECTATION MAXIMIZATION is closer than the results in Table II but still surpassed by k-D-MEDIANS. k-C-L1 MEDIANS is the fastest, followed by k-D-MEDIANS. k-MEANS has problems detecting convergence and sometimes is the slowest. Both EXPECTATION MAXIMIZATION and k-MEANS end up being several times slower that k-D-MEDIANS and k-C-L1 MEDIANS. k-C-L1 MEDIANS can produce some poor results, as illustrated by Table II.

Because the results of these algorithms depend on their random initialization, we have compared them when they are executed 10 times, but for each execution, all algorithms start with the same initial configuration. Using 10 different 3D mixtures (only the true means are different), the first 4 columns of Table III shows the results of 4 algorithms starting 10 times with common random initialization. Clearly EXPECTATION MAXIMIZATION and k-MEANS have large error and small variance. FUZZY-c-MEANS [8] has small error on average and nil variance. This shows that FUZZY-c-MEANS is the least subject to the initialization configuration. In practical settings, all these algorithms should be executed several times and the best solution of all (measured by the criteria they are optimizing) should be adopted as the answer from such algorithm. This multi-start version eliminates the

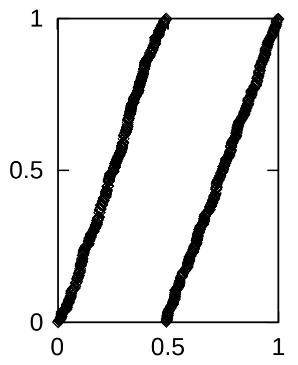

Figure 9. Data is uniform if projected to either axis, but has a clear pattern.

Table II. Results for 3-D mixture of normals with 20% noise.

Algorithm	Estimated $\hat{\vec{\mu}}_j^T$	$\sum_{j=1}^{3} \lVert\hat{\vec{\mu}}_j - \vec{\mu}_j\rVert$	CPU time
k-MEANS	$\vec{\mu}_1^T = (12.7, 8.4, 6.3)$ $\vec{\mu}_2^T = (3.1, 7.8, 13.5)$ $\vec{\mu}_3^T = (16.3, 9.6, 14.8)$	2.83	96 sec
EXPECTATION MAXIMIZATION	$\vec{\mu}_1^T = (10.1, 9.7, 9.4)$ $\vec{\mu}_2^T = (2.7, 7.1, 14.4)$ $\vec{\mu}_3^T = (15.1, 8.6, 11.3)$	7.77	5 sec
k-D-MEDIANS	$\vec{\mu}_1^T = (12.8, 8.0, 6.9)$ $\vec{\mu}_2^T = (2.8, 7.1, 14.4)$ $\vec{\mu}_3^T = (16.6, 9.6, 14.8)$	0.97	0.7 sec
k-C-L1 MEDIANS	$\vec{\mu}_1^T = (7.8, 16.0, 6.0)$ $\vec{\mu}_2^T = (3.8, 7.1, 14.4)$ $\vec{\mu}_3^T = (15.0, 8.5, 9.8)$	16.8	0.4 sec

dependency on the initial configuration. We declared an algorithm a multi-start winner, if it produced the best result in each of 10 multi-starts, each consisting of 10 random initial configurations. The last column of Table III shows that our algorithm was the winner for the 10 data sets. Occasionally our methods provide poor results, on some initial configurations. But their multi-start version is superior to all others.

k-C-L1 MEDIANS is faster than k-D-MEDIANS, but one should be aware that k-C-L1 MEDIANS performs poorly when the clusters are the synergy (interaction) of the attributes. Fig. 9 shows two 2D clusters. They have the very distinctive pattern: they are lines. However, for each coordinate, the data is a uniform distribution. The algorithm

Table III. Results for 10 dataset each consisting of a 3-D mixture of normals with 20% noise.

