Generalized *K*-Harmonic Means
-- Dynamic Weighting of Data in Unsupervised Learning

Bin Zhang
Hewlett-Packard Laboratories, bzhang@hpl.hp.com

Abstract We propose a new class of center-based iterative clustering algorithms, K-Harmonic Means (KHM_p), which is essentially insensitive to the initialization of the centers, demonstrated through many experiments. The insensitivity to initialization is attributed to a dynamic weighting function, which increases the importance of the data points that are far from any centers in the next iteration. The dependency of the K-Means' and EM's performance on the initialization of the centers has been a major problem. Many have tried to generate good initializations to solve the sensitivity problem. KHM_p addresses the intrinsic problem by replacing the *minimum* distance from a data point to the centers, used in K-Means, by the Harmonic Averages of the distances from the data point to all centers. KHM_p significantly improves the quality of clustering results comparing with both K-Means and EM. The KHM_p algorithms have been implemented in both sequential and parallel languages and tested on hundreds of randomly generated datasets with different data distribution and clustering characteristics.

Keywords – Clustering, K-Means, K-Harmonic Means, EM, Data Mining, Dynamic Modeling

1. Introduction

Clustering is one of the principal workhorse techniques in the field of data mining [FPU96], statistical data analysis [KR90], data compression and vector quantization [GG92], and many others. K-Means (*KM*), first developed more than three decades ago [M67], and the Expectation Maximization (*EM*) with linear mixing of Gaussian density function [DLR77] are two of the most popular clustering algorithms [BFR98a], [SI84], [MK97]. See [GG92] for more complete references for K-Means and [MK97][RW84] for EM.

K-Means stands out, among the many clustering algorithms developed in the last few decades, as one of the few most popular algorithms accepted by many application domains. However, K-Means does have a widely known problem – the local optimum it converges to is very sensitive to the initialization. Many people have proposed initialization algorithms.

Instead of inventing or improving an initialization for K-Means, we look into the intrinsic problem that resulted in K-Means sensitivity to initialization – its winner-takes-all partitioning strategy, which makes the association between data points and the nearest center so strong that the membership of a data point is not changed until it is closer to a different center. This strong association prevents the centers from moving out of a local density of data. We use the association provided by the *harmonic means* function, to replace the winner-takes-all strategy of K-Means. With this new strategy, the association of the data points with the centers is distributed (like EM, but EM has certain problems, pointed out in Section 3.2/Fig. 2, that prevent it from reaching a good clustering) and the transition becomes continuous.

We also show that K-Harmonic Means has a "built-in" *dynamic weighting function*, which boosts the data that are not close to any center by giving them a higher weight in the next iteration. The word *dynamic* emphasizes the fact that the weighting function is automatically adjusted in each iteration. With these changes, the new algorithm is essentially insensitive to initialization, demonstrated by starting KHM with various initializations (good and bad) and comparing its convergence quality with KM or EM under the same initialization. We also present all three algorithms (KM KHM, EM) under a unified theoretical view (see the Section 3), which gives a more detailed explanation of KHM's insensitivity to initialization.

The rest of the paper is organized as follows: Section 2 introduces the generalized K-Harmonic Means clustering algorithms. It presents the KHM's performance function, $Perf_{KHM}$, its generalized form, $Perf_{KHMp}$, and the KHM_p algorithm. It also briefly discusses our implementation. Section 3 compares KHMp with KM and EM in detail. It presents a unified view of the three performance functions, as well as a framework for comparing the three algorithms. In particular, it shows how KHM_p uses a dynamic weighting function to boost data points that are not close to any centers in the next iteration. Section 4 compares the computational cost of KM, EM and KHMp.

Section 5 presents experimental results. Section 6 compares K-Means and K-Harmonic Means on a real-world high dimensional dataset. Section 7 concludes the paper.

2. The Generalized K-Harmonic Means Clustering Algorithm

2.1 Finding Clusters

Let M be a set of K centers. For the class of center-based clustering algorithms, including K-Means, K-Harmonic Means, and EM, the quality of the result is measured by the sum of a function $d(x, M)$ over all x,

$$Perf(X, M) = \sum_{x \in X} d(x, M).$$

We write down $d(x,M)$ for the following three algorithms:

K-Means: $d(x, M) = MIN\{||x - m||^2 | m \in M\}$;

K-Harmonic Means: $d(x, M) = HA\{||x - m||^2 | m \in M\} = |M| \Big/ \sum_{m \in M} \frac{1}{||x - m||^2}$;

EM: $d(x, M) = -\log(\sum_{m \in M} p_m * G_m(x))$ [1].

Intuitively, d(x,M) measures how well a particular data point x is taken care of by the set of centers. If a data point is close to one of the centers, it is well represented by the center it is close to.

2.2 The Nature of the Harmonic Average

The harmonic average of K numbers $\{a_1, \ldots\ldots, a_K\}$ is defined as

$$HA(\{a_1, \ldots\ldots, a_K\}) = K \Big/ \sum_{k=1}^{K} \frac{1}{a_k}.$$

The harmonic average is small if *one* of the a_k 's is small. Therefore, *HA()* behaves more like the *MIN()* function than an averaging function. This is the desired property we need for defining a performance function for measuring clustering quality, explained at the end of Section 2.1.

Fig. 1 has a plot of the harmonic average of two numbers *(x,y)* in *[0,10]x[0,10]* and comparing it with the *MIN(x,y)*. The plot of *HA()* is very similar to that of *MIN()*. More detailed mathematical comparisons of the two functions are possible based on their boundary conditions and their derivatives. Due to the limited length of this paper, we omit it.

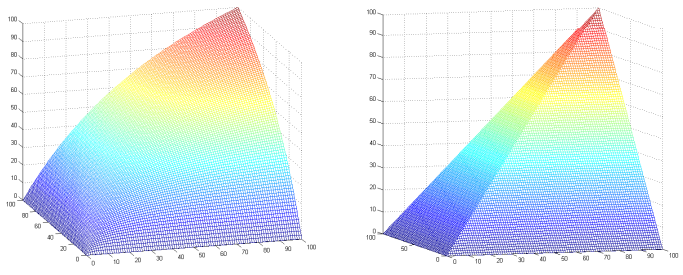

Fig. 1. The plots of *HA()* on the left and *MIN()* on the right.

[1] $G_m(x) \approx \exp(-||x - m||^2)$, the Gaussian density function. We limit the covariance matrix to be identity matrix in this paper for the consistent comparison with K-Means and K-Harmonic Means.

