Linear manifold clustering in high dimensional spaces by stochastic search

Robert Haralick, Rave Harpaz*

Pattern Recognition Laboratory, CS Department, The Graduate Center, City University of New York, 365 Fifth Avenue, New York, NY 10016, USA

Abstract

Classical clustering algorithms are based on the concept that a cluster center is a single point. Clusters which are not compact around a single point are not candidates for classical clustering approaches. In this paper we present a new clustering paradigm in which the cluster center is a linear manifold. Clusters are groups of points compact around a linear manifold. A linear manifold of dimension 0 is a point. So clustering around a center point is a special case of linear manifold clustering. Linear manifold clustering (LMCLUS) identifies subsets of the data which are embedded in arbitrary oriented lower dimensional linear manifolds. Minimal subsets of points are repeatedly sampled to construct trial linear manifolds of various dimensions. Histograms of the distances of the points to each trial manifold are computed. The sampling corresponding to the histogram having the best separation between a mode near zero and the rest is selected and the data points are partitioned on the basis of the best separation. The repeated sampling then continues recursively on each block of the partitioned data. A broad evaluation of some 100 experiments over real and synthetic data sets demonstrates the general superiority of this algorithm over any of the competing algorithms in terms of accuracy and computation time. Its expected computational time is linearly proportional to the data set dimension and data set size. Its accuracy ranges from near 0.90 to 0.99 depending on the experiment and is generally much higher than the accuracy of the competing clustering algorithms.
© 2007 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

Keywords: Clustering; Linear manifold; Subspace; Histogram thresholding; Data exploration; Random projections

1. Introduction

Interest in clustering has increased substantially in recent years due to new areas of application such as data mining of customer databases, document clustering of web pages, image segmentation in computer vision, and DNA microarray analysis in bioinformatics. The problem of clustering can be loosely defined as the partitioning of a set of points in a multidimensional space into groups called *clusters* such that the points in each group are in some sense similar to one another. Finding these clusters is important because their points correspond to observations of different classes of objects that may have been previously unknown. A second kind of latent information that may be of interest are correlations in a data set. A correlation is a linear dependency between two or more features (attributes)

of the data set. Knowing about the existence of a relationship between features may enable us to learn hidden causalities, for example: the influence of the age of a patient and the dose rate of medication on the length of his disease; in recommendation systems for target marketing where groups of customers with similar behavior need to be detected, one searches for positive correlations; in gene expression analysis correlations express the fact that two genes may be coregulated.

Due to recent technological advances in data collection many applications of clustering are now characterized by high dimensional data which poses two challenges. First, the presence of non-information carrying features has the potential of eliminating any clustering tendency and mislead the clustering algorithm by masking clusters in noisy or irrelevant data. Second is the so called '*curse of dimensionality*', which suggests that points tend to become equidistant from one another as dimension increases [1], and therefore learning structure in high dimensional spaces by distance measures that utilize the full set of dimensions becomes increasingly difficult.

* Corresponding author. Tel.: +1 212 817 8192; fax: +1 212 817 1510.

E-mail addresses: haralick@ptah.gc.cuny.edu (R. Haralick), rbharpaz@sci.brooklyn.cuny.edu (R. Harpaz).

Conventional clustering algorithms such as K-Means [2], and DBSCAN [3] are "full-dimensional" in the sense that they give equal relevance to all dimensions, and therefore are likely to fail when applied to high dimensional data, particularly when some of the data dimensions are irrelevant. Initial attempts to tackle this problem involved different types of dimensionality reduction techniques such as *feature selection* and *feature transformation* techniques as preprocessing steps. Feature selection attempts to discover the most relevant attributes, while feature transformation techniques such as principle component analysis (PCA) transform the original space into a lower dimensional space. While effective in reducing the dimensionality of the data, these methods are limited in that they can only be applied to the data set as a whole, and therefore are only capable of detecting structure which is expressed in the data as a whole. However, since different clusters may exist in different subsets of the features, removing dimensions globally is likely to incur a loss of crucial information. Thus, only methods that localize the search for relevant features cluster by cluster are likely to succeed.

An important advance in this area was the introduction of subspace clustering. Subspace clustering is considered an extension to traditional clustering and feature selection techniques [4], in that it attempts to find different clusters embedded in different subspaces of the same data set. A subspace cluster consists of a subset of points and a corresponding subset of attributes, such that these points form a dense region in a subspace defined by the set of corresponding attributes. Most subspace clustering methods such as CLIQUE [5], MAFIA [6], and PROCLUS [7] are restricted to finding clusters in subspaces spanned by some subset of the original measurement features. However, examination of real data often shows that points tend to get aligned along arbitrarily oriented subspaces, which are merely instances of linear manifolds.[1] ORCLUS [8] the most relevant algorithm to the problem of linear manifold clustering is an extension to PROCLUS which allows clusters to exist in arbitrarily oriented subspaces. It is a K-Means like algorithm that uses PCA to peel off noisy dimensions and find projected clusters. Its main drawback like many similar algorithms is that it requires the user to specify the number of clusters and the dimensionality of the subspaces in advance which makes it impractical for real applications. A potential solution to the problem of determining the number of clusters in advance is the use of *visual assessment* techniques such as VAT, reVAT, and bigVAT [9]. This family of techniques takes as input a similarity/distance matrix between all pairs of points, typically constructed using the L2-norm, and outputs an image matrix to reveal underlying cluster tendency. However, because these methods measure similarity in the full space, they are unlikely, as do the "full-dimensional" clustering algorithms, to reveal any clustering tendency when the clusters are embedded in subspaces or linear manifolds, unless the clusters are well separated.