	10 common start configurations Error mesured as $sum_{i=1}^{3} \|\hat{\vec{\mu}}_j - \vec{\mu}_j\|$				Multistart
Dataset	EXPECTATION MAXIMIZATION	k-MEANS	FUZZY-c-MEANS	k-D-MEDIANS	Winner
1	4.75 ±0	5.65 ±4	1.92 ±0	13.8 ±10	k-D-MEDIANS
2	3.96 ±3	6.17 ±3	1.94 ±0	10.90 ±5	k-D-MEDIANS
3	9.84 ±3	15.78 ±2	6.21 ±3	7.77 ±6	k-D-MEDIANS
4	10.59 ±5	4.93 ±4	1.57 ±0	5.12 ±6	k-D-MEDIANS
5	11.16 ±3	5.80 ±4	2.03 ±0	7.54 ±5	k-D-MEDIANS
6	15.09 ±0	13.04 ±5	1.80 ±0	10.09 ±6	k-D-MEDIANS
7	10.88±2	6.29 ±4	1.07 ±0	5.60 ±5	k-D-MEDIANS
8	3.3 ±0	5.31 ±4	4.17 ±6	5.51 ±6	k-D-MEDIANS
9	13.35 ±3	9.52 ±4	2.72 ±0	7.13 ±5	k-D-MEDIANS
10	9.11 ±2	6.79 ±4	2.21 ±0	4.54 ±5	k-D-MEDIANS

k-C-L1 MEDIANS finds cluster with over 40% error 90% of the time and 10% of the time is totally wrong providing two clusters separated by the line $Y = 0.5$. By contrast, k-D-MEDIANS performs very well in this data set, 90% of the time the misclassification is only 10%.

The statistical literature has rejected the L1-metric optimization because the estimator of location can be outside the convex hull of the cloud of points for which it is estimating a center [43]. A simple example is the 3 point set (1,0,0), (0,1,0) and (0,0,1) in 3D. The L_1 center is (0,0,0) which is outside the convex hull.

4.4. A LARGE TEXT DATASET EXPERIMENT

As a benchmark for our algorithms we used a setting in text clustering. We conducted experiments on a well-known REUTERS-21578 dataset[3]. This dataset has been used before as a benchmark for representative-based clustering in data mining [9, 23]. We reproduced the experiments to asses our algorithms.

We derived two datasets from the original text file `reut2-all.sgm` consisting of 21,578 SGML marked documents. Dataset one (D1) has 302 dimensions while dataset two (D2) has 135 dimensions. D1 columns correspond to the 302 most frequent words in the entire collection. A record in D1 corresponds to an document and has the counts for these 302 words as they appear in the document. D2 columns correspond to the 135 topics in the entire collection. A record in D2 is a vector of counts for these 135 topics.

[3] `http://www.research.att.com/~lewis/reuters21578/README.txt`

Our experiment compares clustering these two large datasets with five different clustering methods. These are our two algorithms k-D-MEDIANS and k-CONTINUOUS-MEDIANS, our novel implementation of k-C-L1 MEDIANS and two fast versions of k-MEANS, a simple standard k-MEANS and also k-HARMONIC MEANS.

The 302 most frequent words are extracted from all words between the tag pairs ($< \ TEXT > < /TEXT >$), ($< TITLE > < /TITLE >$) and ($< BODY > < /BODY >$), and included abbreviations like *mln*, *pct* and *corp*.

There is a standard set of 135 topic words for the Reuters dataset. However not all topics appear in the dataset. Actually only 120 topic words appear. Despite some documents have no topic, we do not exclude them in our experiment.

Datasets D1 and D2 are used in our experiment as raw data for clustering analysis. We would like to identify which of all the representative-based clustering algorithms provides the best clusters.

Because the algorithms we are comparing are optimizing different criteria, using any of these criteria to compare all algorithms could be seen as favorable to the algorithm that is explicitly attempting to optimize such criteria. Therefore, because clustering aims at reducing diversity within a class, we decided to use entropy on the attribute-vectors to measure the diversity in clustering results. We also used total gravity for a measure, because this is the criteria used by k-MEANS. We use this to show that k-MEANS is actually the poorest performance even for the criteria it is explicitly optimizing.

