CHAPTER 1

Probabilistic Distance Clustering, Algorithm and Applications

C. Iyigun
RUTCOR–Rutgers Center for Operations Research, Rutgers University
640 Bartholomew Rd., Piscataway, NJ 08854-8003, USA
Email: iyigun@rutcor.rutgers.edu

A. Ben-Israel
RUTCOR–Rutgers Center for Operations Research, Rutgers University
640 Bartholomew Rd., Piscataway, NJ 08854-8003, USA
Email: adi.benisrael@gmail.com

The probabilistic distance clustering method (called PDQ method) of the authors,\textsuperscript{2,8} computes the cluster membership probabilities using the distances of the data points from the cluster centers, and the cluster sizes. The method is based on the joint distance function (JDF), a weighted harmonic mean of the above distances, that approximates the data by capturing the data points in its lowest contours. The cluster centers, and the cluster sizes (if not given), are updated as stationary points of the JDF. The PDQ method is described, and applied to clustering, location problems, and mixtures of distributions, where it is a viable alternative to the EM method. The JDF also helps to determine the correct number of clusters for a given data set.

1. Introduction

We take data points to be vectors \( \mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n \), and a dataset \( D \) consisting of \( N \) data points \( \{ \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \} \). A cluster is a set of data points that are similar, in some sense, and clustering is a process of partitioning a data set into disjoint clusters.

In distance clustering (or \( d\)-clustering), “similarity” is interpreted in terms of a distance function \( d(\mathbf{x}, \mathbf{y}) \) in \( \mathbb{R}^n \), such as

\[
d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|, \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n,
\]  

(1)
where \( \| \cdot \| \) is a norm. In particular, the Mahalanobis distance with the norm,

\[
\| \mathbf{u} \| = \langle \mathbf{u}, \Sigma^{-1} \mathbf{u} \rangle^{1/2},
\]

where \( \Sigma \) is the covariance matrix of the data involved.

**Example 1:** A data set in \( \mathbb{R}^2 \) with \( N = 1100 \) data points is shown in Figure 1. The data on the left was simulated from a normal distribution \( N(\mu, \Sigma) \), with:

\[
\mu_1 = (2, 0), \quad \Sigma_1 = \begin{pmatrix} 0.0005 & 0 \\ 0 & 0.05 \end{pmatrix}, \quad (100 \text{ points}),
\]

and the data on the right consist of 1000 points, simulated in a circle of diameter 1 centered at \( \mu_2 = (3, 0) \) according to a radially symmetric distribution with \( \text{Prob}\{\|x - \mu_2\| \leq r\} = 2r \).

This data will serve as illustration in Examples 3–7 below.

\[\text{Fig. 1. A data set in } \mathbb{R}^2\]

In \( d \)-clustering each point is typically classified to the cluster with the nearest center. After each assignment, the cluster centers may change, resulting in further re-classifications. A \( d \)-clustering algorithm will therefore
iterate between centers and re-assignments. The best known such method is the \(k\)-means clustering algorithm, see Hartigan.\(^5\)

In **probabilistic clustering** the cluster membership is expressed by probabilities \(p_k(x) = \text{Prob}\{x \in C_k\}\), that a data point \(x\) belongs to the cluster \(C_k\). In **probabilistic d–clustering** these probabilities depend on the relevant distances.

Probabilistic d–clustering adjusted for the cluster size \(q\), is called **probabilistic \(\{d,q\}\)–clustering**.

An algorithm for probabilistic \(\{d,q\}\)–clustering, called the PDQ Algorithm, is presented in \(\S\) 2. It is based on the **joint distance function**, see \(\S\) 2.2, a function that captures the data in its low level sets. The centers, and cluster sizes (if not given), are updated by the PDQ algorithm as stationary points of the joint distance function.

In \(\S\) 4 we apply the PDQ algorithm to estimation of the parameters of Gaussian mixtures, and report the results in \(\S\) 5.

The PDQ algorithm is also effective for solving capacitated multi-facility location problems, see \(\S\) 6.

Another application of the PDQ algorithm is for determining the correct number of clusters, see \(\S\) 7.

For other approaches to probabilistic clustering see the surveys in Höppner et al.,\(^7\), Tan et al.\(^14\). A unified optimization framework for clustering was given by Teboulle.\(^15\)

### 2. Probabilistic \(\{d,q\}\)–Clustering

Let a data set \(D \subset \mathbb{R}^n\) be partitioned into \(K\) clusters \(\{C_k : k = 1, \ldots , K\}\),

\[
D = \bigcup_{k=1}^{K} C_k
\]

and let \(c_k\) be the center (in some sense) of the cluster \(C_k\), and let \(q_k\) be the cluster size, which is known in some applications, and an unknown to be estimated in others. In what follows the cluster sizes, or their estimates, are assumed given wherever appearing in the right hand side of a formula.

With each data point \(x \in D\) and a cluster \(C_k\), we associate:

- a **distance** \(d_k(x) := d_k(x, c_k)\), also denoted \(d_k\) if \(x\) is understood,
- a **probability** of membership in \(C_k\), denoted \(p_k(x)\), or just \(p_k\).