2.3 The Performance Function of K-Harmonic Means

K-Means' performance function is

$$Perf_{KM}(\{x_i\}_{i=1}^N, \{m_l\}_{l=1}^K) = \sum_{l=1}^{K} \sum_{x \in S_l} \| x - m_l \|^2, \tag{1}$$

where $S_l \subset X = \{x_i\}_{i=1}^N$ is the subset of *x's* that are closer to m_l than any other centers in $M = \{m_l\}_{l=1}^K$. (Or $\{S_l | l=1,...,K\}$ is the Voronoi partition). The double summation in (1) can be considered as a single summation over all *x* (data points) and the squared distance under the summations can be expressed by *MIN()*. Therefore, the KM performance function can be rewritten as[2]

$$Perf_{KM}(X, M) = \sum_{i=1}^{N} MIN\{\| x_i - m_l \|^2 | \, l = 1,..., K\}, \tag{2}$$

Replacing *MIN()* by *HA(),* we get the performance function of *KHM*:

$$Perf_{KHM\,p}(X, M) = \sum_{i=1}^{N} HA\{\| x_i - m_l \|^2 | \, l = 1,..., K\} = \sum_{i=1}^{N} \frac{K}{\sum_{l=1}^{K} \frac{1}{\| x_i - m_l \|^2}}. \tag{3}$$

The quantity inside the outer summation is the harmonic average of *K* squared distances $\{||x - m_l||^2 \mid l = 1,...,K\}$.

A unified view of the KM', KHM' and EM's performance functions is given later in Section 4, in which all are considered as ways of mixing bell-shape functions.

2.4 The KHM_p Performance Function and the KHM_p Algorithm

Using a general distance function, *d(x,m),* the most general form of K-Harmonic Means performance function is

$$Perf_{KHM}(X, M) = \sum_{i=1}^{N} HA\{d(x_i, m_l) | \, l = 1,..., K\} = \sum_{i=1}^{N} \frac{K}{\sum_{l=1}^{K} \frac{1}{d(x_i, m_l)}}.$$

In our earlier version of K-Harmonic Means paper [ZDH00], we presented KHM for $d(x,m) = ||x-m||^2$, which does not have the desired weighting function (for weighting functions, see Section 3.3) in its recursive optimization algorithm.

In this paper, we show that the desired weighting function can be derived theoretically by using the *p*th power of the L^2-distance as *d(x,m)* in KHM_p[3].

The performance function of KHM_p is defined by:

$$Perf_{KHMp}(X, M) = \sum_{i=1}^{N} HA\{\| x_i - m_l \|^p | \, l = 1,..., K\} = \sum_{i=1}^{N} \frac{K}{\sum_{l=1}^{K} \frac{1}{\| x_i - m_l \|^p}}. \tag{4}$$

To derive an algorithm for minimizing the KHM_p performance function, we take partial derivatives of the KHM's performance function (4) with respect to the center positions m_k, $k=1,...,K$, and set them to zero,

$$\frac{\partial Perf_{KHM}(X, M)}{\partial m_k} = -K \sum_{i=1}^{N} \frac{p(x_i - m_k)}{d_{i,k}^{p+2} (\sum_{l=1}^{K} \frac{1}{d_{i,l}^{p}})^2} = \vec{0} \tag{5}$$

where $d_{i,l} = ||x_i - m_l||$ on the right of (5) are still functions of the centers. "Solving" m_k's from the last set of equations, we get a recursive formula:

$$m_k = \sum_{i=1}^{N} \frac{1}{d_{i,k}^{p+2} (\sum_{l=1}^{K} \frac{1}{d_{i,l}^{p}})^2} x_i \Bigg/ \sum_{i=1}^{N} \frac{1}{d_{i,k}^{p+2} (\sum_{l=1}^{K} \frac{1}{d_{i,l}^{p}})^2}. \tag{6}$$

---

[2] For K-Means, the centroids are the optimal locations of the centers of a given partition only if the distance function is L^2. K-Means can be generalize to L^p space but the center locations will not be the centroids.

[3] We could also use this distance function in K-Means. In this new version of K-Means, the optimal centers will no longer be the centroids. But this replacement does not introduce a weighting function in K-Means.

This is the recursive formula for minimizing the KHM_p's performance function. KHM_p, like KM and EM, is a also center-based, iterative algorithm that refines the clusters defined by the K centers. Starting with a set of initial positions of the centers, KHM_p calculates $d_{i,l} = ||x_i - m_l||$, and then the new positions of the centers from (6) or from the decomposed sequence below,

$$\alpha_i = 1/(\sum_{l=1}^{K} \frac{1}{d_{i,l}^{p}})^2, \quad q_{i,k} = \frac{\alpha_i}{d_{i,k}^{\,p+2}}, \quad q_k = \sum_{i=1}^{N} q_{i,k}, \quad p_{i,k} = \frac{q_{i,k}}{q_k}, \quad m_k = \sum_{i=1}^{N} p_{i,k} x_i. \tag{7.1-7.5}$$

The recursion is continued until the performance value stabilizes.

The calculation of $q_{i,k}$'s (combination of (7.1) and (7.2)) can be done as follows:

$$q_{i,k} = \frac{d_{i,\min}^{2p}}{d_{i,l}^{p+2}[1+\sum_{l\neq\min}(\frac{d_{i,\min}}{d_{i,l}})^p]^2} = \frac{d_{i,\min}^{p-2}(\frac{d_{i,\min}}{d_{i,k}})^{p+2}}{[1+\sum_{l\neq\min}(\frac{d_{i,\min}}{d_{i,l}})^p]^2} \tag{7.6}$$

where $d_{i,\min} = MIN(d_{i,l} \mid l = 1,..., K)$. All the ratios $(d_{i,min}/d_{i,l})$ are in [0,1].

We have implemented KHM_p in several different languages -- C, Matlab and the parallel programming language ZPL (KHM_p has been run on multiple processors in parallel. See [FZ00]). We have tested KHM_p on hundreds of randomly generated datasets without encountering any numerical difficulties.