The model of subspace-clusters may not be adequate to capture correlations in the data as it only considers the physical distance between points. A set of points may be located far away from each other yet induce large correlations among some subset of features, e.g. a set of points embedded in a line. The main motivation to explore a more general type of subspace clustering known as *pattern* or *correlation clustering* came from the need to identify pattern similarities in the analysis of DNA microarrays. In DNA microarray clustering the goal is to identify groups of genes (clusters) whose expression levels rise and fall coherently under a subset of conditions (features). Among the first to discuss these types of similarities in the context of DNA microarray clustering were Cheng et al. [10] who introduced the *bicluster* concept, and Yang et al. [11] who presented the move-based algorithm FLOC to find biclusters a.k.a. δ-clusters more efficiently. Both of these methods and other similar ones focus on two forms of coherence or patterns, the *shifting* (addition) and *amplification* (multiplication) patterns which induce positive correlations only. Negative or correlations induced by a transformation of the original measurement features are not considered by these methods. In a recent paper Harpaz and Haralick [12] demonstrated how the shifting and amplification patterns along with patterns that induce negative or more complex correlations can be generalized to linear manifolds, making the paradigm of linear manifold clustering applicable also to the problem of pattern/correlation clustering.

Dasgupta [13] presents two important results related to *random projections* which have implications to clustering in high dimensional spaces. These results show that it is possible to project high dimensional data into substantially lower dimensions while still retaining the approximate level of separation between clusters. Exploiting these results Haralick et al. [14] presented a hierarchical projection pursuit clustering (HPPC) algorithm that repeatedly bi-partitions a data set by searching for separations in 1D projections of the data. The output of HPPC is a decision tree whose nodes store a projection and a separating threshold, and whose leaves represent the clusters.

In this paper we present a new clustering paradigm in which the cluster center is linear manifold. Clusters are groups of points compact around a linear manifold. A linear manifold of dimension zero is a point. So clustering around a center point is a special case of linear manifold clustering. A linear manifold of dimension one is a line. Clusters may be oriented around a line when the random perturbation affecting the process that produces the data of the cluster has a covariance matrix of rank one. A linear manifold of dimension two is a plane. Clusters may be oriented around a plane when the random perturbation affecting the process that produces the data of the cluster has a covariance matrix of rank two.

The remainder of the paper is organized as follows: In Section 2 we formalize the linear manifold cluster model. Based on this model, we present in Section 3 the algorithm-LMCLUS, followed in Section 4 by an asymptotic time complexity analysis. In Section 5 we present a broad evaluation of LMCLUS applied on synthetic and real data sets, and in Section 6 we conclude the paper giving hints on future work.

---

[1] A linear manifold is a translated subspace. A subspace is a subset of points closed under linear combination.

2. The linear manifold cluster model

A linear manifold is a subspace that may have been translated away from the origin. A subspace is a special case of a linear manifold that contains the origin. Geometrically, a 1D manifold can be visualized as a line embedded in the space, a 2D manifold as a plane, and a 0D manifold as a point. Classical clustering algorithms such as K-Means assume that each cluster is associated with 0D manifold (a point typically the cluster center), and therefore omit the possibility that a cluster may have a non-zero dimensional linear manifold associated with it. For the sake of completeness we give a formal definition of a linear manifold.

Definition 1 (*Linear manifold*). L is a *linear manifold* of vector space V if and only if for some subspace S of V and translation $t \in V$, $L = \{x \in V |$ for some $s \in S, x = t + s\}$. The dimension of L is the dimension of S, and if the dimension of L is one less than the dimension of V then L is called a *hyperplane*. A linear manifold L is rectangularly bounded if and only if for some translation t and bounding vectors a_L and a_H, $L = \{x \in V |$ for some $s \in S, a_L \leqslant s \leqslant a_H, x = t + s\}$. A rectangularly bounded linear manifold has finite extent and is localized with center $t + (a_L + a_H)/2$. In the case that $a_L = -a_H$, its center is the translation t.

The linear manifold cluster model has the following properties: the points in each cluster are embedded in a lower dimensional linear manifold of finite extent. The intrinsic dimensionality of the cluster is the dimensionality of the linear manifold. The manifold is arbitrarily oriented. The points in the cluster induce a correlation among two or more attributes (or a linear transformation of the original attributes) of the data set. In the orthogonal complement space to the manifold the points form a compact densely populated region. More formally let D be a set of d-dimensional points, $C \subseteq D$ be the subset of points that belong to a cluster, $\mathbf{x}$ be a $d \times 1$ vector representing some point in C, $\mathbf{b}_1, \ldots, \mathbf{b}_d$ be a set of orthonormal vectors that span a d-dimensional space, B be a $d \times k$ matrix whose k columns are a subset of the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d$, and $\overline{B}$ be a $d \times d - k$ matrix whose columns are the remaining vectors.

Definition 2 (*Linear manifold cluster model*). Let μ be some point in $\mathbb{R}^d$, λ be a zero mean $k \times 1$ random vector whose entries are i.i.d. $U(-R/2, +R/2)$ where R is the range of the data, and ψ be a zero mean $d - k \times 1$ random vector with small variance independent of λ. Then each $\mathbf{x} \in C$, a linear manifold cluster is modeled by,

$$\mathbf{x} = \mu + B\lambda + \overline{B}\psi. \tag{1}$$

The idea is that each point in a cluster lies close to a k-dimensional linear manifold of finite extent, which is defined by μ, a translation vector, the space spanned by the columns of B, and the range parameter R. Since

$$E[\mathbf{x}] = E[\mu + B\lambda + \overline{B}\psi] = \mu + BE[\lambda] + \overline{B}E[\psi]$$
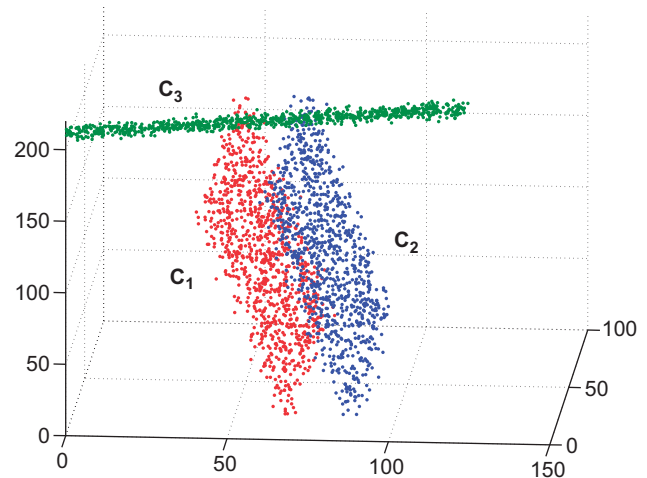$$= \mu + \mathbf{0} + \mathbf{0} = \mu$$

