Clustering by Random Projections

Thierry Urruty[1], Chabane Djeraba[1], and Dan A. Simovici[2]

[1] LIFL-UMR CNRS 8022, Laboratoire d'Informatique Fondamentale de Lille, Université de Lille 1, France, urruty,djeraba@lifl.fr

[2] University of Massachusetts Boston, Department of Computer Science, Boston, Massachusetts 02125, USA, dsim@cs.umb.edu

Abstract. Clustering algorithms for multidimensional numerical data must overcome special difficulties due to the irregularities of data distribution. We present a clustering algorithm for numerical data that combines ideas from random projection techniques and density-based clustering. The algorithm consists of two phases: the first phase that entails the use of random projections to detect clusters, and the second phase that consists of certain post-processing techniques of clusters obtained by several random projections. Experiments were performed on synthetic data consisting of randomly-generated points in $\mathbb{R}^n$, synthetic images containing colored regions randomly distributed, and, finally, real images. Our results suggest the potential of our algorithm for image segmentation.

1 Introduction

Clustering is a central preoccupation in data mining and clustering algorithms impact a multitude of data mining applications ([CUDW02,ZSD02,Dje03]), including multimedia data mining. The problem has been studied by several research communities ranging from statistics to machine learning and the state of the art is exposed in surveys that appeared with some regularity over the years (see [JMF99,JD88]). Clustering in spaces with low dimensionality is relatively easy. For example, in a unidimensional space it is easy to identify the regions of high density of points by a simple linear scan. With increased dimensionality the problem grows in complexity. The notion of projected clustering was introduced by Agrawal et al. in [AGGR98], who made the crucial observations that points may cluster better in subspaces of lower dimensionality than in the entire space $\mathbb{R}^n$. They developed the CLIQUE algorithm that works starting with low dimensional subspaces towards higher dimensional subspaces. In [APW$^+$99] Aggarwal et al. focus on a technique to discover clusters in small dimensional subspaces, which is the focus of their PROCLUS algorithm. The theoretical support of these techniques can be found in Johnson-Lindenstrauss Lemma [JL84] which asserts that a set of points in a high-dimensional Euclidean space can be projected into a low-dimensional Euclidean such that the distance between any two points changes by only a factor of $1 \pm \epsilon$ for $\epsilon \in (0, 1)$. Simplifications of the proof of this result have been obtained by Frankl and Maehara [FM88] and by Dasgupta and Gupta [DG99]. An especially useful source is the monograph [Vem04].

The number of clusters is a given parameter in PROCLUS and the algorithm identifies these clusters and a set of dimensions associated with each cluster such that the

points of the cluster are correlated with these dimensions. Another contribution to projective clustering is [AM04], where an objective function is introduced that takes into account a tradeoff between the dimension of a subspace and the clustering error; an extension of k-means to projective clustering in arbitrary subspaces is introduced. Our approach is similar to the approach adopted in [AGGR98] in that we construct clusters in low dimensional spaces and, then select those dimensions that can best help to identify clusters in the original data set. Our main contribution consists in choosing a random frame of reference for the data set and execute the projections on the subspaces that correspond to this randomly chosen axes. We show that this process has a certain advantage over using the natural system of coordinates in that it diminishes the chance of the occultation phenomenon, which occurs when the projections of two distinct clusters of the data on a subspace are not disjoint. Static segmentation of images regarded as partitioning an image to a number of regions that represent a meaningful part of the image can be helped, as we show, by applying clustering techniques (see [JF96]). Our clustering algorithm combines ideas from random projection techniques and density-based clustering. The distance between points in $\mathbb{R}^n$ is the Euclidean distance. The proposed algorithm is applicable to numeric data, that is, to data in $\mathbb{R}^n$ and involves projecting the data on a randomly chosen base. Then, histograms of the uni-dimensional projections are combined to yield the locations of clusters in $\mathbb{R}^n$.

The paper begins with a probabilistic evaluation of the projection technique. Namely, in Section 2 we evaluate the probability that the distance between random projections on subspaces reproduces to a certain extent the distance between the original points in $\mathbb{R}^n$ and the probabilities that random projections of separate clusters may have non-empty intersection and, therefore, reduce the usefulness of certain projections. In Section 3 we discuss the clustering algorithm including two important post-processing techniques and we show that the time complexity is of the order of $O(N \log N)$ when the size of the data set is large compared to the number of dimensions, comparable with density-based clustering [SEKX98]. Section 4 presents our experimental work performed on three types of data: synthetic data, data obtained from synthetic images, and data obtained from real images. Experiments with groupings of pixels extracted from images, particularly from real images show the potential of the algorithm as a segmentation technique and provide a good criterion for validation of clusterings.