3. A Unified Analysis of K-Means, EM and K-Harmonic Means

3.1 The EM Clustering Algorithm Based on Linear Mixing of Gaussian Distributions

We briefly review a version of the EM algorithm needed later for the comparison with KHM_p and KM. We limit ourselves to the EM algorithm with linear mixing of K identical spherical bell-shape (Gaussian distribution) functions. Let

$$Perf_{EM}(X, M) = -\log\left\{\prod_{i=1}^{N}[\sum_{l=1}^{K} p_l * \frac{1}{(\sqrt{\pi})^D} EXP(-\| x_i - m_l \|^2)]\right\}, \tag{8}$$

be a linear mixing of K identical spherical bell-shape functions. EM algorithm is a recursive algorithm with the following two steps:

E-Step:
$$p(m_l \mid x_i) = \frac{p(x_i \mid m_l) * p(m_l)}{\sum_{i=1}^{N} p(x_i \mid m_l) * p(m_l)}, \tag{9}$$

where $p(x|m)$ is the prior probability with Gaussian distribution, $p(m_l)$ is the mixing probability.

M-Step:
$$m_l = \frac{\sum_{i=1}^{N} p(m_l \mid x_i) * x_i}{\sum_{i=1}^{N} p(m_l \mid x_i)}, \quad and \quad p(m_l) = \frac{1}{N}\sum_{i=1}^{N} p(m_l \mid x_i), \tag{10) and (11)}$$

where N is the size of the whole data set. For more details, see [MK97] and the references there.

3.2 A Unified View of The Three Performance Functions

Without introducing any change, applying the identity mapping $-log(EXP(-(\ )))$ to the performance functions of KM and KHM, we get

$$Perf_{KM}(X,M) = -\log\left\{\prod_{i=1}^{N} EXP\left[-MIN\left(\| x-m \|^2 | m \in M\right)\right]\right\}; \quad (12)$$

$$Perf_{KHM}(X,M) = -\log\left\{\prod_{i=1}^{N} EXP\left[-HA\left(\| x_i-m \|^p | m \in M\right)\right]\right\}. \quad (13)$$

Now they share the same form of the EM's performance function:

$$Perf_{EM}(X,M) = -\log\left\{\prod_{i=1}^{N}\left[\sum_{l=1}^{K} p_l * \frac{1}{\left(\sqrt{\pi}\right)^D} EXP\left(-\| x_i-m_l \|^2\right)\right]\right\}. \quad (14)$$

If we remove the negative sign in front of the "log", all three algorithms can be considered as maximizing the log-likelihood functions except that, for the K-Means and K-Harmonic Means, the functions under the products are not normalized to be probability density functions. Therefore, K-Means and K-Harmonic Means are not exactly EM-type of algorithms.

The quantity inside the brackets "[]" in (14) is the linear mixing of the bell-shape functions – the *EXP()*'s. We can also look at the performance functions of KM and KHM as mixings of bell-shape functions.

Define

MIN Mixing: $$EXP\left[-MIN\left(\| x-m_l \|^2 | l=1,...,K\right)\right] \quad (15)$$

Min-Mixing can also be called Max-Mixing because *EXP(-x)* is monotone decreasing and

$$EXP\left[-MIN\left(\| x-m_l \|^2 | l=1,...,K\right)\right] = MAX\left[EXP\left(-\| x-m_l \|^2 | l=1,...,K\right)\right]. \quad (16)$$

Harmonic Mixing: $$EXP\left[-HA\left(\| x-m_l \|^p | l=1,...,K\right)\right] \quad (17)$$

Linear Mixing: $$\sum_{l=1}^{K} p_l * \frac{1}{\left(\sqrt{\pi}\right)^D} EXP(-\| x-m_l \|^2) \quad (18)$$

We use one dimensional data and two bell-shape functions to illustrate the differences among the three kind of mixings in **Fig. 2**. These are exactly the plots of *exp(-d(x,M))* (See Section 2.1).

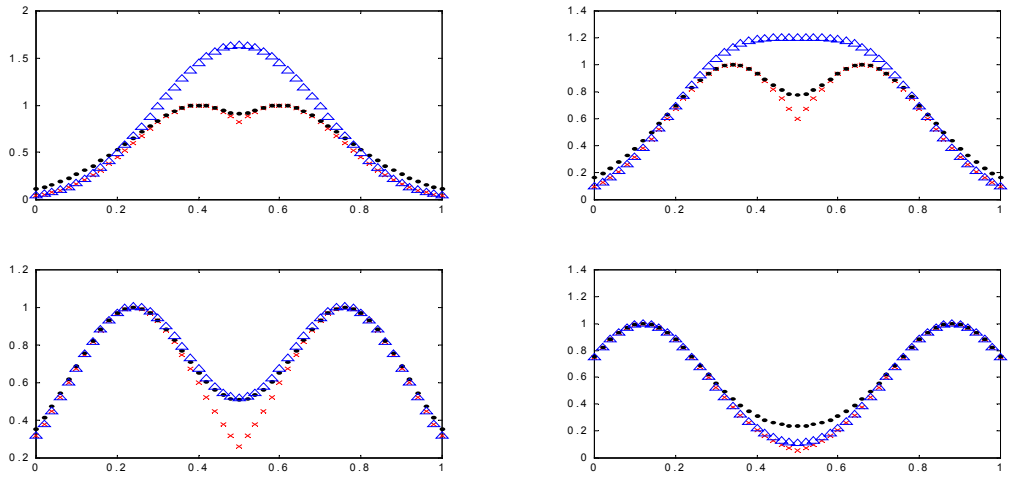

Fig. 2. Comparing Three Different Mixings of Two One-dimensional Bell-shape Functions. As the centers move from near to far, the differences among the three mixings decrease. Legend: 'x' — KM, '.' – KHM, and 'triangle' – EM.

Under linear mixing, when the centers are too close to each other, two or more bell-shape functions merge into a single peak. The maximum of that single peak behaves like a ghost center and the individual centers lose their

identity. For finding clusters, the linear mixing (the limited version with fixed covariance matrix) does not behave properly. This was clearly shown in our experimental results published in [ZDH00].

3.3 A Unified View of Three Algorithms and the Dynamic Weighting Function

We compared three performance functions in the last section. In this section, we compare all three algorithms. All three algorithms take the following form,

$$m_l = \frac{\sum_{i=1}^{N} p(m_l \mid x_i) * a(x_i) x_i}{\sum_{i=1}^{N} p(m_l \mid x_i) * a(x_i)}, \quad a(x) > 0, \quad p(m_l \mid x_i) \geq 0 \quad and \quad \sum_{l=1}^{K} p(m_l \mid x_i) = 1. \tag{19.1-19.4}$$

a(x), the dynamic weighting function of the data points, decides how much of each data point $\boldsymbol{x}$ participates in the next iteration of calculating the new center locations. $p(m_l \mid x_i)$, the "membership" functions", decides the portion of $a(x_i)*x_i$ that is associated with m_l.