For the entropy analysis, we first compute dissimilarity inside each cluster. For a given cluster C_j with n_j elements and m dimensions, after normalizing each vector, we add all the values in each attribute. That is, we find the vector

$$\vec{e} = \sum_{\vec{x_i} \in C_j} \vec{x_i}.$$

Now if $\vec{e}^T = (e_1, e_2, \cdots, e_m)$, the entropy of cluster C_j is given by

$$E_n(C_j) = -\sum_{i=0}^{m} \frac{e_i}{n_j} \log_2(\frac{e_i}{n_j}). \tag{7}$$

The entropy of a clustering result is just the sum of the entropy in all clusters. The clustering method that has smaller entropy has reduced diversity in the clusters the best. The gravity of each cluster is the sum of squared Euclidean distance of all points in the cluster C_j to their representative, and the total gravity is the sum of gravity over all clusters C_j.

Table IV. Algorithms ranked by quality of the clustering on Reuters datasets consisting of 21,578 records.

$n = 21,578$	Data set D1 (dimensions $D = 302$)			Data set D2 (dimension $D = 135$)		
Algorithm	Entropy	Gravity	CPU s	Entropy	Gravity	CPU s
k-D-MEDIANS	440	2 622,467	3,521	40	3,742	312
k-CONTINUOUS-MEDIANS	423	2 764,454	885	49	4,093	226
k-MEANS	505	2 782,003	288	62	5,713	72
k-HARMONIC MEANS	524	3 342,582	768	41	10,034	75
k-C-L1 MEDIANS	553	3 024,238	2,066	68	10,983	739

We performed this experiment on implementations in Java. Table IV shows a summary of the experimental results. Clearly, for both, Entropy and Gravity assessment of the clusters, our algorithms obtain better clustering results. Also, our algorithms are scalable, they are linear in D (the dimension) and $O(n \log n)$ time is required, where n is the size of the datasets. The CPU time measurements show that they are a constant factor of about 5 slower than fast versions of k-MEANS(this is across the arguments n and D). However, they compensate this fixed overhead by much improved clustering results. The readers may note that once k-HARMONIC MEANS performed better with respect to Entropy, but much worse with respect to Gravity. This seems to indicate that Gravity is inherently a less effective clustering criteria, sensitive to outliers [38].

5. Final remarks

Section 4.1 shows that our algorithms are slightly more costly than k-MEANS but certainly much faster than alternatives like EXPECTATION MAXIMIZATION and Gibbs Sampling. Section 4.2 shows that our algorithm provide much more resistance to noise and outliers. Section 4.3 shows that they offer high statistical quality. Section 4.4 shows how our algorithms can be applied successfully to a case study previously used in the Data Mining literature. Our algorithms produce clusterings with improved results.

The algorithms presented here are suitable for exploratory data analysis. They do not depend on the order of the data, as some variants of k-MEANS and they do not demand detailed initialization. Their use brings insight into the structure of a large multidimensional data set. Because they are faster than EXPECTATION MAXIMIZATION, they can be applied in combination with criteria for determining the number k of clusters. Recall that the most robust criteria to find an estimate of the value of k by repeatedly cluster with different values of k [39, 40].