In general, the distance functions \(d_k(\cdot)\) are different for different clusters. In particular, different Mahalanobis distances

\[
d_k(x) = \langle x - c_k, \Sigma_k^{-1}(x - c_k) \rangle^{1/2},
\]

\(\text{for } k = 1, \ldots , K\).
using estimates \( \{c_k, \Sigma_k\} \) for the cluster in question.

There are several ways to model the relationship between distances and probabilities.\(^2\) The following assumption is our basic principle.

**Principle.** For each \( x \in D \) and cluster \( C_k \), the probability \( p_k(x) \) satisfies

\[
p_k(x) \frac{d_k(x)}{q_k} = \text{constant}, \quad k = 1, \ldots, K.
\]

Cluster membership is thus more probable the closer the data point is to the cluster center and the bigger is the cluster.

### 2.1. Probabilities

From the above principle, and the fact that probabilities add to 1 we get

**Theorem 2:** Let the cluster centers \( \{c_1, c_2, \ldots, c_K\} \) be given, let \( x \) be a data point, and let \( \{d_k(x) : k = 1, \ldots, K\} \) be its distances from the given centers. Then the membership probabilities of \( x \) are

\[
p_k(x) = \frac{\prod_{j \neq k} d_j(x)}{\sum_{i=1}^{K} \prod_{j \neq i} d_j(x)/q_j}, \quad k = 1, \ldots, K.
\]

**Proof:** Using (5) we write for \( i, k \)

\[
p_i(x) = \left( \frac{p_k(x) d_k(x)}{q_k} \right) / \left( \frac{d_i(x)}{q_i} \right).
\]

Since \( \sum_{i=1}^{K} p_i(x) = 1 \),

\[
p_k(x) \sum_{i=1}^{K} \left( \frac{d_k(x)/q_k}{d_i(x)/q_i} \right) = 1.
\]

\[
\therefore \ p_k(x) = \frac{1}{\sum_{i=1}^{K} \left( \frac{d_k(x)/q_k}{d_i(x)/q_i} \right)} = \frac{\prod_{j \neq k} d_j(x)/q_j}{\sum_{i=1}^{K} \prod_{j \neq i} d_j(x)/q_j}.
\]

In particular, for \( K=2 \),

\[
p_1(x) = \frac{d_2(x)/q_2}{d_1(x)/q_1 + d_2(x)/q_2}, \quad p_2(x) = \frac{d_1(x)/q_1}{d_1(x)/q_1 + d_2(x)/q_2},
\]

and for \( K = 3 \),

\[
p_1(x) = \frac{d_2(x)d_3(x)/q_2q_3}{d_1(x)d_2(x)/q_1q_2 + d_1(x)d_3(x)/q_1q_3 + d_2(x)d_3(x)/q_2q_3}, \text{ etc.}
\]
2.2. The Joint Distance Function

We denote the constant in (5) by $D(x)$, a function of $x$. Since the probabilities

$$p_k(x) = D(x)/\left(\frac{d_k(x)}{q_k}\right), \quad k = 1, \ldots, K,$$

add to 1 we get,

$$D(x) = \frac{\prod_{j=1}^{K} d_j(x)}{\sum_{i=1}^{K} \prod_{j \neq i} d_j(x)}.$$  \hfill (9)

$D(x)$ is called the joint distance function (abbreviated JDF) of $x$, and is, up to a constant, the harmonic mean of the $K$ weighted distances $\{d_k(x)/q_k\}$, see Arav.\textsuperscript{1}

In particular, for $K = 2$,

$$D(x) = \frac{d_1(x) d_2(x)/q_1 q_2}{d_1(x)/q_1 + d_2(x)/q_2},$$  \hfill (10)

and $K = 3$,

$$D(x) = \frac{d_1(x) d_2(x) d_3(x)/q_1 q_2 q_3}{d_1(x) d_2(x)/q_1 q_2 + d_1(x) d_3(x)/q_1 q_3 + d_2(x) d_3(x)/q_2 q_3}.$$  \hfill (11)

Example 3: Figure 2(a) shows level sets of the JDF (10) for the data of Example 1.

2.3. An Extremal Principle

The principle (5) may be derived from an extremal principle. For notational simplicity we consider here the case of 2 clusters, but the results are essentially the same in the general case.

Let $x$ be a given data point with distances $d_1(x), d_2(x)$ to the cluster centers, and assume the cluster sizes $q_1, q_2$ known. Then the probabilities in (7) are the optimal solutions of the extremal problem

$$\min \frac{d_1(x) p_1^2}{q_1} + \frac{d_2(x) p_2^2}{q_2}$$

s.t. $p_1 + p_2 = 1$

$p_1, p_2 \geq 0$
Indeed, the Lagrangian of this problem is

$$L(p_1, p_2, \lambda) = \frac{d_1(x)}{q_1} p_1^2 + \frac{d_2(x)}{q_2} p_2^2 + \lambda (1 - p_1 + p_2)$$  \hspace{1cm} (13)

and zeroing the partials \( \partial L / \partial p_i \) gives the principle (5).

Substituting the probabilities (7) in the Lagrangian (13) we get the optimal value of (12),

$$L^*(p_1(x), p_2(x)) = \frac{d_1(x) d_2(x) / q_1 q_2}{d_1(x)/q_1 + d_2(x)/q_2}$$  \hspace{1cm} (14)

which is again the JDF (10).