For a given algorithm of the following type, which covers KM, KHM and EM:

$$m_l = \sum_{i=1}^{N} c_{l,i} x_i \Big/ \sum_{i=1}^{N} c_{l,i}, \tag{20}$$

the membership function is calculated by normalizing $c_{l,i}$ over l so that (19.4) is satisfied:

$$p(m_l \mid x_i) = c_{l,i} \Big/ \sum_{l=1}^{K} c_{l,i}, \tag{21}$$

and the weighting function is simply

$$a(x) = \sum_{l=1}^{K} c_{l,i}. \tag{22}$$

For K-Means, each data point belongs to the closest center 100% (winner-takes-all). Therefore, $p(m_l|x_i)=1$ if m_l is the closest center to x_i, otherwise *0*. The weighting function *a(x) =1,* for all data points in all iterations.

For EM: the membership function is derived from Bayes' rule. Let $p(x_i|\ m_l)$ be the lth Gaussian density function and $p(m_l)$ the weight of $p(x_i|\ m_l)$ (See (8) and (9) in Section 3.1). The membership function is

$$p(m_l \mid x_i) = \frac{p(x_i \mid m_l) * p(m_l)}{\sum_{i=1}^{N} p(x_i \mid m_l) * p(m_l)}.$$

Since the membership function is already normalized (satisfying (19.4)), the weighting function *a(x) =1*.

For K-Harmonic Means, the iterative procedure (6) (in Section 2.4) can be written as

$$\vec{m}_k = \frac{\sum_{i=1}^{N} \frac{1}{d_{i,k}^{\,p+2} (\sum_{l=1}^{K} \frac{1}{d_{i,l}^{\,p}})^2} \vec{x}_i}{\sum_{i=1}^{N} \frac{1}{d_{i,k}^{\,p+2} (\sum_{l=1}^{K} \frac{1}{d_{i,l}^{\,p}})^2}} = \frac{\sum_{i=1}^{N} \frac{\frac{1}{d_{i,k}^{\,p+2}}}{\sum_{l=1}^{K} \frac{1}{d_{i,l}^{\,p+2}}} * \frac{\sum_{l=1}^{K} \frac{1}{d_{i,l}^{\,p+2}}}{(\sum_{l=1}^{K} \frac{1}{d_{i,l}^{\,p}})^2} * \vec{x}_i}{\sum_{i=1}^{N} \frac{\frac{1}{d_{i,k}^{\,p+2}}}{\sum_{l=1}^{K} \frac{1}{d_{i,l}^{\,p+2}}} * \frac{\sum_{l=1}^{K} \frac{1}{d_{i,l}^{\,p+2}}}{(\sum_{l=1}^{K} \frac{1}{d_{i,l}^{\,p}})^2}} \tag{23}$$

where $d_{i,k}=||x_i-m_k||$. For KHM_p, we have

$$p(m_k \mid x_i)=\frac{\frac{1}{d_{i,k}^{p+2}}}{\sum_{l=1}^{K}\frac{1}{d_{i,k}^{p+2}}} \quad and \quad a_p(x)=\frac{\sum_{l=1}^{K}\frac{1}{\|x-m_l\|^{p+2}}}{[\sum_{l=1}^{K}\frac{1}{\|x-m_l\|^{p}}]^2}. \tag{24}$$

As a data point x is approached by a center m, the weighting function satisfies $a_p(x)=O(||x-m||^{p-2})$ near m where m is the center closest to x (See **Fig. 3**). For $p>2$, $a_p(x)$ has a smaller value for the data points that are closer to one of the centers. This property serves as a dynamic weighting function. It boosts, in the next iteration, the participation of the data points that are not close to any centers. The more centers are near a data point the smaller the weight for that data point. This has the effect of flattening out a local density that trapped more than one centers and reduces the chance of multiple centers being trapped by a single local cluster of data. Based on the fact that the weight of each data point in the calculation of the center locations in the next iteration depends on the current location of the centers, we call this dynamic weighting of the data points. This is the most important difference between K-Harmonic Means and K-Means or EM. **Fig. 3** has a plot of the weighting functions for $p=2, 2.5, 3, 3.5, 4$. For $p=2$, in **Fig. 3**, the weighting function does not have the desired boosting behavior. The weight of the data points that are close to one of the centers is not lowered. This explains the experimental results presented in Section 5.

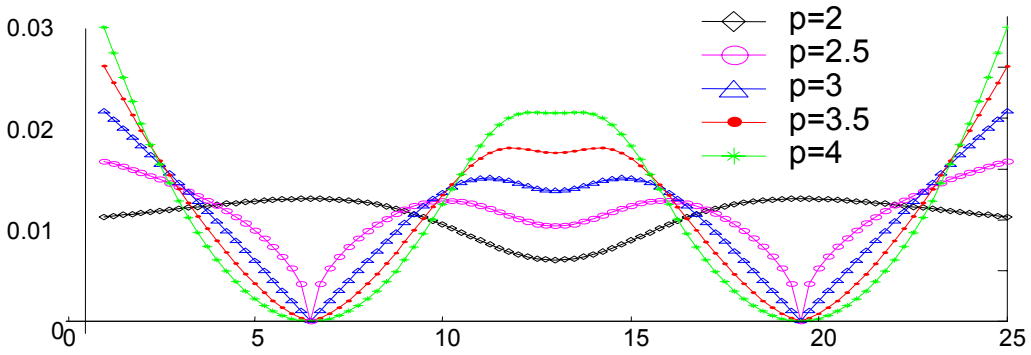

Fig. 3. A Plot of $a(x)$ for K-Harmonic Means with two centers in one-dimensional space. The two centers are located at 6.5 and 19.5.

4. Computational Costs in Each Iteration

In each iteration, calculating all the pair-wise distances from N data points to K centers (of D dimensional vectors) costs $O(N*K*D)$. KM and EM (linear mixing) share the same cost on this part. After getting the coefficients $p_{i,k}$, calculating the linear combinations $m_k = \sum p_{i,k}*x_i$ costs another $O(N*K*D)$. EM costs the same on this part. KM costs less ($O(N*D)$) on this due to the partitioning but an additional $O(N*K)$ comparisons and assignments (marking) are used to do the partitioning. After calculating the distances, all quantities used in the algorithm no longer depend on the dimension and all other costs are $O(N*K)$. The leading asymptotic term for all three algorithms are the same $O(N*K*D)$.