2 Clusters and Random Projections

Let S be a finite subset of $\mathbb{R}^n$ and let δ be a positive real number. Consider a measure $m : \mathcal{P}(\mathbb{R}^n) \longrightarrow \mathbb{R}_{\geq 0}$. The value $m(C)$ is, in general, the volume of the projection of C on a subspace of $\mathbb{R}^n$.

A *δ-clustering of S* is a family $\kappa = \{C_1, \ldots, C_p\}$ of non-empty subsets of $\mathbb{R}^n$ (referred to as the *constituents* of the clustering) that satisfy the following conditions:

1. the sets of κ that are pairwise disjoint;
2. for every i, $1 \leq i \leq p$ density of the points of S in any of the sets C_i exceeds δ, that is, we have:
$$\frac{|S \cap C_i|}{m(C_i)} \geq \delta.$$

The *clusters* of the clustering κ are the sets $S \cap C_i$ for $1 \leq i \leq p$.

The set of points located outside the sets C_i, $\mathsf{UNC}(\kappa) = S - \bigcup_{i=1}^{p} C_i$ is the *set of unclassified points of* S.

A *clustering of* S is a family $\kappa = \{C_1, \ldots, C_p\}$ that is a δ-clustering of S for some $\delta > 0$.

The second condition of the above definition insures that the density of the points in each of the sets C_i is sufficiently high.

Projections on the subspace of $\mathbb{R}^n$ determined by the coordinates $i_1, \ldots, i_t$ are denoted by $\mathsf{proj}_{i_1 \cdots i_t} : \mathbb{R}^n \longrightarrow \mathbb{R}^t$.

Let $\mathbf{H}$ be a random $n \times n$-matrix that is orthogonal. Such a matrix can be obtained, for example, by randomly choosing the components on an $n \times n$-matrix using an uniform distribution on an interval and, then, applying the Gram-Schmidt technique to produce an orthogonal matrix.

A random projection of $\mathbb{R}^n$ is a linear transformation $\Phi_{\mathbf{H}} : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ defined by $\Phi_{\mathbf{H}}(\mathbf{x}) = \mathbf{H}\mathbf{x}$ for $\mathbf{x} \in \mathbb{R}^n$. The set of rows $\mathbf{u}_1, \ldots, \mathbf{u}_n$ of $\mathbf{H}$ is referred to an n-dimensional *random frame*.

Identifying clusters in one-dimension is a relatively straightforward process using an algorithm (described in Section 3) that builds histograms of the line coordinates of the projections of the points. The inverse statement does not hold; if the projection of a subset K of $\mathbb{R}^n$ on a lower dimension subspace is a cluster we cannot conclude that the set K itself is a cluster. Another difficulty is that disjoint clusters in the n-dimensional space may have non-disjoint projections on lower-dimensional subspaces of $\mathbb{R}^n$, a phenomenon that we refer to as *occultation*.

Let C, D be two clusters in $\mathbb{R}^n$ and let $\mathbf{u}$ be a unit vector in the same space. To simplify the presentation assume that C and D are approximated by spheres of radius r_1 and r_2, centered in the points $\mathbf{c}$ and $\mathbf{d}$, respectively. The orthogonal projection of a set K on a vector $\mathbf{u}$ is the set:

$$\mathsf{proj}_u(K) = \{\mathbf{u} \cdot \mathbf{x} | \mathbf{x} \in K\}.$$

An ***u**-occultation* of the clusters C, D occurs if

$$\mathsf{proj}_u(C) \cap \mathsf{proj}_u(D) \neq \emptyset,$$

a situation which is represented in Figure 1.

This is an inconvenient situation from our point of view since it fuses the two projections of C and D on the vector $\mathbf{u}$.

We need to evaluate the probability that an $\mathbf{u}$-occultation may occur for clusters since we will use cluster uni-dimensional projections for the identification of these clusters in $\mathbb{R}^n$. As before, we assume that $\mathbf{u}$ is a unit random vector. The angle α between $\mathbf{u}$ and the vector $\mathbf{d} - \mathbf{c}$ is uniformly distributed in the interval $[0, 2\pi]$. The discussion is essentially the same for projections on subspaces having an arbitrary dimensionality. Under the previous assumptions an $\mathbf{u}$-occultation of the clusters C, D occurs when the length of the projection of the segment that joins $\mathbf{c}$ to $\mathbf{d}$ is inferior to $r_1 + r_2$; in other words if $|\mathbf{u} \cdot (\mathbf{d} - \mathbf{c})| = \| \mathbf{d} - \mathbf{c} \| \, |\cos \alpha| \leq r_1 + r_2$.