The corresponding extremal problem for the data set \( D = \{x_1, x_2, \ldots, x_N\} \) is

$$\min \sum_{i=1}^{N} \left( \frac{d_1(i) p_1(i)^2}{q_1} + \frac{d_2(i) p_2(i)^2}{q_2} \right)$$  \hspace{1cm} (15)

s.t.  \( p_1(i) + p_2(i) = 1 \),

\[ p_1(i), p_2(i) \geq 0, \; i = 1, \cdots, N, \]

where \( p_1(i), p_2(i) \) are the cluster probabilities at \( x_i \) and \( d_1(i), d_2(i) \) are the corresponding distances. The problem separates into \( N \) problems like (12),
Probabilistic Distance Clustering

and its optimal value is

\[
\sum_{i=1}^{N} \frac{d_1(i) d_2(i) / q_1 q_2}{d_1(i)/q_1 + d_2(i)/q_2}
\]

(16)

the sum of the joint distance functions of all points.

2.4. An Extremal Principle for the Cluster Sizes

Taking the cluster sizes as variables in the extremal principle (15),

\[
\min \sum_{i=1}^{N} \left( \frac{d_1(i) p_1(i)^2}{q_1} + \frac{d_2(i) p_2(i)^2}{q_2} \right)
\]

s.t. \( q_1 + q_2 = N \)

\( q_1, q_2 \geq 0 \)

with \( p_1(i), p_2(i) \) assumed known, we have the Lagrangian

\[
L(q_1, q_2, \lambda) = \sum_{i=1}^{N} \left( \frac{d_1(i) p_1(i)^2}{q_1} + \frac{d_2(i) p_2(i)^2}{q_2} \right) + \lambda(q_1 + q_2 - N)
\]

(18)

Zeroing the partials \( \partial L / \partial q_k \) gives,

\[
q_k^2 = \frac{1}{\lambda} \left( \sum_{i=1}^{N} d_k(i) p_k(i)^2 \right), \quad k = 1, 2
\]

(19)

and since \( q_1 + q_2 = N \),

\[
\frac{q_k}{N} = \left( \frac{\left( \sum_{i=1}^{N} d_1(i) p_1(i)^2 \right)^{1/2}}{\sum_{i=1}^{N} d_1(i) p_1(i)^2} \right)^{1/2} + \left( \frac{\left( \sum_{i=1}^{N} d_2(i) p_2(i)^2 \right)^{1/2}}{\sum_{i=1}^{N} d_2(i) p_2(i)^2} \right)^{1/2}, \quad k = 1, 2
\]

(20)

2.5. Centers

We rewrite (16) as a function of the cluster centers \( c_1, c_2 \),

\[
f(c_1, c_2) = \sum_{i=1}^{N} \frac{d(x_i, c_1) d(x_i, c_2)}{q_1 q_2} = \sum_{i=1}^{N} \frac{\|x_i - c_1\| \|x_i - c_2\|}{q_1 q_2}
\]

(21)

and look for centers \( c_1, c_2 \) minimizing \( f \).
Theorem 4: A stationary point of the function $f$ of (21), with Euclidean norm $\| \cdot \|$, is

$$c_1 = \sum_{i=1}^{N} \left( \frac{u_1(x_i)}{\sum_{j=1}^{N} u_1(x_j)} \right) x_i, \quad c_2 = \sum_{i=1}^{N} \left( \frac{u_2(x_i)}{\sum_{j=1}^{N} u_2(x_j)} \right) x_i,$$

where

$$u_1(x_i) = \left( \frac{d_2(x_i)^2}{q_2} \right)^2 \frac{q_1}{d_1(x_i)} \left( \frac{d_1(x_i)}{q_1} + \frac{d_2(x_i)}{q_2} \right)^2, \quad u_2(x_i) = \left( \frac{d_1(x_i)^2}{q_1} \right)^2 \frac{q_2}{d_2(x_i)} \left( \frac{d_1(x_i)}{q_1} + \frac{d_2(x_i)}{q_2} \right)^2,$$

or equivalently, in terms of the probabilities (7),

$$u_1(x_i) = \frac{p_1(x_i)^2 q_1}{d_1(x_i)}, \quad u_2(x_i) = \frac{p_2(x_i)^2 q_2}{d_2(x_i)}.$$

Proof: The gradient of $d(x, c) = \| x - c \|$ with respect to $c$ is

$$\nabla_c \| x - c \| = -\frac{x - c}{\| x - c \|} = -\frac{x - c}{d(x, c)}.$$

Therefore, $\nabla_{c_1} f(c_1, c_2)$ is

$$1 \sum_{i=1}^{N} \left( \frac{d_1(x_i)}{q_1} + \frac{d_2(x_i)}{q_2} \right) \frac{d_2(x_i)}{q_2} \left( \frac{d_1(x_i)}{q_1} \right)^2\left( \frac{d_2(x_i)}{q_2} \right)^2 \frac{1}{q_2} \left( -\frac{x_i - c_1}{d_1(x_i)} \right)$$

$$= \sum_{i=1}^{N} \left( \frac{d_1(x_i)}{q_1} + \frac{d_2(x_i)}{q_2} \right) \frac{d_2(x_i)}{q_2} \left( \frac{d_1(x_i)}{q_1} \right)^2\left( \frac{d_2(x_i)}{q_2} \right)^2.$$