The asymptotic computational complexity *per iteration* for KM, KHM and EM (linear mixing model) are all $O(N*K*D)$. For all three algorithms, since the costs are dominated (especially for high dimensional data) by the distance calculations $||x_i-m_k||$, and there are exactly the same number of distances to be calculated, the coefficients of the cost term $N*K*D$ of all three algorithms are very close. It is the convergence rate and the convergence quality that differentiate them in real world applications[4].

[4] Due to the partitioning nature, faster algorithms/implementations have been designed for KM using trees to do spatial partition of either the centers or the data [GG92],[PM99].

Space complexity of KHM is *NxD* for data points, *KxD* for the *K* centers and *KxD+2*K* for temporary storage. The temporary storage requirement tends to be lower than KM because the later needs a *O(N)* temporary storage to keep the membership information and *N>>K* in real problems.

For low dimensional data (D=2), KHM and EM are slower than K-Means. This difference disappears quickly as *D* increases.

5. Experimental Results

A number of experimental results on a special version of KHM (*p=2* plus heuristics) have been published in [ZDH00]. In this paper, we focus on the experimental results on KHM_p for different *p* values and comparing them with K-Means and EM.

We experimented with seven algorithms: KHM_p with *p=2, 2.5, 3, 3.5, 4*, K-Means and EM. For each algorithm, 3 different type of initializations are used:

Type-1: Very bad -- all 50 centers are initialized to be within a small region relative to the data.

Type-2: Bad -- all centers are randomly initialized, not related to the data. The centers have a bigger spread than the data itself.

Type-3: Better -- the centers are initialized to 50 randomly chosen data points from the dataset.

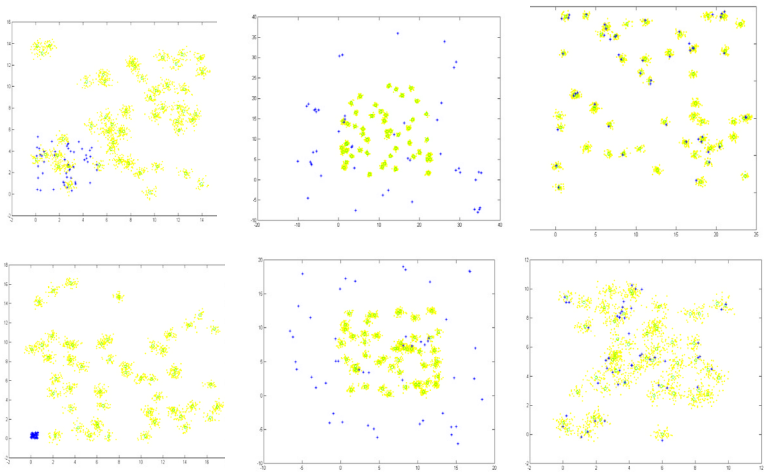

Fig. 4 Two samples of each type of initializations. Ordered from left to right: Type-1, Type-2, Type-3. The light backgrounds are the data points and the dark dots are the initial center locations.

We randomly generated 100 datasets, *Dataset(i), i=1, ..., 100*. For each dataset, we randomly generated three initializations, one of each type, *Init(i,j), j=1, 2, 3*, for all *i*. We ran all seven algorithms on each pair, *(Dataset(i), Init(i,j))*. A total of *100*3*7=2100* experiments were conducted.

The following function, with *N=2500, K=50, D=2,* is used to generate all the datasets. *r* is randomly generated between *10* and *30*.

```
function [dataset,centers] = ClusGen(K, N, D, r)
% K = # clusters, N = #data points, D = dimensionality, r = within cluster variance/inter-cluster variance.
%Step 1: Generate cluster centers.
    centers = r * rand(K,D);    % K center locations are generated and scaled up by the factor r.
% Step 2: Generate the random sizes of the K clusters.
    s = 2*rand(K,1)+1;      s = round(N*s/sum(s));      N1 = sum(s);      diff = abs(N-N1);
    s(1:diff) = s(1:diff) + sign(N-N1);  % adjust the size so that they add up to N.
 %Step 3: Generate clusters one-by-one.
    for k=1:K
        cluster = randn(s(k),D); % normal distribution.
        % move the clusters to the kth center location.
```

```
        mean = sum(cluster)/s(k);        Sk = repmat(centers(k,:)-mean,s(k),1)+cluster;
        % merge the cluster into the dataset.
        dataset = [dataset' Sk']';
    end;  % of for loop.          %End of the cluster dataset generator.
```

To compare the results from seven different algorithms, we need a common measure. We used the *square-root* of the K-Means performance function to measure the quality of the clusters derived by all seven algorithms,

$$Perf(Dataset, Centers) = \sqrt{Perf_{KM}(Dataset, Centers)}. \tag{25}$$

We chose K-Means' performance function because it is more popular and simpler than others. We took the square-root of it because the original K-Means performance is quadratic, which makes the bad results look a lot worse than they really are[5]. For the phrase – "the performance is with-in 2*optimum" to make sense, we need a linear measure.

Measurements are compared with (divided by) the optimal performance, which is also measured by the square-root of the K-Means performance function and is derived by running K-Means on the location of the "true" centers of the clusters returned by the ClusGen() function[6]. The average values and the coefficient of standard deviation of *the ratio between the actual performance value and the optimal performance value* are given in Table 1. Formulas in (26) give the details of the calculations:

$$ratio_i = \frac{Perf(Dataset(i), Centers)}{Optimum(Dataset(i))}, \quad avg = \frac{\sum_{i=1}^{100} ratio_i}{100}, \quad \sigma^2 = \frac{\sum_{i=1}^{100} (ratio_i - avg)^2}{100} \quad and \quad \theta = \frac{\sigma}{avg}. \tag{26}$$

Formulas in (26) are applied to the performance measure of each algorithm under each type of initializations. These average values are also plotted in **Fig. 5** and **Fig. 6. Fig. 5** shows the sensitivity of the average performance ratio over the optimal performance of each algorithm to the initialization of the centers. **Fig. 6** shows the performance ratio over optimal for just the Type 3 initialization with improved resolution along the vertical axis. *When the global optimal performance is achieved, the ratio will be 1.* From Table 1, we see that KHM_p performs best at *p=3.5* (for two dimensional data). No matter what type initialization is used, KHM(p>=2.5) always out perform KM and EM, measured by K-Means performance function. This is not only true on average but also true on most individual experiments. To give a more detailed picture, we also plot the results from individual runs for KM, EM and KHM(p=3.5) in **Fig. 7**. The 100 experiments are listed along the horizontal axis. The ratios of performance value over the optimum are along the vertical axis.