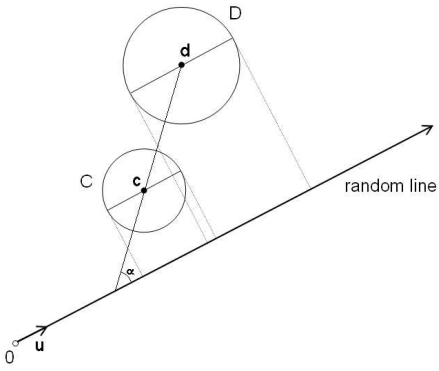

Fig. 1. Cluster Occultation

Consequently, the probability of an **u**-occultation of the clusters is the number:

$$P\left(-\frac{r_1+r_2}{\parallel \mathbf{d}-\mathbf{c} \parallel} \leq \cos\alpha \leq \frac{r_1+r_2}{\parallel \mathbf{d}-\mathbf{c} \parallel}\right),$$

which is easily seen to equal to

$$1-\frac{2}{\pi}\arccos\left(\frac{r_1+r_2}{\parallel \mathbf{d}-\mathbf{c} \parallel}\right),$$

whenever $\parallel \mathbf{d}-\mathbf{c} \parallel \geq r_1+r_2$, which happens when the clusters are sufficiently tight. Using the MacLaurin series expansion of $\arccos z$,

$$\arccos z = \frac{\pi}{2} - \left(z + \frac{z^3}{6} + \frac{3}{40}\frac{z^5}{5} + \cdots\right),$$

we can approximate this value by $z = \frac{2(r_1+r_2)}{\pi\|\mathbf{d}-\mathbf{c}\|}$.

Let O_i be the event that takes place when an occultation occurs on the i-th projection of the random frame, for $1 \leq i \leq n$. We need to evaluate the probability that there is at least one projection that avoids the occultation, that is, $P(\overline{O_1} \cup \cdots \cup \overline{O_n}) = 1 - P(O_1 \cap \cdots \cap O_n)$. Assuming that the events $O_1, \ldots, O_n$ are independent we have

$$P(\overline{O_1} \cup \cdots \cup \overline{O_n}) = 1 - \left(\frac{2(r_1+r_2)}{\pi \parallel \mathbf{d}-\mathbf{c} \parallel}\right)^n.$$

Of course, the independence supposition does not hold in reality. We adopt it here to obtain an estimate that is plausible and is verified by experimental work.

Thus, the probability that there is a dimension that avoids occultation is increasing quite rapidly with the number of dimensions and with the inter-cluster separation. This shows the usefulness of the randomly chosen frame in separating, at least partially, and with a degree of uncertainty, clusters that may not be differentiated through their projections on the initial system of coordinates.

3 The Clustering Algorithm

Our algorithm has a heuristic nature. The input consists of a numerical n-dimensional data set D and entails two phases: in the first phase we apply random projections to the data set and we obtain the primary clusters; in the second phase we refine the clustering by using two processes: bimodulation and cluster expansion.

The projection phase begins with a randomly chosen orthogonal $n \times n$-matrix $\mathbf{H}$ of real numbers whose rows are the $\mathbf{u}_1, \ldots, \mathbf{u}_n$. Then, the data set D is projected onto each of the n dimensions of the newly chosen randomly chosen base, resulting in n histograms. We begin by clustering the points of each of the selected uni-dimensional projections.

Each i-th histogram contains a number of k bins of width ℓ_i and the choice of k depends on the size of D. For example, for $|D| = 10^4$ we used $k = 50$. On each histogram we identify the peaks and the valleys. The peaks $h^i_1, \ldots, h^i_{m_i}$ of this histograms may correspond to n-dimensional clusters.

Suppose that the peak h^i_j of the i-th projection is located between the lows l^i_j and l^i_{j+1}. Then, the set C^i_j consists of the points that belong to the p bins located at the left of h^i_j whose heights vary between $\beta h^i_j + (1-\beta)l^i_j$ and h^i_j and the q bins located at the right of h^i_j whose heights vary between $\beta h^i_j + (1-\beta)l^i_{j'}$ and h^i_j (see Figure 2). Here β is a parameter chosen by the user that allows us to guarantee a certain cluster density.