Fig. 1. Sample data set of three non-overlapping clusters, each of which is embedded in a different linear manifold of one (C_3) or two dimensions (C_1, C_2).

the cluster mean is μ. On the manifold the points are assumed to be uniformly distributed in each direction (the k column vectors of B) according to $U(-R/2, +R/2)$. However this assumption is not binding, and the uniform distribution can be replaced by any other distribution with symmetric support. It is in this manifold that the cluster is embedded, and, therefore, the intrinsic dimensionality of the cluster will be k. What characterizes this type of cluster is the third component that models a small random error associated with each point on the manifold. The idea is that each point may be perturbed in directions that are orthogonal to the subspace spanned by the columns of B, that is the subspace defined by the $d - k$ columns of $\overline{B}$. We model this behavior by requiring that ψ be a $(d - k) \times 1$ random vector, normally distributed according to $N(\mathbf{0}, \Sigma)$, where the largest eigenvalue of Σ is much smaller than R, the range of the data. Otherwise the signal cannot be distinguished from the noise. In addition since the variance along each dimension orthogonal to the manifold is now much smaller than the range R of the manifold, the points are likely to form a compact and densely populated region that can be used to cluster the data. The concept of the error term can be visualized by a 1D manifold which transforms from a line into a thin cylinder after the addition of an error term.

Traditional "full-space" clustering algorithms take $k = 0$, and therefore assume that each point in a cluster can be modeled by $\mathbf{x} = \mu + \overline{B}\psi$ where $\overline{B}$ is simply the identity matrix. Subspace clustering algorithms focus their clustering effort on the space spanned by the column vectors of $\overline{B}$, and when restricted to axis parallel subspaces, they assume both B and $\overline{B}$ contain columns of the identity matrix.

Fig. 1 is an example of data set modeled by Eq. (1). The data set contains three non-overlapping clusters C_1, C_2, C_3 each consisting of 1000 points. C_1, C_2 which are almost planar and parallel to each other are embedded in 2D linear manifolds. Their points are uniformly distributed in the manifold and they include a small error term in the space complementary to the

manifold. Similarly, C_3 an elongated line-like cluster, is embedded in a 1D linear manifold with an error element in the 2D space complementary to the manifold.

3. The algorithm

LMCLUS can be viewed as a hierarchical-divisive clustering procedure. It executes three levels of iteration (Fig. 2), and expects three inputs: K, an upper limit on the dimension of the linear manifolds in which we believe clusters may be embedded; S, a sampling level parameter used to determine the number of trial linear manifolds of a given dimensionality that will be examined in order to reveal the best possible partitioning of a given set of points; Γ, a sensitivity or "goodness of separation" threshold, which is used to determine whether or not a partitioning should take place based on a trial linear manifold.

At the highest level of iteration the algorithm monitors the size of the data which is being partitioned. When no data is left to be partitioned the algorithm terminates. The second level of iteration causes the algorithm to iterate over a range of manifold dimensionalities, commencing with 1D manifolds, and terminating with K-dimensional manifolds, where K is an input parameter. For each linear manifold dimension the algorithm enters the third level of iteration, in which *FindSeparation* (Fig. 3) is invoked in an attempt to reveal separations among subsets of the data and to determine whether some of the points are embedded in linear manifolds. The idea behind *FindSeparation* is to successively sample points that can define a linear manifold of a given dimension, and select the linear manifold that is closest to a substantial number of points. This subset of closest points will typically correspond to a cluster. The proximity of the input data points to the manifold is captured by a distance histogram. If the manifold indeed has some subset of points near it, then the distance histogram will have a mixture of two distributions. One of the distributions has a mode near zero and is the distribution of distances of points belonging to a cluster. The other distribution is the distribution of the remaining points. The problem of separating the cluster points from the rest is then cast into a histogram thresholding problem. Upon termination *FindSeparation* returns four values γ—which is a measure of the "goodness" of the separation, τ—a proximity threshold that is computed from the histogram and is used to split the data into two groups, β—the basis of the manifold which exposes the separation, and ϕ—a point on the manifold representing its origin. When γ exceeds the value of the input sensitivity threshold parameter Γ, indicating that a worthy separation has been found, then the data set is split according to τ. This split corresponds to the partitioning of all the points which are located close enough to the just determined manifold, i.e. all points that potentially belong to a given cluster, and those that belong to other clusters. In addition the dimension of the manifold which revealed the separation is recorded by assigning its value to *LmDim*. The third iteration continues reapplying *FindSeparation* in an attempt to further partition the cluster which may consist of sub-clusters, until the selected data points cannot be further separated. At this point the algorithm will

retract to the second level of iteration in an attempt to partition the cluster in higher dimensions, a process which will continue until the dimension limit K is reached. When K is reached we have a subset of the points that cannot be partitioned any more, and declare that a cluster is found. This cluster is labeled and added to the set of found clusters along with its dimensionality recorded by *LmDim*. The algorithm then retracts to the first level of iteration and is reapplied on the remaining set of points until no more points are left to be partitioned, detecting one cluster at a time. We note that if outliers exist then the last cluster/partition that is found will contain this set of points. By definition outliers do not belong to any cluster and, therefore, will remain the last group of points to be associated to any other group. Since they are unlikely to form any clusters the algorithm will not be able to partition them, and they will therefore be all grouped together.

3.1. Finding separations

Let D, C, B, $\overline{B}$, and μ be as defined in Section 2. The distance of a point $\mathbf{y}$ in D to a linear manifold defined by μ and the column vectors of B is given by

$$\|(I - BB^{\mathrm{T}})(\mathbf{y} - \mu)\| = \|\overline{B}\,\overline{B}^{\mathrm{T}}(\mathbf{y} - \mu)\|. \tag{2}$$

As mentioned earlier in Section 2 the points of C are likely to form a compact and dense region in the space orthogonal to the manifold in which they are embedded. Therefore, by projecting D into the space spanned by the column vectors of $\overline{B}$ and executing some form of clustering in the reduced space it is possible to identify and separate C from the rest of the data. However, Eq. (2) shows that the distance of a point to the cluster center in the reduced space is equivalent to the distance of a point to the linear manifold. Thus, rather than clustering in the reduced space it is also possible to measure distances from the manifold and collect all the points that lie in the vicinity of this manifold, essentially executing 1D clustering. Since we are interested in estimating B, and because we are interested in detecting one cluster at a time, and since 1D clustering is typically faster than clustering in higher dimensions, we choose to take this path.