It is not surprising to see that KHM(p=2) does not perform better than K-Means on good initializations (see **Fig. 6**) because the desired dynamic weighting function is not in KHM(p=2) (See **Fig. 4** and the explanation there). However, KHM(p=2) is still much more insensitive to initializations than K-Means and EM (see **Fig. 5**). The dynamic weighting function explains only partially the insensitivity of KHM_p to initialization. The insensitivity of KHM(p=2) to initialization has to be explained by other properties of the K-Harmonic Means function and its minimization algorithms.

In general, the higher the dimensionality, the larger *p* value is desired. We have to omit the details on high dimensional datasets due to the limited length of this paper. A separate paper for clustering high dimensional datasets is in progress.

We run 40 iterations, which are sufficient, for all seven algorithms on all datasets and initializations. It is difficult to plot all the convergence curves because there are such a large number of them. We plot the average "convergence speed" over 100 experiments under each type of initialization in **Fig. 8**. In each figure, the horizontal axis is the number of iterations; the vertical axis is the average ratio (over 100 experiments) between the actual performance value and the global optimum, both measured by the K-Means performance function. In general, K-Means converges faster than KHM under good initializations and KHM converges faster than K-Means under bad initializations. Some of the EM's convergence curves are not monotone decreasing because its performance is measured by K-Means performance function instead of EM's. The same thing could also happen to KHM's convergence curves.

---

[5] For example, if the deviation of the centers from they optimal locations is doubled, the performance value is quadrupled.

[6] which is, of course, only an approximation of the true global optimum.

Table 1. The average and the coef. of std. dev. of the ratios between actual performance values and the global optimum (under K-Means), all measured by the K-Means performance function.

	type 1 initialization		type 2 initialization		type 3 initialization	
	Avg.	*Coef. Std. Dev.*	*Avg.*	*Coef. Std. Dev.*	*Avg.*	*Coef. Std. Dev.*
EM	4.6091	0.5452	3.1050	0.4238	1.6995	0.2436
KM	3.7222	0.4822	2.8191	0.3388	1.3765	0.2198
KHM 2.0	1.6143	0.2934	1.3379	0.1815	1.5810	0.2781
KHM 2.5	1.1709	0.1170	1.1607	0.1097	1.2264	0.1452
KHM 3.0	1.1409	0.1127	1.1463	0.1091	1.1563	0.1098
KHM 3.5	1.1500	0.1077	1.1515	0.1127	1.1395	0.1066
KHM 4.0	1.2343	0.1673	1.1896	0.1455	1.2028	0.1562

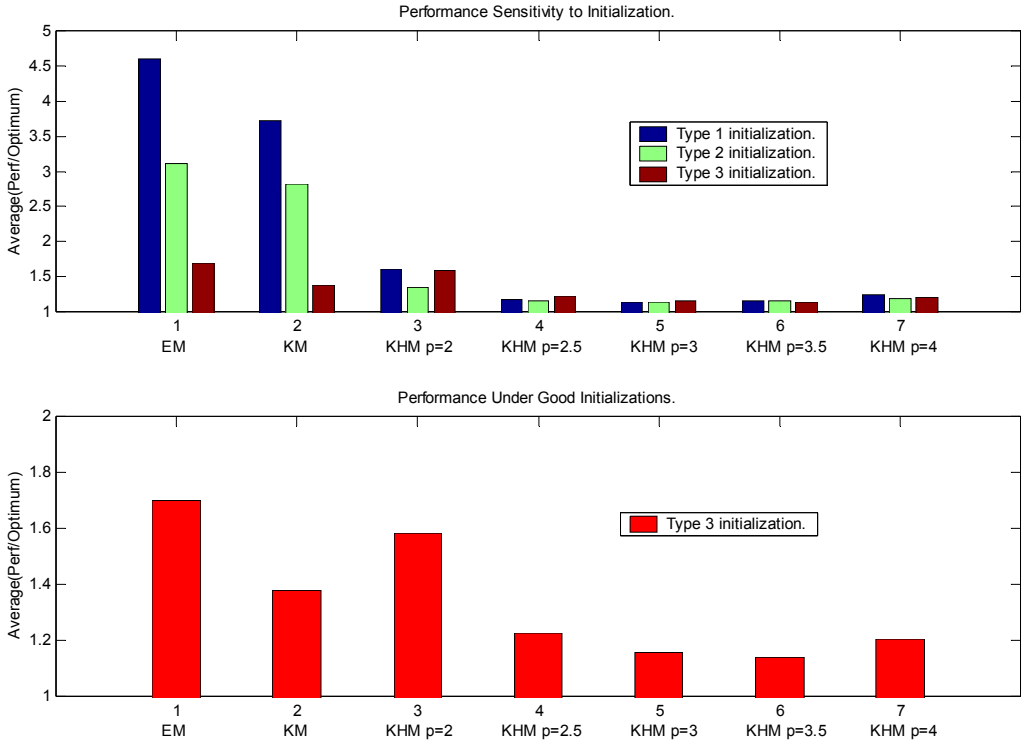

Fig. 5 (top) Robustness of KHM: KHM_p is largely insensitive to initializations while EM and KM are very sensitive.

Fig. 6 (bottom) Performance of KHM: Even with careful initializations, KHM with p>=2.5 (I.e., when desirable dynamic weighting is in effect) consistently performs better than EM and KM. The average ratio of the performance values over the global optima are averaged over 100 different datasets. For p=3 and 3.5, the performance of KHM_p are about 1.2 times optimum on average on the 100 datasets.