References

1. M.S. Aldenderfer and R.K. Blashfield. *Cluster Analysis.* Sage Publications, Beverly Hills, USA, 1984.
2. T.M. Apostol. *Calculus — Volume II.* John Wiley & Sons, NY, USA, second edition, 1969.
3. S.F. Arnold. Gibbs sampling. In C.R. Rao, editor, *Handbook of Statistics 9*, pages 599–625, Amsterdam, 1993. North Holland.
4. S. Arora, P. Raghavan, and S. Rao. Approximation schemes for Euclidean k-medians and related problems. In *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 106–113, Dallas; TX, May 1998. ACM, ACM Press. ISBN: 0897919629.
5. C. Bajaj. Proving geometric algorithm non-solvability: An application of factoring polynomials. *Journal of Symbolic Computation*, 2:99–102, 1986.
6. J.D. Banfield and A.E. Raftery. Model-Based Gaussian and non-Gaussian clustering. *Biometrics*, 49:803–821, September 1993.
7. M.J.A. Berry and G. Linoff. *Data Mining Techniques — for Marketing, Sales and Customer Support.* John Wiley & Sons, NY, USA, 1997.
8. J.C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, New York, 1981.
9. P.S. Bradley and U. Fayyad. Refining the initial points in k-means clustering. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 91–99, San Mateo, CA, 1998. Morgan Kaufmann Publishers.
10. P.S. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In R. Agrawal and P. Stolorz, editors, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 9–15. AAAI Press, 1998.
11. P.S. Bradley, O.L. Mangasarian, and W.N. Street. Clustering via concave minimization. *Advances in neural information processing systems*, 9:368–, 1997.
12. M.G. Bulmer. *Principles of Statistics.* Dover, NY, second edition, 1979.
13. G. Casela and R.L. Berger. *Statistical Inference.* Wadsworth & Brooks/Cole, Belmont, CA, 1990.
14. G. Celeux and G. Govaret. Gaussian parsimonious clustering models. *Pattern Recognition*, 28(5):781–793, 1995.
15. V. Cherkassky and F. Muller. *Learning from Data — Concept, Theory and Methods.* John Wiley & Sons, NY, USA, 1998.
16. A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likehood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.
17. D. Dowe, R.A. Baxter, J.J. Oliver, and C. Wallace. Point estimation using the Kullback-Leibler loss function and MML. In X. Wu, R. Kotagiri, and K.K. Korb, editors, *Proceedings of Second Pacific-Asia Conference on Knowledge Discovery and Data Mining PAKDD-98*, pages 87–95, Melbourne, Australia, 1998. Springer-Verlag Lecture Notes in Artificial Intelligence 1394.
18. R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, NY, USA, 1973.
19. V. Estivill-Castro. Why so many clustering algorithms – a position paper. *SIGKDD Explorations*, 4(1):65–75, June 2002.
20. V. Estivill-Castro and M.E. Houle. Robust distance-based clustering with applications to spatial data mining. *Algorithmica*, 30(2):216–242, June 2001.

21. V. Estivill-Castro and A.T. Murray. Discovering associations in spatial data - an efficient medoid based approach. In X. Wu, R. Kotagiri, and K.K. Korb, editors, *Proceedings of the 2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-98)*, pages 110–121, Melbourne, Australia, 1998. Springer-Verlag Lecture Notes in Artificial Intelligence 1394.
22. B. Everitt. *Cluster Analysis.* Halsted Press, New York, USA, 2nd. edition, 1980.
23. U. Fayyad, C. Reina, and P.S. Bradley. Initialization of iterative refinement clustering algorithms. In R. Agrawal and P. Stolorz, editors, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 194–198. AAAI Press, 1998.
24. C. Fraley and A.E. Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *Computer Journal*, 41(8):578–588, 1998.
25. R.L. Francis. *Facility layout and location: An analytical approach.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1974.
26. A. Gelman, J.B. Carlin, H.S. Stern, and D.B. Rubin. *Bayesian Data Analysis.* Chapman & Hall, London, 1995.
27. R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics.* Addison-Wesley Publishing Co., Reading, MA, 1989.
28. S.K. Gupta, K.S. Rao, and V. Bhatnagar. K-means clustering algorithm for categorical attributes. In M. Mohania and A.M. Tjoa, editors, *Data Warehousing and Knowledge Discovery DaWaK-99*, pages 203–208, Florence, Italy, 1999. Springer-Verlag Lecture Notes in Computer Science 1676.
29. Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, 1998.
30. M.I. Jordan and R.I. Jacobs. Supervised learning and divide-and-conquer: A statistical approach. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 159–166, San Mateo, CA, 1993. Morgan Kaufmann Publishers.
31. L. Kaufman and P.J. Rousseuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley & Sons, NY, USA, 1990.
32. H.W. Kuhn. On a pair of dual non-linear problems. In J. Abadie and S. Vajda, editors, *Nonlinear programming*, page Chapter 3, NY, USA, 1967. John Wiley & Sons.
33. H.W. Kuhn. A note on Fermat's problem. *Mathematical Programming*, 4(1):98–107, 1973.
34. H.W. Kuhn and E. Kuenne. An efficient algorithm for the numerical solution of the generalized Weber problem in spatial economics. *Journal of Regional Science*, 4(2):21–33, 1962.
35. J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. Le Cam and J. Neyman, editors, *5th Berkley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967. Volume 1.
36. S. Massa, M. Paolucci, and P.P. Puliafito. A new modelling technique based on Markov chains to mine behavioral patterns in event based time series. In M. Mohania and A.M. Tjoa, editors, *Data Warehousing and Knowledge Discovery DaWaK-99*, pages 331–342, Florence, Italy, 1999. Springer-Verlag Lecture Notes in Computer Science 1676.
37. I. Meilijson. A fast improvement to the EM algorithm in its own terms. *Journal of the Royal Statistical Society B*, 51(1):127–138, 1989.