Lemma 1. $\|(I - BB^{\mathrm{T}})(\mathbf{y} - \mu)\| = \sqrt{\|(\mathbf{y} - \mu)\|^2 - \|B^{\mathrm{T}}(\mathbf{y} - \mu)\|^2}.$

Proof. Let $\mathbf{z} = \mathbf{y} - \mu$,

$$\begin{aligned}
\|(I - BB^{\mathrm{T}})\mathbf{z}\|^2 &= \|\mathbf{z} - BB^{\mathrm{T}}\mathbf{z}\|^2 \\
&= (\mathbf{z} - BB^{\mathrm{T}}\mathbf{z})^{\mathrm{T}}(\mathbf{z} - BB^{\mathrm{T}}\mathbf{z}) \\
&= \mathbf{z}^{\mathrm{T}}\mathbf{z} - 2\mathbf{z}^{\mathrm{T}}BB^{\mathrm{T}}\mathbf{z} - \mathbf{z}^{\mathrm{T}}(BB^{\mathrm{T}})^2\mathbf{z} \\
&= \mathbf{z}^{\mathrm{T}}\mathbf{z} - \mathbf{z}^{\mathrm{T}}BB^{\mathrm{T}}\mathbf{z} \\
&\quad (BB^{\mathrm{T}} \text{ is idempotent so } (BB^{\mathrm{T}})^2 = BB^{\mathrm{T}}) \\
&= \|\mathbf{z}\|^2 - \|B^{\mathrm{T}}\mathbf{z}\|^2. \quad \square
\end{aligned}$$

Lemma 1 provides us a much more efficient way of computing the distance of a point to a manifold. If d is the

Algorithm $LMCLUS(dataset: D, \ max \ LM \ dim: K, \ sampling \ level: S, \ sensitivity \ threshold: \Gamma)$
 $C := \emptyset;$ # *list of labeled clusters initially empty.*
 $Dims := \emptyset;$ # *list of intrinsic dimensionalities of each cluster.*
 $i := 1;$ # *cluster label*
 while $D \neq \emptyset$ **do**
  $D' := D, \ LmDim := 1;$
  for $k := 1$ **to** K **do**
   while $[\gamma, \ \tau, \ \o, \ \beta] := FindSeparation(D', \ k, \ S), \ \gamma > \Gamma$ **do**
    # *a separation is revealed by a k-dimensional*
    # *manifold. Collect all points residing in the vicinity*
    # *of that manifold.*
    $D' := \{x| \ x \in D', \| \ (x - \o) \ \|^2 - \| \ \beta^T(x - \o) \ \|^2 < \tau\};$
    $LmDim := k;$
  # *a cluster is found, add it to the list of labeled clusters.*
  $c_i := D', \ C := C \cup \{c_i\};$
  # *recored the intrinsic dimensionality of the cluster*
  $dim_i := LmDim, \ Dims := Dim \cup \{dim_i\};$
  $i := i + 1;$
  # *remove cluster points from the dataset.*
  $D := D - D';$
 return $C, \ Dims;$

Fig. 2. The clustering algorithm.

Algorithm $FindSeparation \ \{dataset: \ D, \ dimension: k \ sampling \ level: S\}$
 $\gamma := -\infty, \ \tau := -\infty, \ \o = \mathbf{0}, \ \beta := \mathbf{0};$
 $N := min\{\log \epsilon / \log(1 - (1/S)^k), c|D|\};$
 for $i = 1$ **to** N **do**
  $M := Sample \ (k + 1) \ points \ from \ D;$
  $O := x \in M$
  $B := FormOrthonormalBasis(M, \ O);$
  $Distances := \emptyset;$
  for $each \ x \in D - M$ **do**
   $x' := x - O;$
   $Distances := Distances \cup \{\| \ x' \ \|^2 - \| \ B^T x' \ \|^2\};$
  $H := MakeHistogram(Distances);$
  $T := FindMinimumErrorThreshold(H);$
  $G := EvaluateGoodnessOfSeparation(T, \ H);$
  if $G > \gamma$ **then**
   $\gamma := G, \ \tau := T, \ \o := O, \ \beta := B;$
 return $[\gamma, \ \tau, \ \o, \ \beta];$

Fig. 3. Detecting separations among clusters embedded in lower dimensionality linear manifolds.

dimension of the data, then computing the distance using Lemma 1 gives us a speedup of $O(d)$, which for high dimensional data becomes a significant factor. To simplify the computation even further we choose to use the squared distance rather than the distance, henceforth we will use the term "distance" to mean the squared distance.

3.1.1. Minimum error thresholding

The problem of separating all the points that lie in the vicinity of a manifold can be cast into the problem of finding a *minimum error threshold* that is used to classify points as either embedded in the manifold (belonging to C) or not, based on their distances to the manifold. Kittler and Illingworth [15] (KI) describe an efficient method for finding the minimum error threshold. Their method was designed for segmenting an object from its background in gray scale images using a gray level histogram of the image. Their method views the histogram segmentation problem as a two class classification problem, where the goal is to minimize the number of misclassified pixels. Analogous to our problem, the distance histogram can be viewed as an estimate of the probability density function of the mixture population comprising of distances of points belonging to a linear manifold cluster and those that do not.

The KI procedure is based on the assumption that each component of the mixture is normally distributed. To support this assumption in our case, we note that as a consequence to the central limit theorem, the distances to the manifold which are merely sums of random variables will approach distribution-wise the normal, as the dimension of the space increases. Additional support was given by Diaconis and Freedman [16], who showed that most projections of high dimensional data to lower dimensions will be approximately normal.