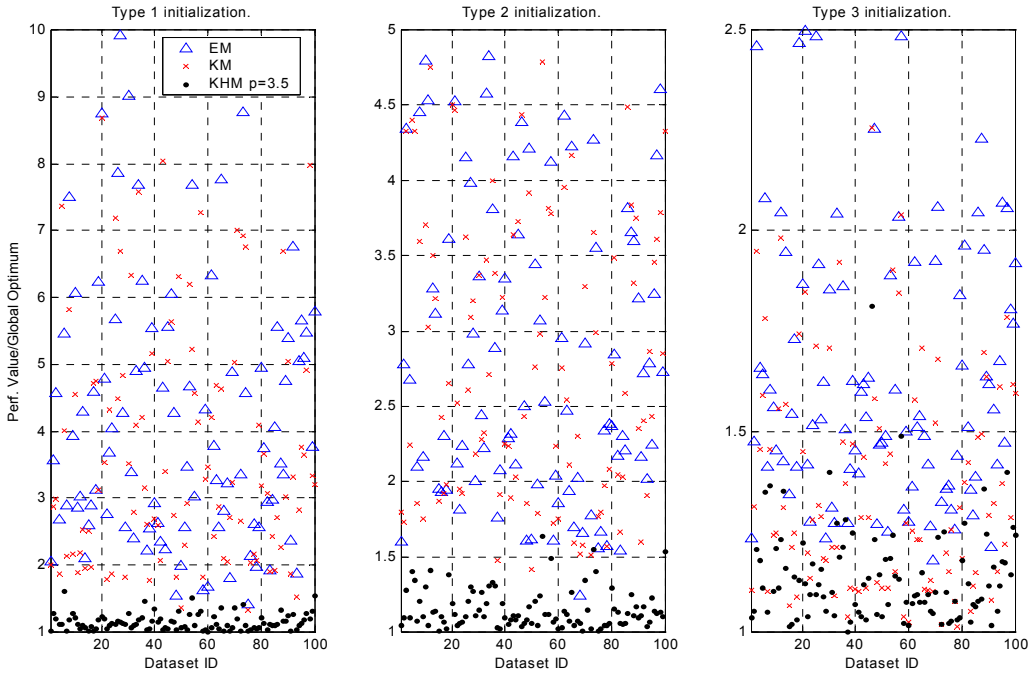

Fig. 7 The ratios of performance values over the optimum from 100 individual runs are plotted here without averaging. KHM not only out-perform KM and EM on average, but also on most individual runs especially under bad initializations. To avoid over-crowding, only KHM(p=3.5) is plotted. For other $p>2$, the results are similar. The ranges of the vertical scale of the plots are different but they all start at 1.

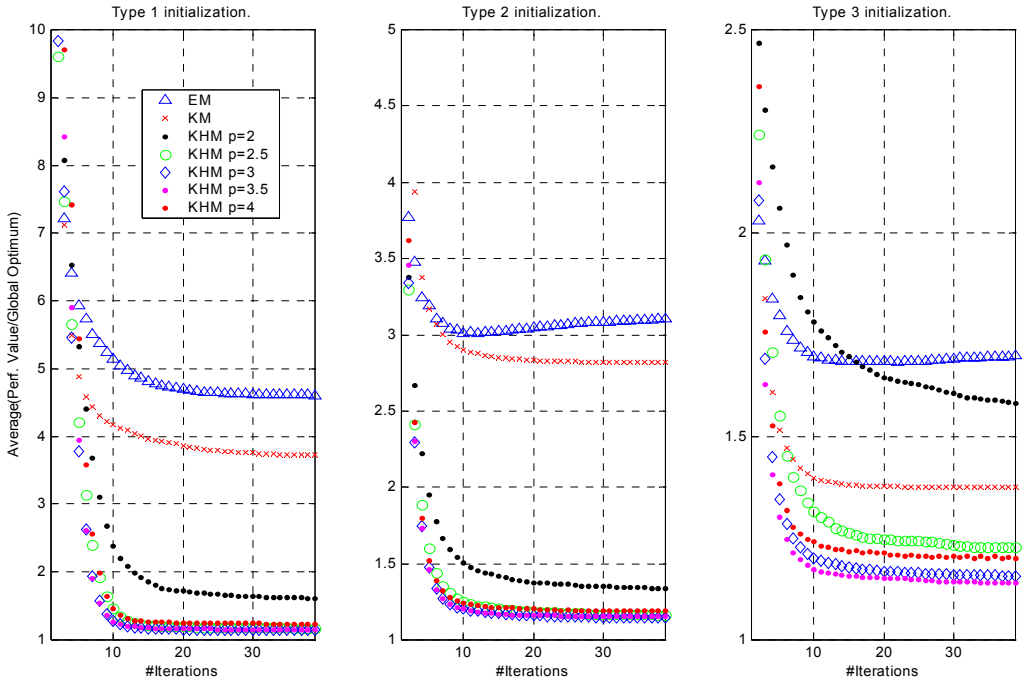

Fig. 8 The convergence speed of KM, EM and KHM, averaged over 100 individual runs are plotted. In general KM converges faster under good initializations and KHM converges faster under bad initializations. The overall rate is about the same. The theoretical asymptotic convergence rate of KHM is not known. All three plots share the same legend but the vertical scales of them are different.

6. Application on Real-World Datasets

The KHM algorithm has been applied to a few real-world datasets. We present the results on the dataset from the 1998 KDD-CUP data mining contest here. The dataset contains information about people who have made charitable donation in response to direct mailing requests. The dataset contains 95412 records, each of which has 481 fields. This dataset has been reduced, by Farnstrom et al. [FLE00], to 56 dimensions and coded as a real-valued vectors (18 of them are binary). To give equal weight to all features, each feature is normalized to zero mean and variance of one. The binary version of the data is 21.4 Mbytes.

To run KHM on large datasets, we implemented both K-Means and Generalized KHM in C. Compiled by Microsoft C++ compiler and run under Windows NT 4.0 on a HP Kayak XU PC with a 733 MHz Intel processor.

We run both K-Means and KHM (p=6) on the dataset starting from the same random initialization of the centers. The experiment is replicated 36 times with different random initializations. Each replicate is run for up to 100 iterations which is sufficient for both algorithms to convergence. The quality of the clustering results for both K-Means and KHM are measured by the K-Means performance function. The mean and standard deviation of the results over the 36 replications are shown in Table 2. KHM performance slightly better than K-Means. Due to KHM's insensitivity to initialization, the standard deviation from KHM (p=6) is about one quarter of that of K-Means. The results of the 36 individual runs are plotted in Fig. 9.

Table 2. The mean and variance of the performance.