Zeroing the last expression we get

$$\sum_{j=1}^{N} \frac{d_2(x_j)^2/q_2^2}{d_1(x_j)/q_1} c_1 = \sum_{i=1}^{N} \frac{d_2(x_i)^2/q_2^2}{d_1(x_i)/q_1} x_i,$$

or

$$\sum_{j=1}^{N} \frac{d_2(x_j)^2}{d_1(x_j)/q_1 + d_2(x_j)/q_2} c_1 = \sum_{i=1}^{N} \frac{d_2(x_i)^2}{d_1(x_i)/q_1 + d_2(x_i)/q_2} x_i.$$

which can be written, using (23), as
\[
\sum_{j=1}^{N} u_1(x_j) c_1 = \sum_{i=1}^{N} u_1(x_i) x_i
\]
proving \( c_1 \) in (22). The rest follows similarly from \( \nabla c_2 \mathbf{f}(c_1, c_2) = 0 \).

The formulas (22)–(23) hold also if the norm used in (21) is elliptic,
\[
\|u\| = \langle u, Q u \rangle^{1/2},
\]
with \( \langle \cdot, \cdot \rangle \) the standard inner product, and \( Q \) a positive definite matrix.

**Corollary 5:** Consider the function \( f(c_1, c_2) \)
\[
f(c_1, c_2) = \sum_{i=1}^{N} \frac{d(x_i, c_1)}{\sqrt{q_1 q_2}} \frac{d(x_i, c_2)}{\sqrt{q_1 q_2}} = \sum_{i=1}^{N} \frac{\|x_i - c_1\|}{q_1} \frac{\|x_i - c_2\|}{q_2},
\]
using the distance functions,
\[
d(x, c_k) = \langle (x - c_k), Q_k (x - c_k) \rangle^{1/2}, \quad k = 1, 2,
\]
for some positive-definite matrices \( Q_k \). Then a stationary point \((c_1, c_2)\) of \( f \) is given by (22)–(23).

**Proof:** The proof of Theorem 4 can be adapted. The result analogous to (26) is
\[
\sum_{j=1}^{N} \frac{d_2(x_j)^2/q_2^2}{d_1(x_j)/q_1} Q_1 c_1 = \sum_{i=1}^{N} \frac{d_2(x_i)^2/q_2^2}{d_1(x_i)/q_1} Q_1 x_i
\]
where \( Q_1 \) can be cancelled.

Corollary 5 applies, in particular, to the Mahalanobis distance
\[
d(x, c_k) = \sqrt{(x - c_k)^T \Sigma_k^{-1} (x - c_k)}
\]
where \( \Sigma_k \) is the (given or computed) covariance matrix of the cluster \( C_k \).
The representation (24) is convenient for the general case of $K$ clusters.

**Corollary 6:** Let the sum, for all data points, of the joint distance functions (9), be written as a function of the cluster centers $c_k$,

$$f(c_1, c_2, \ldots, c_K) = \sum_{i=1}^{N} \frac{K}{j=1} \prod_{j=1}^{K} \frac{\|x_i - c_j\|^q_j}{q_j}.$$  

(29)

Then a stationary point $(c_1, c_2, \ldots, c_K)$ for $f$ is

$$c_k = \sum_{i=1}^{N} \left( \frac{u_k(x_i)}{\sum_{j=1}^{K} u_j(x_i)} \right) x_i, \text{ with } u_k(x_i) = \frac{P_k(x_i)^2 q_k}{d_k(x_i)},$$  

(30)

for $k = 1, \ldots, K$, $i = 1, \ldots, N$.

### 3. The PDQ Algorithm

The above ideas are implemented in an algorithm for the unsupervised clustering of data. We call it the **PDQ Algorithm** (P for probability, D for distance and Q for the cluster sizes.)

For simplicity, we describe the algorithm for the case of 2 clusters.

**Algorithm 1: The PDQ Algorithm.**

| Initialization: | given data set $D$ with $N$ points, |
| any two centers $c_1, c_2$, |
| any two cluster sizes $q_1, q_2$, $q_1 + q_2 = N$, |
| $\epsilon > 0$ |

| Iteration: | compute distances from $c_1, c_2$ for all $x \in D$ |
| Step 1 | |
| Step 2 | update the cluster sizes $q_1^+, q_2^+$ (using (20)) |
| Step 3 | update the centers $c_1^+, c_2^+$ (using (22)–(23)) |
| Step 4 | if $\|c_1^+ - c_1\| + \|c_2^+ - c_2\| < \epsilon$ stop |
| return to step 1 |

The algorithm iterates between the cluster size estimates (20), the cluster **centers**, (22), expressed as stationary points for the JDF (21), and the **distances** of the data points to these centers. The cluster **probabilities**, (7), are not used explicitly.
Remarks

(a) The distances used in Step 1 can be Euclidean or elliptic (the formulas (22)–(23) are valid in both cases.)