Let δ be the distance of a point to the manifold, $p(\delta|i)$ be the probability density function of the distances of class i, where $i \in \{1, 2\}$, μ_i, σ_i be the mean and standard deviation of distances in class i, and P_i be the prior of class i. Then because of the normality assumption

$$p(\delta|i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(\frac{-(\delta - \mu_i)^2}{2\sigma_i^2}\right).$$

Given μ_i, σ_i, P_i, and $p(\delta|i)$ there exists a threshold τ such that

$$P_1 p(\delta|1) > P_2 p(\delta|2) \ \text{if} \ \delta \leqslant \tau \quad \text{and}$$

$$P_1 p(\delta|1) < P_2 p(\delta|2) \ \text{if} \ \delta > \tau,$$

where τ is the Bayes minimum error threshold [17], which can be found by solving for δ the following equation:

$$P_1 \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left(\frac{(\delta - \mu_1)^2}{-2\sigma_1^2}\right) = P_2 \frac{1}{\sqrt{2\pi}\sigma_2} \exp\left(\frac{(\delta - \mu_2)^2}{-2\sigma_2^2}\right).$$

However, the true values of μ_i, σ_i, P_i are usually unknown. KI propose to obtain these estimates from the distance histogram h. Suppose that the histogram is thresholded at an arbitrary threshold t, then we can model the two resulting populations by a normal density $h(\delta|i, t)$ with parameters:

$$P_i(t) = \sum_{\delta=a}^{b} h(\delta), \quad \mu_i(t) = \frac{\sum_{\delta=a}^{b} \delta * h(\delta)}{P_i(t)},$$

$$\sigma_i^2(t) = \frac{\sum_{\delta=a}^{b} (\delta - \mu_i(t))^2 * h(\delta)}{P_i(t)},$$

where $a=0$ and $b=t$ if $i=1$, and $a=t+1$ and $b=max(\delta)$ if $i=2$. Now using the model $h(\delta|i, t)$ for $i \in \{1, 2\}$, the conditional probability of δ being correctly classified is given by

$$p(i|\delta, t) = \frac{h(\delta|i, t) P_i(t)}{h(\delta)}.$$

We wish to find the threshold t that maximizes this probability. Since $h(\delta)$ is independent of i and t it can be safely ignored. Furthermore, since the logarithm is a strictly increasing function, taking the logarithm and multiplying by a constant will not change the maximizing value. Therefore,

$$\varepsilon(\delta, t) = \left(\frac{\delta - \mu_i(t)}{\sigma_i(t)}\right)^2 + 2\log \sigma_i(t) - 2\log P_i(t),$$

can be considered as an alternative index of the correct classification performance, and the overall performance is given by

$$J(t) = \sum_{\delta} h(\delta)\varepsilon(\delta, t),$$

which reflects indirectly the amount of overlap between the two Gaussian populations. Substituting $\mu_i(t)$, $\sigma_i(t)$, $P_i(t)$, and $\varepsilon(\delta, t)$ into $J(t)$ we get

$$J(t) = 1 + 2(P_1(t)\log \sigma_1(t) + P_2(t)\log \sigma_2(t))$$
$$\quad - 2(P_1(t)\log P_1(t) + P_2(t)\log P_2(t)) \quad (3)$$

and the minimum error threshold selection problem can be formulated as

$$\tau = \arg \min_t J(t).$$

$J(t)$ can be computed easily and finding its minima is a relatively simple task as the function is smooth. We note that the tails of the distribution have been truncated by the thresholding operation and therefore the models $h(\delta|i, t)$, $i \in \{1, 2\}$ will be biased estimates of the true mixture components.

Cho et al. [18] proposed an improvement of KI's criterion function that corrects the biased variance estimates.

3.2. Sampling linear manifolds

A line, which is a 1D linear manifold, can be defined by two points, a plane which is a 2D manifold can be defined using three points. To construct a random k-dimensional linear manifold by sampling points from the data we need to sample $k + 1$ linearly independent points. Let $\mathbf{x}_0, \ldots, \mathbf{x}_k$ denote these points. Choosing one of the points, say $\mathbf{x}_0$ as the origin, the k vectors spanning the manifold are given by

$$\mathbf{y}_i = \mathbf{x}_i - \mathbf{x}_0,$$

where $i = 1 \ldots k$. Assuming each of these sampled points came from the same cluster, then according to Eq. (1)

$$\mathbf{y}_i = \mathbf{x}_i - \mathbf{x}_0 = (\mu_0 + B\lambda_i + \overline{B}\psi_i) - (\mu_0 + B\lambda_0 + \overline{B}\psi_0)$$
$$= B(\lambda_i - \lambda_0) + \overline{B}(\psi_i - \psi_0).$$

If the cluster points did not have an error component off the manifold, i.e. they all lie on the linear manifold, then sampling any $k + 1$ points which are linearly independent and belong to the same cluster would enable us to reconstruct B. So in order to get a good approximation of B we would like each of the sampled points to come from the same cluster and to be as close as possible to the linear manifold. From the above equation we see that this occurs when

$$\psi_i - \psi_0 \approx \mathbf{0},$$

resulting in a set of k vectors which are approximately a linear combination of the original vectors in B. A good indication as to why this is likely to occur when the sampled points come from the same cluster, is given by the fact that

$$E[\psi_i - \psi_0] = \mathbf{0},$$

and that normally distributed data ($\psi_i - \psi_0$ follows a normal distribution) tends to cluster around its mean. In cases where the clusters are well separated, the requirement that $\psi_i - \psi_0 \approx \mathbf{0}$ can be relaxed. That is, when the clusters are well separated more sets of points coming from the same cluster, and not only those that are relatively close to the manifold will be good candidates to construct a manifold that will induce a large valley in the distance histogram that separates the linear manifold cluster from the remaining points. As a consequence, the problem of sampling a linear manifold that will enable us to separate a linear manifold cluster from the rest of the data can be reduced to the problem of sampling $k + 1$ points that all come from the same cluster.