Note that the density of the cluster $\mathsf{proj}_i(S) \cap C^i_j$ is at least

$$\frac{h^i_j + p[\beta h^i_j + (1-\beta)l_j] + q[\beta h^i_j + (1-\beta)l_{j+1}]}{(p+q+1)\ell_i},$$

which is easily seen to be at least $\frac{h^i_j \beta}{\ell_i}$. So, if we choose

$$\beta \geq \max\{\frac{\delta_1 \ell_i}{\min h^i_j} | 1 \leq i \leq n\},$$

we guarantee that the uni-dimensional clusters have the minimal density δ_1. The choice of δ_1 is determined, as we shall see by the parameter δ.

The quality of a projection is evaluated using the product between the average height of peaks and the logarithm of the number of peaks of the histogram. Only a percentage of the dimensions that correspond to these top histograms are retained for the next phase. Initially we seek to obtain a clustering that corresponds to these projections. In our experiments we used the top 10% of the histograms, a choice that is supported by our experiments (see Section 4.2).

Suppose that the peaks of the i-th random projection correspond to the intervals $C^i_1, \ldots, C^i_{p_i}$. Let t be the number of top projections. We use a file $\mathcal{F}$ that contains records having $1 + n + t$ components. Each record represents one of the points $\mathbf{x} = (x_1, \ldots, x_n)$ to be clustered and contains a point identifier, the original n coordinates $x_1, \ldots, x_n$, and, for each projection i, a number $B(\mathbf{x}, i)$ defined by:

$$B(\mathbf{x}, i) = \begin{cases} j & \text{if the } i^{th} \text{ projection of} \\ & \mathbf{x} \text{ belongs to } C^i_j, \\ 0 & \text{otherwise.} \end{cases}$$

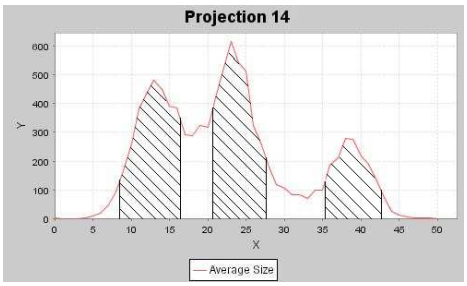

Fig. 2. Intervals around peaks in a projection

$\mathcal{F}$

Point id.	x_1	$\cdots$	x_n	$B(1,\mathbf{x})$	$\cdots$	$B(t,\mathbf{x})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
h	a_1	$\cdots$	a_n	b_1	$\cdots$	b_t
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Records containing at least one 0 are discarded since they contain points that at this stage of the algorithm are not yet affiliated with any cluster. Then, the file $\mathcal{F}$ is sorted on the fields $B(1,\mathbf{x}),\cdots,B(t,\mathbf{x})$. Each set of points that correspond to a vector $(b_1,\ldots,b_t)$ corresponds to a set

$$C^{i_1\cdots i_t}_{b_1\cdots b_t} = C^{i_1}_{b_1}\times\cdots\times C^{i_t}_{b_t}$$

which we regard as a constituent of the clustering. The condition

$$\frac{|\mathsf{proj}_{i_1\cdots i_t}(S)\cap C^{i_1\cdots i_t}_{b_1\cdots b_t}|}{m(C^{i_1\cdots i_t}_{b_1\cdots b_t})}\geq\delta$$

insures that the clusters

$$\mathsf{proj}_{i_1\cdots i_t}(S)\cap C^{i_1\cdots i_t}_{b_1\cdots b_t},$$

where $\mathsf{proj}_{i_1\cdots i_t}(S)$ is the projection of S on the dimensions $i_1\cdots i_t$ of the random frame of coordinates will have the minimum density δ provided by the user. In our experiments we used $\delta=0.01$ which reflects our decision of regarding clusters that contain less that 1% as consisting of outliers.

Note that

$$|\mathsf{proj}_{i_1\cdots i_t}(S)\cap C^{i_1\cdots i_t}_{b_1\cdots b_t}|\leq\min_r|\mathsf{proj}_{i_r}(S)\cap C^{i_r}_{b_r}|.$$

Therefore, the minima density condition imposed on the clusters implies that the unidimensional density δ_1 must be at least $\delta\ell^{t-1}$, where ℓ is the width of the bins defined above.

The time required to compute the histograms is $O(n^2N)$, where n is the number of dimensions and N is the number of points to be clustered. The cost of sorting the file

$\mathcal{F}$ is $O(N \log N)$, which brings the total cost of the algorithm to $O(n^2 N + N \log N)$. Thus, the asymptotic cost of the algorithm is $O(N \log N)$; however, when the number of dimensions is important relative to the logarithm of the number of points the $O(n^2 N)$ component is not negligible.