(b) In particular, if the Mahalanobis distance
\[
d(x, c_k) = \sqrt{(x - c_k)^T \Sigma_k^{-1} (x - c_k)}
\] is used, the covariance matrix \( \Sigma_k \) of the \( k \)th-cluster, can be estimated at each iteration by
\[
\Sigma_k = \frac{\sum_{i=1}^{N} u_k(x_i)(x_i - c_k)(x_i - c_k)^T}{\sum_{i=1}^{N} u_k(x_i)}
\]
with \( u_k(x_i) \) given by (23).

(c) If the cluster sizes \( q_1, q_2 \) are known, they are used as the initial estimates and are not updated thereafter, in other words, Step 2 is absent.

(d) The computations stop (in Step 4) when the centers stop moving, at which point the cluster membership probabilities may be computed by (7). These probabilities are not used in the algorithm, but can be used the final classification of the data.

(e) By truncating the final probabilities, the final clusters can be determined, and the estimates of the covariance matrices refined based on these clusters.

Example 7: Figure 2(b) shows probability level sets for the data of Example 1, as determined by the principle (5) using the centers and covariances computed by Algorithm 1.

4. Estimation of Parameters of Normal Distribution

The PDQ Algorithm of §3 is an alternative to the well known EM method for de-mixing distributions. Given observations from a density \( \phi(x) \), that is itself a mixture of two densities,
\[
\phi(x) = \pi \phi_1(x) + (1 - \pi) \phi_2(x),
\]
it is required to estimate the weight \( \pi \), and the relevant parameters of the distributions \( \phi_1 \) and \( \phi_2 \).
A common situation is when the distribution $\phi$ is a mixture of normal distributions $\phi_k$, each with its mean $c_k$ and covariance $\Sigma_k$ that need to be estimated.

For the purpose of comparison with Algorithm 1, we present here the EM Method for a Gaussian mixture (32) of two distributions,

$$
\phi_k(x) = \frac{1}{\sqrt{(2\pi)^n|\Sigma_k|}} \exp \left\{ -\frac{1}{2} (x - c_k)^T \Sigma_k^{-1} (x - c_k) \right\}.
$$

(33)

For further detail see, e.g., Hastie et al.,\textsuperscript{6}.

Algorithm 2: The EM Method.

| Initialization: | given data set $D$ with $N$ points, |
|                | initial guesses for the parameters $\hat{c}_1, \hat{c}_2, \hat{\Sigma}_1, \hat{\Sigma}_2, \hat{\pi}$ |
| Iteration:     | Step 1: For all $x_i \in D$ compute the “responsibilities” :
|                | $p_1(x_i) = \frac{\hat{\pi} \phi_1(x_i)}{\hat{\pi} \phi_1(x_i) + (1 - \hat{\pi}) \phi_2(x_i)}$ , |
|                | $p_2(x_i) = 1 - p_1(x_i)$ . |
|                | Step 2 update the centers and covariances:
|                | $\hat{c}_k = \sum_{i=1}^{N} \left( \frac{p_k(x_i)}{\sum_{j=1}^{N} p_k(x_j)} \right) x_i , $ |
|                | $\hat{\Sigma}_k = \sum_{i=1}^{N} \left( \frac{p_k(x_i)}{\sum_{j=1}^{N} p_k(x_j)} \right) (x_i - \hat{c}_k)(x_i - \hat{c}_k)^T , k = 1, 2$ |
|                | Step 3 update the mixing probabilities (weights):
|                | $\hat{\pi} = \sum_{i=1}^{N} p_1(x_i)$ |
|                | Step 4 stop or return to step 1 |

Remarks

(a) The “responsibilities” in Step 1 correspond to the cluster membership probabilities in Algorithm 1.
(b) Step 1 requires both the Mahalanobis distance (2) and the evaluation of the density (33).
(c) Step 2 is computationally similar to Step 3 of Algorithm 1.
(d) The stopping rule (Step 4) is again the convergence of centers as in Algorithm 1.

4.1. A Comparison of the PDQ Algorithm (Algorithm 1) and the EM Method (Algorithm 2.)

(a) The EM Algorithm is based on maximum likelihood, and therefore depends on the density functions in the mix, requiring different computations for different densities. The PDQ Algorithm is parameter free, making no assumptions about the densities, and using the same formulas in all cases.

(b) In each EM iteration the density functions must be evaluated, requiring $K N$ function evaluations (in Step 1) where $K$ is the number of densities in the mixture. In comparison, the PDQ iterations are cheaper, requiring no function evaluations.

(c) Because the EM iterations are costly, it is common to use another method, e.g., the $K$–means method, as a pre–processor, to get closer to the centers before starting EM. The PDQ Algorithm need no such switch, and works well with a cold start.

(d) If correct assumptions are made regarding the mixing distributions, then the EM method has an advantage over the PDQ method, as illustrated in Example 10 below.

(e) While the numerical comparison between the two algorithms should best be done by others, our preliminary tests show the two algorithms to be roughly equivalent in terms of the returned results, with the PDQ Algorithm somewhat faster.

5. Numerical Examples

In Examples 8–10 below the PDQ and EM Algorithms were applied to the same data, in order to compare their performance. The results are reported in Tables 1–4. These examples are typical representatives of many numerical tests we did.

Both programs used here were written in MATLAB, the EM code by Tsui, and the PDQ code by the first author.