Assuming there are S clusters in the data set whose size is distributed with low variance, then for large data sets the probability that a sample of $k + 1$ points all come from the same cluster is approximately

$$\left(\frac{1}{S}\right)^k.$$

If we want to ensure with probability $(1 - \varepsilon)$ that at least one of our random samples of $k + 1$ points all come from the same cluster, then we should expect to make at least n selections of $k + 1$ points, where

$$\left[1 - \left(\frac{1}{S} \right)^k \right]^n \leqslant \varepsilon,$$

yielding that

$$n \geqslant \frac{\log \varepsilon}{\log(1 - (1/S)^k)}. \tag{4}$$

Therefore, by computing n given ε, and S which is an input to the algorithm we can approximate a lower bound on the number of samples required, such that with high probability, at least one of the n samples contains $k + 1$ points that all come from the same cluster. Unlike K-Means and ORCLUS, the user input S does not predetermine the number of clusters LMCLUS will output. It is a rough estimate of the number of clusters in the data set, which is only used to compute an initial estimate of the sample size required to ensure we sample points coming from the same cluster. It is rather a "gauge" with which we can tradeoff accuracy with efficiency.

The sample size n grows exponentially with the dimensionality of the linear manifold k which is being sampled. Although LMCLUS is not designed to identify clusters embedded in linear manifolds of high dimension, it is possible to bound the sample size from above. Dasgupta and Gupta [19] showed that using random projections it is possible to project high dimensional data into a low dimension subspace and preserve distances up to a factor of $1 + \varepsilon$ with probability $1/|D|$, where $|D|$ is the size of the data set. This in turn means that by performing $O(|D|)$ random projections it is possible to find a projection that will be able to retain a good level of separation between clusters with high probability. Combining this result with the sample size computed in Eq. (4) the number of samples N which will be drawn can be determined by the following heuristic:

$$N = \min(\log \varepsilon / \log(1 - (1/S)^k), c|D|), \tag{5}$$

where c is some constant independent of $|D|$ which can simply be set to equal one, making $|D|$—the size of the data being clustered an alternative sample size approximation. We note that for small k and large data sets the sample size computed by Eq. (4) is likely to be much smaller than $|D|$, and will therefore be preferred.

Putting it all together, for each sample of points $\{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$ we construct an orthonormal basis B of a linear manifold using the Gram–Schmidt process. (If the Gram–Schmidt process indicates that the sampled points are not linearly independent, a new sample of points is taken.) Then using KI's method we compute a threshold τ. Of all possible thresholds corresponding to different linear manifolds which induce different separations, we prefer the one which induces the best separation. The best separation is defined as the separation which induces
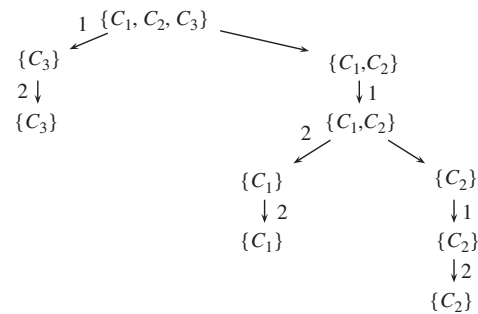

Fig. 4. A dendrogram summarizing the clustering process of the sample data set from Fig. 1. The labels on the arrows specify the dimension of the linear manifold which was used to separate the clusters.

the largest discriminability given by

$$discriminability = \frac{(\mu_1(\tau) - \mu_2(\tau))^2}{\sigma_1(\tau)^2 + \sigma_2(\tau)^2} \tag{6}$$

and the one which causes the deepest broadest minimum in KI's criterion function—$J(t)$. The deepest minimum can be quantified by computing the difference/depth of the criterion function evaluated at the minimum τ and the value evaluated at the closest local maxima τ', i.e.

$$depth = J(\tau') - J(\tau).$$

The composite measure of the goodness of a separation is then given by

$$G = discriminability \times depth. \tag{7}$$

A typical run of the algorithm is illustrated in Figs. 4 and 5 by a dendrogram which summarizes the clustering process of the sample data set depicted in Fig. 1, and the corresponding histograms that were used to separate the linear manifold clusters in this data set. At the beginning of the process the algorithm searches for 1D linear manifolds in which some clusters may be embedded. Since cluster C_3 is such a cluster it is separated from C_1, C_2 using the threshold returned by KI's procedure, and the algorithm proceeds by trying to further partition it in higher dimensions. Since it cannot be further partitioned using 2D manifolds, cluster C_3 is declared to be found. The algorithm then attempts to separate the remaining clusters C_1, C_2 using 1D manifolds. Since both these clusters are embedded in 2D linear manifolds the algorithm will fail. However by trying to separate them using 2D manifolds the algorithm will succeed. At this point the algorithm will attempt to further partition each of the clusters C_1 and C_2, however, since they are inseparable C_1 and C_2 are declared to be found, and the algorithm terminates.

4. Complexity analysis

Because LMCLUS uses sampling, complexity analysis of its running time is not straightforward. As a consequence the following result is only a worst case upper bound on its running
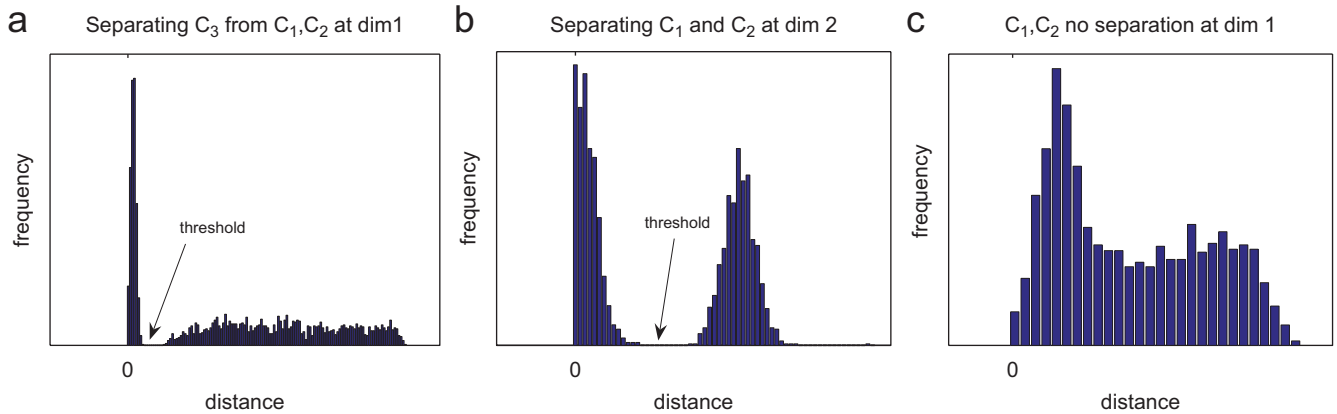
Fig. 5. Histograms used to separate the clusters from Fig. 1: (a) C_3 is separated from C_2 and C_3 by sampling 1D linear manifolds; (b) C_1 is separated from C_2 by sampling 2D linear manifolds; (c) a histogram for which no separation can be found.

time, and does not reflect its performance in practice as demonstrated by the experiments presented in Section 5.