	KM	KHM
Mean	35669.58	34561.76
Std.	860.6185	207.7973

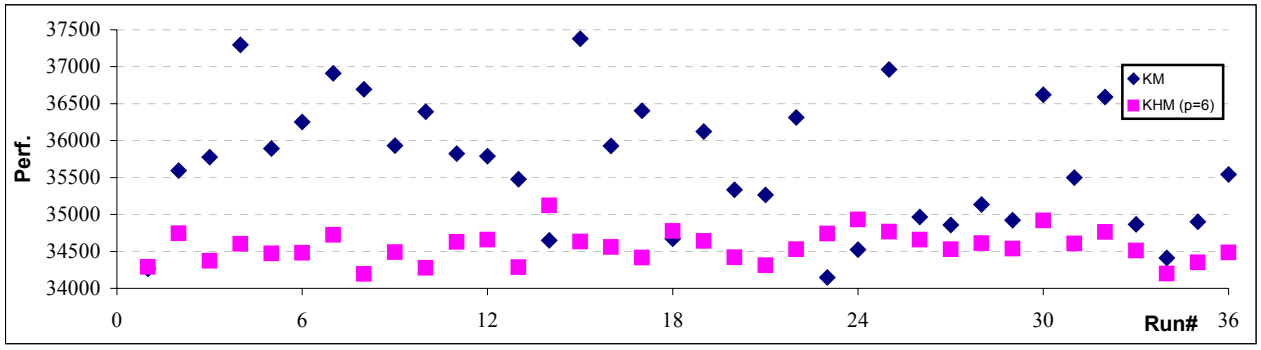

Fig 9. Comparison of KM and KHM over 36 runs on the 1998 KDD-CUP Contest dataset.

7. Conclusions and Future Work

We discovered, through developing the K-Harmonic Means clustering algorithms, that the dynamic weighting of data helps clustering algorithm escape certain local optima and converge to a better local optimum, which contributed to the insensitivity of KHM_p to the initialization of the centers. This is clearly demonstrated through the comparison of KHM_p (p>2) with K-Means, EM and KHM(p=2). The concept of dynamic weighting of data is similar, in certain ways, to the boosting concept in supervised learning but the details on the weight calculation and aggregation of the results are different. We focused our comparisons of KHM with the most popular algorithms – K-Means and EM in this paper. We emphasize that KHM optimizes its own performance function instead of K-Means', even though, it does do better job on optimizing K-Means' performance than K-Means itself in many cases.

Acknowledgements

We like to thank Dr. Meichun Hsu and Dr. Charles Elkan for their comments on the KHM algorithm. We also like to thank Meichun for her suggestions on the presentation of the experimental results, which significantly improved the clarity of this paper, and her proofreading of the manuscript.

References

[A73] Anderberg, M. R. 1973. Cluster analysis for applications. Academic Press, New York. xiii + 35p.

[B99] Bay, S. D. (1999). The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science.

[BFR98] Bradley, P., Fayyad, U. M., and Reina, C.A., "Scaling EM Clustering to Large Databases," MS Technical Report, 1998.

[BF98] Bradley, P., Fayyad, U. M., C.A., "Refining Initial Points for KM Clustering", MS Technical Report MSR-TR-98-36, May 1998.

[BFR98a]Bradley, P.,Fayyad, U.M., and Reina, C.A., "Scaling Clustering to Large Databases", KDD98, 1998.

[DH72] Duda, R., Hart, P., "Pattern Classification and Scene Analysis", John Wiley & Sons, 1972.

[DLR77]Dempster, A. P., Laird, N.M., and Rubin, D.B., "Miximum Likelyhood from Incomplete Data via the EM Algorithm", Journal of the Royal Statistical Society, Series B, 39(1):1-38, 1977.

[F95] Fritzke, B. , "A growing neural gas network learns topologies", In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 625-632. MIT Press, Cambridge MA, 1995a.

[FLE00] Farnstrom, F., Lewis, J. and Elkan, C., "True Scalability for Clustering Algorithms," SIGKDD Explorations, Vol. 2, Issue 1, July 2000.

[FPU96]Fayyad, U. M., Piatetsky-Shapiro, G. Smyth, P. and Uthurusamy, R., "Advances in Knowledge Discovery and Data Mining", AAAI Press 1996

[FZ00] Forman, G. & Zhang, B., "Linear Speedup for a parallel non-approximating recasting of center-based clustering algorithms, including K-Means, K-Harmonic Means and EM," KDD-2000 Workshop on Distributed and Parallel Knowledge Discovery, Sixth ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August, 2000, Boston, USA.

[GG92] Gersho & Gray, "Vector Quantization and Signal Compression", KAP, 1992

[GMW85] Gill, P.E., Murray, W. and Wright, H.M., "Practical Optimization", Academic Press, 1981.

[G85] Gonzales,T.F,"Clustering to Minimize the Maximum Intercluster Distance",Theo.Comp.Sci.38, p293-306, 85.

[KR90] Kaufman, L. and Rousseeuw, P. J., "Finding Groups in Data : An Introduction to Cluster Analysis", John Wiley & Sons, 1990.

[M67] MacQueen, J. 1967. Some methods for classification and analysis of multivariate observations. Pp. 281-297 *in*: L. M. Le Cam & J. Neyman [eds.] Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Vol. 1. University of California Press, Berkeley. xvii + 666 p.

[MA] McKenzie, P. and Alder, M., "Initializing the EM Algorithm for Use in Gaussian Mixture Modeling", The Univ. of Western Australia, Center for Information Processing Systems, Manuscript.

[MK97]McLachlan, G. J. and Krishnan, T., "The EM Algorithm and Extensions.", John Wiley & Sons, Inc., 1997

[PM99] Pelleg, D. and Moore, A, "Accelerating Exact K-Means Algorithms with Geometric Reasoning", KDD-99, Proc. of the Fifth ACM SIGKDD Intern. Conf. On Knowledge Discovery and Data Mining, page 277-281.

[RW84]Rendner, R.A. and Walker, H.F., "Mixture Densities, Maximum Likelihood and The EM Algorithm", SIAM Review, vol. 26 #2, 1984.

[SI84] Selim, S.Z. and Ismail, M.A., "*K*-Means-Type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality", IEEE Trans. On PAMI-6, #1, 1984.

[ZHD00] Zhang, B., Hsu, M., Dayal, U., "K-Harmonic Means", International Workshop on Temporal, Spatial and Spatio-Temporal Data Mining, TSDM2000, Lyon, France Sept. 12, 2000.

[ZHF00] Zhang, B., Hsu, M., Forman, G., "Accurate Recasting of Parameter Estimation Algorithms using Sufficient Statistics for Efficient Parallel Speed-up", PKDD 2000, Lyon, France, Lecture Notes in AI #1910, Springer, p243.