The post-processing of the clusters described below does not alter this asymptotic evaluation.

Assigning points left outside the clusters, to the extent that this is possible, is achieved using multiple random projections. Suppose that two random projections yield two clusterings:

$$\kappa = \{C_1, \ldots, C_p\} \text{ and } \mu = \{D_1, \ldots, D_q\}$$

and let $U_i = \mathsf{UNC}(\mu) \cap C_i$ for $1 \leq i \leq p$ and $V_j = \mathsf{UNC}(\kappa) \cap D_j$ for $1 \leq j \leq q$. Clusterings obtained by distinct random projections may be used to produce better clusterings by a process that will be referred here as *bimodulation*.

Let $\mathsf{PART}(S)$ be the set of partitions of a set S and let $\mathsf{CL}(S)$ be the set of clusterings of same set S. Every clustering $\kappa = \{C_1, \ldots, C_p\}$, defines a partition $\pi_\kappa = \{C_1, \ldots, C_p, \mathsf{UNC}(\kappa)\}$ of the set S.

Let $d : \mathsf{PART}(S) \times \mathsf{PART}(S) \longrightarrow \mathbb{R}$ be a distance defined on the set of partitions of the set S. A *d-bimodulation* is a mapping: $\Psi : \mathsf{CL}(S) \times \mathsf{CL}(S) \longrightarrow \mathsf{CL}(S) \times \mathsf{CL}(S)$ such that if $\Psi(\kappa, \mu) = (\kappa', \mu')$, then $d(\kappa', \mu') \leq d(\kappa, \mu)$. In other words, an application of a bimodulation to a pair of clusterings results in a new pair of clusterings whose partitions are closer to each other than the partitions associated to the initial pair of clusterings. We can use as a distance between partitions the Barthélemy-Montjardet distance introduced in [BL95]. If π, σ are two partitions in $\mathsf{PART}(S)$ given by:

$$\pi = \{K_1, \ldots, K_m\} \text{ and } \sigma = \{H_1, \ldots, H_n\},$$

then the distance between π and σ is given by:

$$d(\pi, \sigma) = \sum_{i=1}^{m} |K_i|^2 + \sum_{j=1}^{n} |H_j|^2 - 2 \sum_{i=1}^{m} \sum_{j=1}^{n} |K_i \cup H_j|^2.$$

It is possible to show that for any two clusterings $\kappa = \{C_1, \ldots, C_p\}$ and $\mu = \{D_1, \ldots, D_q\}$ a d-bimodulation can be defined by adding to each cluster C_i the set of objects located in the cluster D_j that has the largest intersection with C_i and applying a similar expansion to the clusters D_j.

The second method applied for cluster post-processing is using the minimum bounding hyper-rectangle $\mathsf{MBH}(C)$ of a cluster C. Suppose that:

$$\mathsf{MBH}(C) = [a_1, b_1] \times \cdots \times [a_r, b_r].$$

The *density of C* is defined as the number:

$$\mathsf{dens}(C) = \frac{|C|}{\mathsf{vol}(\mathsf{MBH}(C))}.$$

An *ϵ-expansion of C* is the set $C^\epsilon = C \cup L^\epsilon$, where

$$L^\epsilon = \mathsf{UNC}(\kappa) \cap ([a_1 - |a_1|\epsilon, b_1 + |b_1|\epsilon] \times \cdots \\ \cdots \times [a_r - |a_r|\epsilon, b_r + |b_r|\epsilon]).$$

If $C_i^\epsilon \cap C_j^\epsilon \neq \emptyset$, then we assign the points of K_ϵ to the cluster that has the larger density among the clusters C_i^ϵ or to C_j^ϵ.

Experimental results show that these post-processing techniques improve significantly the quality of the clustering; this is clearly visualized by the improvement of the quality of the image segmentation that we discuss in Section 4.3.

4 Experimental Results

We performed experimental work on three types of data: synthetic data consisting of randomly-generated points in $\mathbb{R}^n$, synthetic images containing colored regions randomly distributed, and, finally, real images. The implementations of k-means [Mac67] and DBSCAN [SEKX98] provided by the open-source WEKA package [WF05] were used for performance comparisons.

To evaluate the extent to which the algorithm retrieves the original clusters we computed several classification-oriented measure of cluster validity (see [TSK06], p. 549). We assume that we start with r clusters $K_1, \ldots, K_r$ and the clusters retrieved by the algorithm are C_j, where $1 \leq j \leq q$. Also, the probability that an object of the cluster C_j belongs to K_ℓ is the number $p_{j\ell} = \frac{|C_j \cap K_\ell|}{|D_j|}$, which is also known as the *precision* of C_j relative to K_ℓ.