38. A.T. Murray and V. Estivill-Castro. Cluster discovery techniques for exploratory spatial data analysis. *International Journal of Geographic Information Systems*, 12(5):431–443, 1998.
39. R.T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In J. Bocca, M. Jarke, and C. Zaniolo, editors, *Proceedings of the 20th Conference on Very Large Data Bases (VLDB)*, pages 144–155, San Francisco, CA, 1994. Santiago, Chile, Morgan Kaufmann Publishers.
40. J.J. Oliver, R.A. Baxter, and C.S. Wallace. Unsupervised learning using MML. In L. Saitta, editor, *Proceedings of the 13th Machine Learning Conference*, pages 364–372, San Mateo, CA, July 1996. Morgan Kaufmann Publishers.
41. M.L. Overton. A quadratically convergent method for minimizing a sum of Euclidean norms. *Mathematical Programming*, 27:34–63, 1983.
42. G.W. Rogers, B.C. Wallet, and E.J. Wegman. A mixed measure formulation of the EM algorithm for huge data set applications. In L. Billard and N.I. Fisher, editors, *Proceedings of the 28th Symposium on the Interface between Computer Science and Statistics*, pages 492–497, Sydney, Australia, July 1997. Interface Foundation of North America.
43. P.J. Rousseeuw and A.M. Leroy. *Robust regression and outlier detection.* John Wiley & Sons, NY, USA, 1987.
44. S.Z. Selim and M.A. Ismail. k-means-type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(1):81–86, January 1984.
45. A.F.M. Smith and G.O. Roberts. Bayesian computation via the Gibbs sampler and reated Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society B*, 55(1):2–23, 1993.
46. M.A. Tanner. *Tools for Statistical Inference.* Springer-Verlag, NY, US., 1993.
47. D.M. Titterington, A.F.M. Smith, and U.E. Makov. *Statistical Analysis of Finite Mixture Distributions.* John Wiley & sons, UK, 1985.
48. C.S. Wallace and P.R. Freeman. Estimation and inference by compact coding. *Journal of the Royal Statistical Society, Series B*, 49(3):240–265, 1987.
49. G. Wesolowsky. The Weber problem: history and perspectives. *Location Science*, 1:5–23, 1993.
50. B. Zhang, M. Hsu, and U. Dayal. K-harmonic means — a spatial clustering algorithm with boosting. In J. Roddick and K. Hornsby, editors, *Proceedings of the International Workshop on Temporal, Spatial and Spatio-Temporal Data Mining - TSDM2000, in conjunction with the 4th European Conference on Principles and Practices of Knowledge Discovery and Databases*, pages 31–42, Lyon, France, 2000. Springer-Verlag Lecture Notes in Artificial Intelligence 2007.

Address for Offprints: School of Computing and Information Technology, Griffith University, Nathan, QLD 4111, Australia.