The comparison is subject to the following limitations:

(a) The EM program code uses the $K$–means method (Hartigan, as a
preprocessor to get a good start. The number of iterations and running time reported for this program in Table 4 is just for the EM part, not including the initialization work by the $K$–means part.

(b) Our PDQ code is the first un–optimized, un–finessed version, a verbatim implementation of Algorithm 1.

(c) The number of iterations depends on the stopping rule. In the PDQ Algorithm, the stopping rule is Step 4 of Algorithm 1, and the number of iterations will increase the smaller is $\epsilon$. In the EM Algorithm the stopping rule does involve also the convergence of the likelihood function, and the effect of the tolerance $\epsilon$ is less pronounced.

(d) The number of iterations depends also on the initial estimates, the better the estimates, the fewer iterations will be required. In our PDQ code the initial solutions can be specified, or are randomly chosen. The EM program gets its initial solution from its $K$–means preprocessor.

Example 8: Algorithms 1 and 2 were applied to the data of Example 1. Both algorithms give good estimates of the true parameters, see Table 1. The comparison of running time and iterations is inconclusive, see Table 4.

<table>
<thead>
<tr>
<th>Center</th>
<th>True Parameters</th>
<th>The PDQ Algorithm (Algorithm 1)</th>
<th>The EM Method (Algorithm 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>$(2, 0)$</td>
<td>$(2.0036, -0.0542)$</td>
<td>$(2.0011, -0.0284)$</td>
</tr>
<tr>
<td>$c_2$</td>
<td>$(3, 0)$</td>
<td>$(2.9993, -0.0010)$</td>
<td>$(3.0033, -0.0018)$</td>
</tr>
</tbody>
</table>

Example 9: Consider the data set shown in Figure 3. The points of the right cluster were generated using a radially symmetric distribution function $\text{Prob}\{\|x - \mu_2\| \leq r\} = (4/3)r$ in a circle of diameter 1.5 centered at $\mu_2 = (1, 0)$, and the smaller cluster on the left was similarly generated in a circle of diameter 0.1 centered at $(0, 0)$. The ratio of sizes is 1:20.

As shown in Table 2 and Figure 4(b), the EM Method gives bad estimates of the left center, and of the weights. The estimates provided by the PDQ Algorithm are better, see Figure 4(a).
The EM Method also took long time, see Table 4. In repeated trials, it did not work for $\epsilon = 0.1$, and sometimes for $\epsilon = 0.01$.

**Example 10:** Consider the data set shown in Figure 5(a). It consists of three clusters of equal size, 200 points each, generated from Normal dis-
Table 2. A comparison of methods for the data of Example 9

<table>
<thead>
<tr>
<th></th>
<th>True Parameters</th>
<th>The PDQ Algorithm (Algorithm 1)</th>
<th>The EM Method (Algorithm 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Centers</strong></td>
<td><strong>c₁=(0,0)</strong></td>
<td><strong>c₁=(0.0023 ,0.0022)</strong></td>
<td><strong>c₁=(0.5429 ,0.0714)</strong></td>
</tr>
<tr>
<td></td>
<td><strong>c₂=(1,0)</strong></td>
<td><strong>c₂=(1.0080 ,0.0063)</strong></td>
<td><strong>c₂=(1.0603 ,0.02451)</strong></td>
</tr>
<tr>
<td><strong>Weights</strong></td>
<td>(0.0476 , 0.9524)</td>
<td>(0.0534 , 0.9466)</td>
<td>(0.1851 , 0.8149)</td>
</tr>
</tbody>
</table>

tributions $N(\mu_i, \Sigma_i)$, with parameters $\mu_i, \Sigma_i$ given in the left column of Table 3. A similar example appears as Fig. 9.6, p. 593, in Tan et al.\textsuperscript{14}

![a data set with three clusters](image1.png)  ![the level sets of the JDF](image2.png)

Fig. 5. The results of PDQ Algorithm for Example 10

As noted in § 4.1(d), if the assumptions on the mixing distributions are justified, the EM Method gives good estimates of the relevant parameters. The PDQ Algorithm, does not require or depend on such assumptions, and still gives decent estimates. This is illustrated in Table 3.

6. Multi Facility Location Problems

The **location problem** is to locate a facility, or facilities, to serve optimally a given set of customers. The customers are given by their coordinates and
Table 3. A comparison of methods for the data of Example 10