4.1 Experiments on Synthetic Data

We tested our technique on a data set containing 10000 points in $\mathbb{R}^{30}$ distributed in four clusters: K_1, K_2, K_3, K_4. We recaptured a major part of the data set, as shown in Table 1.

C_i identified	K_1	K_2	K_3	K_4
C_1	0	54	0	0
C_2	0	305	0	0
C_3	0	272	0	0
C_4	0	0	0	274
C_5	0	0	0	1170
C_6	2	0	74	0
C_7	1103	0	0	0
C_8	138	0	0	0
UNC$(data)$	2094	1083	712	2146

Table 1. Intersections between initial clusters and retrieved clusters before postprocessing

C_i identified	K_1	K_2	K_3	K_4
C_1	0	514	2	0
C_2	0	312	0	0
C_3	0	801	9	0
C_4	0	0	0	1189
C_5	0	0	0	2930
C_6	30	85	770	41
C_7	1761	0	0	0
C_8	1543	0	0	0
UNC$(data)$	3	2	5	5

Table 2. Intersections between initial clusters and retrieved clusters after postprocessing

Table 1 represents the intersections between the original clusters $K_1, \ldots, K_4$ (which correspond to the columns of the table) with the clusters $C_1, \ldots, C_4$ obtained by our algorithm. The last row represents the points that the algorithm left outside the clusters. Before the postprocessing phase a substantial fraction of the points of the initial clusters

are placed into clusters (almost 50%); however, many points are left unaffiliated with any of the clusters. These points are classified using the second phase of the algorithm.

After bimodulation and a 5% expansion the data distribution looks as shown in Table 2.

We further tested our algorithm using a similar data set (10000 points in $\mathbb{R}^{30}$) and added 10% of noisy data. The results are shown in Table 3, which shows that the noise has little effect on the clusters.

Clusters identified	K_1	K_2	K_3	K_4	Noise
C_1	0	2885	3	0	8
C_2	0	2	803	0	3
C_3	0	0	0	1929	10
C_4	1056	0	0	0	14
Data outside clusters	328	1472	591	931	965

Table 3. Intersections between initial clusters and retrieved clusters after introduction of noise

We applied the k-means and the DBSCAN algorithms to the same data sets and obtained similar results for synthetic data without noise containing four clusters. Then, we tested these algorithms on a data set containing four clusters to which 10% noise was added. Several runs using the k-means algorithm with $k = 4$ result sometimes in having the noise distributed among each of the four clusters and, on occasions, producing 1 to 3 clusters with the remaining classes containing noise. The results of DBSCAN are similar to those of k-means; however, the misclassifications are much less frequent and the noise is well detected.

The application of the three algorithms to a database containing 11,000 objects in with 30 dimensions results in computation times of 750s, 1700 s, and 9s for our algorithm, the k-means algorithm and the DBSCAN algorithm, respectively. Our time is less than half of the DBSCAN. The k-means algorithm is much faster, but, as we shall see, has a rather bad precision and recall in experiments on synthetic images.

4.2 Experiments on Synthetic Images

In a second series of experiments we tested the algorithm on Mondrian-like images [Mon] containing randomly distributed and randomly colored rectangles. The typical images used in these experiments contained between 10 and 40 such regions and we show an example of an image in Figure 3.

The objectives of this series of experiments were to demonstrate that the algorithm can retrieve the original clusters and also, to test the behavior of the algorithm on data with a larger number of dimensions. Starting from an image containing $240 \times 320 = 76,800$ pixels represented as a set of points in $\mathbb{R}^5$ we grouped the pixels into 4×4 squares containing 16 pixels. Each square was represented as a vector in $\mathbb{R}^{80}$ and we worked with sets of 9600 points in $\mathbb{R}^{80}$.

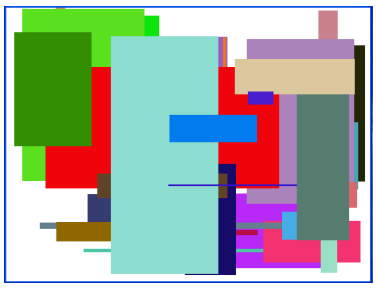

Fig. 3. Example of an image

True positive TP	Colored points retrieved as colored	63.29%
False positive FP	White points retrieved as colored	0.09%
True negative TN	White points retrieved as white	28.9%
False negative FN	Colored points retrieved as white	7.79%

Fig. 4. Terms used in algorithm evaluation

In a first phase we examined the capability of our algorithm to differentiate between the colored regions which we treat as clusters) and the white pixels. We are using the top five projections with an expansion factor of 5%. The terms used for this evaluation are shown in Figure 4. The precision and recall for this type of evaluations are given by

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = 0.99 \text{ and Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = 0.89,$$

respectively. The F_1 measure that is the harmonic average of precision and recall is 0.94. These numbers indicate a high capability of our algorithm in identifying points that belong to clusters.