Lemma 2. *The overall worst case time complexity (an upper bound) of LMCLUS in terms of the size of the data-N, the dimensionality of the data-D, the largest dimension of the linear manifolds in which clusters are embedded-K, and the number of clusters which are detected-C, is $O(N^2C^2K^3D)$.*

Proof. The overall complexity of LMCLUS depends the complexity of the *FindSeparation* procedure, and on the number of calls to it. Starting with the *FindSeparation* procedure, each set of $k + 1$ points that are sampled in order to construct a k-dimensional manifold must first be transformed to a set of k orthonormal vectors (*FormOrthonormalBasis*) by a Gram–Schmidt process whose complexity is $O(k^2d)$, where d is the dimensionality of the data. Measuring distances to the manifold amounts to a projection whose complexity is $O(kd)$. Assuming there are n points in data, the overall complexity of this step is $O(nkd)$. Making a distances histogram requires $O(n)$ operations (assigning each distance to a bin). Assuming the number of bins in the histogram is bound by some constant, then finding the minimum error threshold by Kittler and Illingworth's method has complexity $O(1)$ (constant time). This is because the complexity of this procedure only depends on the number of bins in the histogram. Each of the steps described above must be repeated for each linear manifold that is sampled. Since the number of linear manifolds sampled is bounded from above by the number of points that are to be clustered (Eq. (5)), the overall complexity of *FindSeparation* is $O(nk^2d + n^2kd + n^2 + n)$.

We now need to determine the number of calls to the *FindSeparation* procedure. Assuming the clusters are separable, then in the worst case each cluster is found at the highest dimension which is K, that is, up to dimension $K - 1$ the data is inseparable, and at dimension K, a cluster is found only after repeated calls to *FindSeparation* that gradually peals off other clusters one at a time. If there are C clusters in the data, of

approximately the same size, then the overall complexity of LMCLUS as a function of N, C, K, D is given by

$$\sum_{c=1}^{C} \left[\sum_{k=1}^{K-1} \frac{cN}{C} \left(k^2D + \frac{cNkD}{C} + \frac{cN}{C} + 1 \right) \right.$$
$$\left. + \sum_{i=1}^{c} \frac{iN}{C} \left(K^2D + \frac{iNKD}{C} + \frac{iN}{C} + 1 \right) \right]$$
$$= O(N^2C^2K^3D).$$

This result shows that LMCLUS's running time is only linear in the dimensionality of the data and cubic in the dimensionality of the linear manifolds in which clusters are embedded. Related methods such as ORCLUS are cubic in the dimensionality of the data due to the computation of principle components, or like CLIQUE, exponential in the dimension of subspaces in which clusters are embedded. We also note that although the worst case complexity of LMCLUS is quadratic in the number of points, in practice when the dimension of the linear manifolds is much smaller than the size of the data set, the sample size computed by Eq. (5) will be much smaller than size of the data, resulting in an algorithm whose complexity is only linear in the size of the data.

5. Empirical evaluation

In this section we present a broad evaluation of LMCLUS applied on synthetic and real data sets. LMCLUS as well as three other related methods: DBSCAN a representative of the full-dimensional clustering methods, ORCLUS a representative of subspace clustering methods, and HPPC a projection pursuit based clustering method, were implemented in C + +. The experiments were performed on a Linux based workstation with an Intel P4 3.0 GHz CPU and 1 GB of memory. The aim of the experiment was to evaluate LMCLUS's performance in comparison to the other methods with respect to accuracy, efficiency, scalability and its stability as a stochastic algorithm.

5.1. Synthetic data generation

Several dozen data sets were generated according to the linear manifold cluster model specified in Eq. (1). The underlying idea is to first generate each cluster in an axis parallel manner centered at the origin, and then randomly translate and rotate it, where the rotation is used to achieve the effect of an arbitrary orientation of the cluster in the space. In addition to the clusters, noise points are scattered uniformly in the space. Due to space limitations a detailed description of the synthetic data generation process is omitted, and the reader is referred to Ref. [8] which we used as a guideline.

A candidate data set which we aimed at generating was one in which the clusters were relatively close in some subspace with minimal overlap, and yet sparse enough in the full space that canonical clustering algorithms would not be able to detect. Fig. 1 shows an example of such a data set. The requirement for minimal overlap is due to the fact that LMCLUS is designed to identify discrete partitions of the data. We used the *cluster sparsity coefficient* [8] as a measure of the relative sparsity of the generated data sets, and indirectly as a measure of the relative "hardness" or difficulty a potential data set might present to a clustering algorithm. This measure ranges in $[0, 1]$, and is essentially a weighted average of the within cluster variances divided by the total data variance. More formally, let μ_i be the mean of cluster i, μ be the mean of the whole data set, C_i be the set of points in cluster i, D the whole data set, K the number of clusters in the data set, and x some point in D, then the cluster sparsity coefficient is defined by

$$\frac{1}{K} \sum_{i=1}^{K} \frac{1/|C_i|\sum_{\mathbf{x}\in C_i}\|\mathbf{x}-\mu_i\|^2}{1/|D|\sum_{\mathbf{x}\in D}\|\mathbf{x}-\mu\|^2}.$$

The authors of Ref. [8] note that high values of this measure indicate that the data set is one on which full-dimensional or axis parallel subspace clustering would be meaningless. High values, however, do not only indicate that the data set or clusters are very sparse but may also suggest a significant overlap among the clusters. This is because clusters that significantly overlap are likely to have centers which are close to each other, and therefore close to the center of the whole data set.