<table>
<thead>
<tr>
<th>True Parameters</th>
<th>The PDQ Algorithm (Algorithm 1)</th>
<th>The EM Method (Algorithm 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mathbf{c}_1 = (0, 1) )</td>
<td>( \hat{\mathbf{c}}_1 = (0.0053, 1.0239) )</td>
<td>( \hat{\mathbf{c}}_1 = (0.0049, 0.9916) )</td>
</tr>
<tr>
<td>( \mathbf{c}_2 = (1, 0.7) )</td>
<td>( \hat{\mathbf{c}}_2 = (0.9604, 0.7146) )</td>
<td>( \hat{\mathbf{c}}_2 = (0.9855, 0.6939) )</td>
</tr>
<tr>
<td>( \mathbf{c}_3 = (1, 1.3) )</td>
<td>( \hat{\mathbf{c}}_3 = (1.0735, 1.2748) )</td>
<td>( \hat{\mathbf{c}}_3 = (1.0376, 1.3083) )</td>
</tr>
<tr>
<td>Covariance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \Sigma_1 = \begin{pmatrix} 0.01 &amp; 0 \ 0 &amp; 0.1 \end{pmatrix} )</td>
<td>( \hat{\Sigma}_1 = \begin{pmatrix} 0.0134 &amp; -0.0006 \ -0.0006 &amp; 0.1074 \end{pmatrix} )</td>
<td>( \hat{\Sigma}_1 = \begin{pmatrix} 0.0091 &amp; -0.0018 \ -0.0018 &amp; 0.1059 \end{pmatrix} )</td>
</tr>
<tr>
<td>( \Sigma_2 = \begin{pmatrix} 0.1 &amp; 0 \ 0 &amp; 0.01 \end{pmatrix} )</td>
<td>( \hat{\Sigma}_2 = \begin{pmatrix} 0.0828 &amp; 0.0023 \ 0.0023 &amp; 0.0117 \end{pmatrix} )</td>
<td>( \hat{\Sigma}_2 = \begin{pmatrix} 0.1012 &amp; 0.0053 \ 0.0053 &amp; 0.0122 \end{pmatrix} )</td>
</tr>
<tr>
<td>( \Sigma_3 = \begin{pmatrix} 0.1 &amp; 0 \ 0 &amp; 0.01 \end{pmatrix} )</td>
<td>( \hat{\Sigma}_3 = \begin{pmatrix} 0.0907 &amp; -0.0040 \ -0.0040 &amp; 0.0123 \end{pmatrix} )</td>
<td>( \hat{\Sigma}_3 = \begin{pmatrix} 0.0981 &amp; -0.0005 \ -0.0005 &amp; 0.0090 \end{pmatrix} )</td>
</tr>
<tr>
<td>Weights</td>
<td>( (0.333, 0.333, 0.333) )</td>
<td>( (0.3297, 0.3345, 0.3358) )</td>
</tr>
</tbody>
</table>

Table 4. Summary of computation results for 3 examples. See §5(a) for explanation of the EM running time and iterations count.

<table>
<thead>
<tr>
<th>Example</th>
<th>( \epsilon )</th>
<th>PDQ Algorithm</th>
<th>EM Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Iterations</td>
<td>Time (sec.)</td>
</tr>
<tr>
<td>Example 8</td>
<td>0.01</td>
<td>5</td>
<td>3.32</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>2</td>
<td>1.42</td>
</tr>
<tr>
<td>Example 9</td>
<td>0.01</td>
<td>8</td>
<td>3.89</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>2</td>
<td>1.02</td>
</tr>
<tr>
<td>Example 10</td>
<td>0.01</td>
<td>11</td>
<td>2.29</td>
</tr>
</tbody>
</table>

Assuming \( N \) customers, the data of the problem is a set of points \( \mathcal{A} = \{ \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \} \) in \( \mathbb{R}^n \) and a corresponding set of positive weights (demands) \( \{ w_1, w_2, \ldots, w_N \} \).

### 6.1. Fermat–Weber location problem

The **Fermat–Weber location problem** is to find a point \( \mathbf{c} \) in \( \mathbb{R}^n \) that minimizes

\[
f(\mathbf{c}) = \sum_{i=1}^{N} w_i \| \mathbf{c} - \mathbf{x}_i \|,
\]

the sum of the weighted Euclidean distances between the customers \( \mathbf{x}_i \) and the facility \( \mathbf{c} \). The gradient of \( f \)

\[
\nabla f(\mathbf{c}) = \sum_{i=1}^{N} w_i \frac{\mathbf{c} - \mathbf{x}_i}{\| \mathbf{c} - \mathbf{x}_i \|}
\]
exists for all \( c \notin A \). A point \( c^* \) is optimal iff \( 0 \in \partial f(c^*) \), which reduces to \( \nabla f(c^*) = 0 \) if \( f \) is differentiable at \( c^* \). It follows from (35) that \( c^* \) is a convex combination of the points of \( A \),

\[
c^* = \sum_{i=1}^{N} \lambda_i(c^*) x_i ,
\]

with weights

\[
\lambda_i(c) = \frac{w_i \| c - x_i \|^{-1}}{N \sum_{j=1}^{N} w_j \| c - x_j \|^{-1}} . \tag{37}
\]

The Weiszfeld Method,\textsuperscript{17} for solving this problem is an iterative method with updates

\[
c_+ := \sum_{i=1}^{N} \lambda_i(c) x_i , \tag{38}
\]

giving the next iterate \( c_+ \) as a convex combination, with weights \( \lambda_i(c) \) computed by (37) for the current iterate \( c \). Note that \( \lambda_i(c) \) is undefined if \( c = x_i \). If the Weiszfeld iterates converge to a point \( c^* \), then \( c^* \) is optimal by (36).

The Weiszfeld method is the best-known method for solving the Fermat–Weber location problem, see the history in \textsuperscript{13}.