The dependency of the precision, recall and F_1 measures are shown in Figures 5-7, respectively.

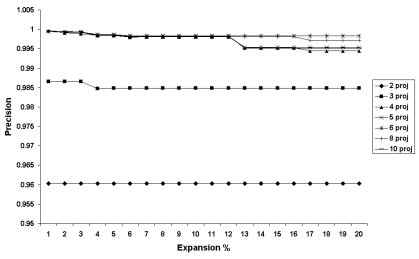

Fig. 5. Dependency of precision on ϵ

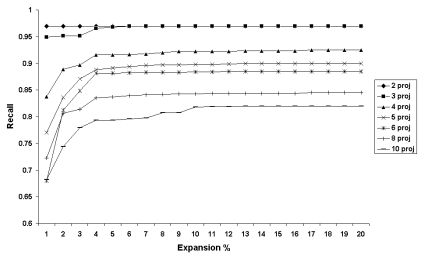

Fig. 6. Dependency of recall on ϵ

By contemplating these figures it becomes apparent that there is no substantial improvement of the recall or of the F_1 factor when expansion is greater than 6% and the number of projections considered is greater than 6 (see Figures 7 and 6). This observation informs the experiments described in the next section.

The time requirements of the algorithm were validated in experiments including data in $\mathbb{R}^{20}$ and in $\mathbb{R}^{80}$ (see Figure 8). The results shown represent averages over 4-

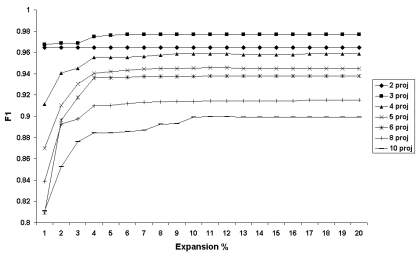

Fig. 7. Dependency of F_1 on ϵ

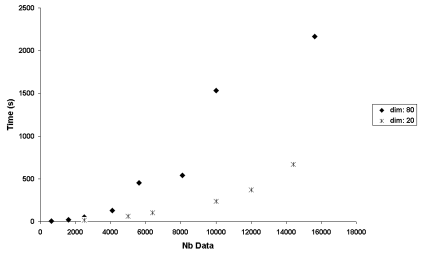

Fig. 8. Dependency of time (sec.) on the number of objects

fold complete applications of the algorithm including post-processing with different random projection frames. They are consistent with our previous asymptotic estimate of $O(N \log N)$.

In Table 4 we compare the precision, recall and F_1 measure for k-means, DBSCAN, and for our algorithm.

	Algorithm		
	k-means	DBSCAN	Our algorithm
TP (%)	62	58	63.3
FP (%)	28	3	0.1
TN (%)	10	34	28.9
FN (%)	1	5	7.8
Precision	0.69	0.95	0.99
Recall	0.98	0.92	0.89
F_1	0.81	0.94	0.94

Table 4. Time measures

For synthetic images the precision of the k-means algorithm is rather low even if k is chosen to obtain the best results. On the other hand, the results of DBSCAN and of our algorithm are comparable; we obtain a better precision but a lower recall which results in similar values for the F_1 measures.

The advantage of our algorithm over DBSCAN is a better time performance, which is more evident with the increase in the size of the data set.

4.3 Experiments on Real Images

To contemplate possible applications of our algorithm to multimedia data set we used the data set underlying the left picture shown in Figure 9. This data set contains 32,000 points in $\mathbb{R}^5$. The dimensions correspond to the two spatial coordinates and the three

color components of each pixel (red, green and blue). The two following images of the same figure correspond to two clusterings obtained using our random projection algorithm.

We have applied the bimodulation technique to the clusters contained in the second and third images of Figure 9 which consist of 10 and 12 clusters, respectively; the images that correspond to the resulting clusterings are shown in the fourth and fifth images of Figure 9. The clusterings shown in these two images consist of 11 and 13 clusters, respectively. One can visually remark the improvement of certain features shown in these clusters. However, an important fraction of the data points still remain unclassified; these unclassified data correspond to the white spots of the illustrations.