### 6.2. Multiple Facility Location Problem

The **multiple facility location problem** (abbreviated MFLP) is to locate several facilities so as to serve optimally the given customers. We assume no interactions between facilities. If the number of facilities \( K \) is given, the problem is to:

- (a) determine the locations \( c_k \) of the facilities (location decision), and
- (b) assign customers to facilities (allocation or assignment),

so as to minimize the sum of weighted distances from facilities to assigned points.

\[
\min \sum_{k=1,\ldots,K} \sum_{i \in C_k} w_i \| x_i - c_k \| \tag{39}
\]

where \( C_k \) is the index set of the customers assigned to the \( k \)th facility. If the number of facilities is not given in advance, and must be determined,
the problem is written as,

\[
\min_{K=1, \ldots, N} \sum_{k=1, \ldots, K} \sum_{i \in C_k} w_i \| x_i - c_k \| \quad (40)
\]

When \( K = 1 \), the Weiszfeld method, (38), expresses the facility location as a convex combination of the customers' coordinates.

For \( K > 1 \) the PDQ center formulas (30) represent each facility as a convex combination of the customers' coordinates, and these reduce to Weiszfeld's formula if \( K = 1 \), in which case all the probabilities in (30) equal 1. When applied to MFLP, the PDQ Algorithm is thus an extension of Weiszfeld’s Method.\(^9\)

Examples 11 and 12 illustrate the PDQ Algorithm for solving MFLP’s.

**Example 11:** (Cooper,\(^3\) p. 47) It is required to locate 3 facilities to serve the following 15 customers

<table>
<thead>
<tr>
<th>Customer</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )-coordinate</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>13</td>
<td>12</td>
<td>13</td>
<td>28</td>
<td>21</td>
<td>25</td>
<td>31</td>
<td>39</td>
<td>39</td>
<td>45</td>
<td>41</td>
<td>49</td>
</tr>
<tr>
<td>( y )-coordinate</td>
<td>9</td>
<td>25</td>
<td>48</td>
<td>4</td>
<td>19</td>
<td>39</td>
<td>37</td>
<td>45</td>
<td>50</td>
<td>9</td>
<td>2</td>
<td>16</td>
<td>22</td>
<td>30</td>
<td>31</td>
</tr>
</tbody>
</table>

These data points are shown in Fig. 6(a). The PDQ algorithm, with \( \epsilon = 0.001 \) (in Step 4), required 6 iterations to determine the three clusters, with approximate centers. The final centers, computed after the clusters were determined (see Remark 3(e)), are shown in Fig. 6(b). In the top left cluster, the facility practically coincides with one of the customers.

**Example 12:** Fig. 7 shows a data set with \( N = 1000 \) random points in \([-10, 10]^2\), representing the customers. It is required to locate \( K = 4 \) facilities to serve the customers. The algorithm starts with 4 random initial locations (centers.) Using different symbols: o, x, +, * for 4 clusters, Figure 7(a) illustrates the convergence from arbitrary initial points. The final clusters, obtained by truncating the cluster probabilities, allow better estimates of the facilities locations (centers), see Remark 3(e). Figure 7(b) shows the final clusters and facilities.

The PDQ Algorithm also solves **Capacitated MFLP**’s, where the cluster sizes \( q_k \) in (30) play the role of the facility capacities. When these are
given, we have a capacitated MFLP, and the PDQ Algorithm simplifies further, see § 3, note (c). This is illustrated in Example 13 and Figure 8 below.

**Example 13:** Consider the same 1000 random data points of Example 12, and 4 facilities with capacities given in percentages as 35%, 25%, 15%, and 25% of the total demand. The PDQ Algorithm starts with 4 random initial
facilities (centers). Figure 8(a) shows the level sets of the JDF computed by the PDQ algorithm, and Figure 8(b) shows the final facilities and their clusters.

![Level sets of the JDF](image1)

![Final clusters and centers](image2)

Fig. 8. Results for Example 13

7. Determining the “right” number of clusters

A difficult problem in clustering is to determine the “right” number $K$ of clusters that fits a data set with $N$ points. In dichotomous situations the unambiguous answer is $K = 2$, but in general, the answer lies between the two extremes of $K = 1$ (one cluster fits all), and $K = N$ (each point is a cluster.)

Many answers and criteria were proposed for this problem, see, e.g., the survey in 4.

The joint distance function (29) helps resolve this issue. Indeed, the value of the JDF decreases monotonically with $K$, the number of clusters, and the decrease is precipitous until the “right” number is reached, and after that the rate of decrease is small. This is illustrated in Example 14 and Figures 9–11 below.

This approach is useful because the PDQ algorithm is fast, and clustering for several values of $K$ is feasible if finding the correct $K$ is important.11

Example 14: Figure 9(a) shows a data set with 2 clusters. The PDQ
algorithm was applied to this data set, and the values of the JDF are computed for values of $K$ from 1 to 10, the results are plotted in Figure 9(b). Note the change of slope of the JDF at $K = 2$, the correct number of clusters.

Figs. 10(a) and 11(a) show similarly data sets with $K = 3$ and $K = 4$ clusters, respectively. The JDF, computed by the PDQ algorithm, shown in Figs. 10(b) and 11(b), reveal the correct number of clusters.

References

Probabilistic Distance Clustering

Fig. 10. Note the change of slope of the JDF at $K = 3$.

Fig. 11. Note the change of slope of the JDF at $K = 4$.