Finally, we used the ϵ-expansion of the minimally bounding rectangles of the clusters. The new clusters obtained by applying a 10% expansion are presented in the last two images shown in Figure 9.

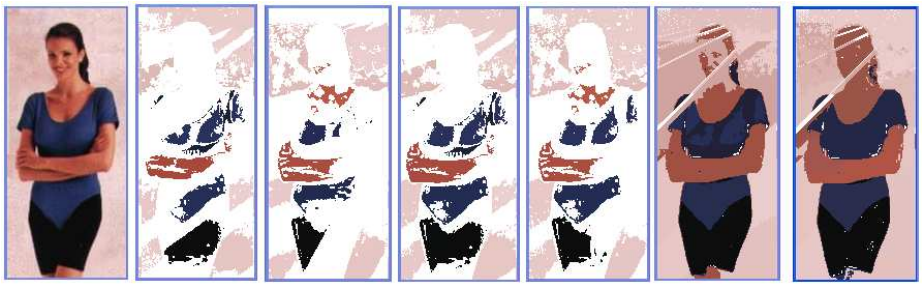

Fig. 9. Images obtained at different phases of our algorithm

The quality of the last two images is clearly improved over the others images. This observation suggests that our clustering techniques have the potential of being helpful in image segmentation .

5 Conclusions

We proposed a new method of clustering using random projections. The algorithm consists in two phases: a projection phase (which creates uni-dimensional histograms and aggregates these histograms to produce the initial clusters) and a post-processing phase that improves the clusterings using two supplementary techniques: bimodulation and ϵ-expansion. The time requirement of the algorithm is $O(N \log N)$, where N is the number of objects subjected to clustering. The algorithm has a potential for being useful for multimedia applications, which will be the focus of our future investigations.

We will investigate future directions of cluster post-processing as well as more refined ways of combining projection histograms. Finding optimal values for the parameters chosen in the execution of the algorithm based on the statistical distribution of the set of objects remains an open problem.

References

[AGGR98] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM-SIGMOD Int. Conf. Management of Data,*, pages 94–105, 1998.

[AM04] P. Agarwal and N. H. Mustafa. k-means projective clustering. In *Proceedings of PODS*, pages 155–165, 2004.

[APW+99] C. C. Aggarwal, C. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *Proceedings of ACM-SIGMOD Conference on Management of Data*, pages 61–72, 1999.

[BL95] J. P. Barthélemy and B. Leclerc. The median procedure for partitions. In *Partitioning Data Sets*, pages 3–14, Providence, RI., 1995. American Mathematical Society.

[CUDW02] A. B. Chaudri, R. Unland, C. Djeraba, and W.Lindner, editors. *XML-Based Data Management and Multimedia Engineering - EDBT 2002 Workshops*, volume LNCS 2490 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2002.

[DG99] S. Dasgupta and A. Gupta. An elementary proof of the johnson-lindenstrauss lemma. Technical Report TR-99-006, International Computer Science Institute, 1999.

[Dje03] C. Djeraba, editor. *Multimedia Mining - A Highway to Intelligent Multimedia Documents*. Kluwer, Boston, 2003.

[FM88] P. Frankl and H. Maehara. The johnson-lindenstrauss lemma and the sphericity of some graphs. *J. Comb. Theory B*, 44:355–362, 1988.

[JD88] A. K. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, NJ, 1988.

[JF96] A. K. Jain and P. J. Flynn. Image segmentation using clustering. In *Advances in Image Understanding: A Festschrift for Azriel Rosenfeld*, pages 65–83, Piscataway, NJ., 1996. IEEE Press.

[JL84] W. B. Johnson and J. Lindenstrauss. Extensions of lipshitz mappings into hilbert spaces. *Contemporary Mathematics*, 26:189–206, 1984.

[JMF99] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31:264–323, 1999.

[Mac67] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, Berkeley, 1967. University of California Press.

[Mon] P. Mondrian. http://artchive.com/artchive/M/mondrian.html.

[SEKX98] J. Sander, M. Ester, H. P. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Mining and Knowledge Discovery, an International Journal*, 2:169–194, 1998.

[TSK06] P. N; Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Pearson/Addison-Wesley, Boston, 2006.

[Vem04] S. S. Vempala. *The Random Projection Method*. American Mathematical Society, Providence, Rhode Island, 2004.

[WF05] I. H. Witten and E. Frank. *Data Mining - Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, second edition, 2005.

[ZSD02] O. R. Zaïane, S. Simoff, and C. Djeraba, editors. *Mining Multimedia and Complex Data*, volume LNAI 2797 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, 2002.