Two families of data sets were generated as part of the evaluation. The first having cluster sparsity coefficient in the range $[0.5, 0.6]$, yielding data sets that we found to be relatively hard for full-dimensional clustering algorithms, yet with very small and possibly no cluster overlap. The data set depicted in Fig. 1 for example has a cluster sparsity coefficient value of 0.54. The second type, which we call *star* data sets due to their star like geometry, yielded cluster sparsity coefficient values close to 0.99. The star data sets were generated so that cluster centers were relatively close to each other, yet with minimal overlap.

5.2. LMCLUS parameter setting

LMCLUS requires the setting of three input parameters: K, an upper limit on the dimension of the linear manifolds in

Table 1
A sample of 15 data sets used to evaluate the "accuracy" of selected clustering algorithms

	Data size	Num. of clusters	Space dim.	Manifold dim.
D_1	3000	3	4	2–3
D_2	3000	3	20	13–17
D_3	30,000	4	30	1–4
D_4	6000	3	30	4–12
D_5	4000	3	100	2–3
D_6	90,000	3	10	1–2
D_7	5000	4	10	2–6
D_8	10,000	5	50	1–4
D_9	80,000	8	30	2–7
D_{10}	5000	5	3	1–2
$*D_{11}$	1500	3	3	1
$*D_{12}$	1500	3	3	2
$*D_{13}$	1500	3	7	3
$*D_{14}$	5000	5	20	4
$*D_{15}$	4000	4	50	3

The table lists the characteristics of each of the data sets.

which we believe clusters may be embedded; S, a sampling level parameter which based on the heuristic outlined in Section 3.2 we suggest setting to a rough estimate of number of clusters that might exist in the data set; Γ, a sensitivity or "goodness of separation" threshold, which the higher it is set the more coarse the clustering will be, and the lower it is set the finer (more and smaller clusters) the clustering will be. The first two input parameters are fairly intuitive and easy to set. For Γ we have yet to find a general approach, and similar to other methods, we suggest setting by experimenting with different values. Nonetheless, we found that not many trials are required until a reasonable clustering is obtained. Moreover, we found that when clusters are well separated setting $\Gamma = 1.0$ will yield good results and anything higher than 1.0 will only return the same results. I.e. $\Gamma = 1.0$ can be thought of as a threshold for the case of well separated clusters. When clusters are closer to each other or tend to overlap the value of Γ must be decreased. Based on many observations and experiments with data sets having clusters that are relatively close, we suggest setting Γ in the range $[0.4, 0.5]$. Any clustering obtained by setting $\Gamma < 0.4$ indicates significant cluster overlap.

5.3. Accuracy

From a pool of dozens of synthetic data sets on which LMCLUS was applied, a representative sample of 15 was selected for illustrative purposes. Their characteristics are summarized in Table 1, where the *star* data sets are marked with a star ('*'). As a measure of accuracy we used *cluster purity* [20] operating on a *confusion matrix*. Confusion matrices are used to indicate how well the output clusters match with the input clusters, also referred to as "ground truth" in a supervised environment. The (i, j) entry of the matrix specifies the number of points belonging to output cluster labeled i, which were generated as part of input cluster labeled j. Ideally, when the clustering algorithm performs well, there should be a single dominant entry in each

Table 2
Accuracy and running times (in hours, minutes and seconds) comparison

	LMCLUS		ORCLUS		DBSCAN		HPPC	
D_1	0.95	0:0:08	0.80	0:0:22	0.34	0:0:9	0.72	0:0:51
D_2	0.98	0:0:33	0.59	0:2:18	0.65	0:0:36	0.97	0:1:39
D_3	1.00	0:15:38	0.65	1:5:30	1.00	1:31:52	0.99	0:1:32
D_4	0.99	0:9:22	0.98	0:8:20	0.66	0:3:49	0.97	0:0:12
D_5	1.00	0:0:20	0.88	0:54:30	0.65	0:5:24	0.99	0:3:54
D_6	0.99	0:0:29	1.00	0:29:02	0.67	4:58:49	1.00	0:1:23
D_7	0.99	0:2:05	0.99	0:2:41	0.74	0:0:54	0.96	0:0:35
D_8	0.99	0:1:42	0.63	1:33:52	1.00	0:17:00	0.99	0:3:43
D_9	0.99	3:12:46	0.96	13:30:30	1.00	10:51:15	0.99	0:4:57
D_{10}	0.86	0:0:48	0.68	0:0:45	0.59	0:0:5	0.78	0:0:33
$*D_{11}$	0.98	0:0:01	0.99	0:0:10	0.43	0:0:02	0.33	0:0:52
$*D_{12}$	0.97	0:0:02	0.99	0:0:11	0.34	0:0:02	0.33	0:0:26
$*D_{13}$	0.97	0:0:05	0.99	0:0:17	0.33	0:0:04	0.33	0:0:34
$*D_{14}$	0.99	0:5:46	1.00	0:10:42	0.21	0:1:39	0.20	0:1:30
$*D_{15}$	0.99	0:9:14	1.00	0:25:52	0.25	0:2:34	0.25	0:3:20

Each algorithm was applied on a sample of 15 data sets whose characteristics are listed in Table 1.

row that establishes a match of an output cluster to an input cluster. Cluster purity is then used to quantify this match and the overall quality of the clustering which we refer to as accuracy. Let K be the number of output clusters, $|D|$ the size of the data set, $|C_i|$ the size of output cluster i, and $|C_{ij}|$ the number of points output cluster i and input cluster j have in common. Then the purity of cluster C_i is defined as

$$Purity(C_i) = \frac{1}{|C_i|} \max_j (|C_{ij}|)$$

and overall purity or what we refer to as the overall accuracy of the clustering as

$$Purity = \sum_{i=1}^{K} \frac{|C_i|}{|D|} purity(C_i) = \frac{1}{|D|} \sum_{i=1}^{K} \max_j (|C_{ij}|),$$

which is essentially a weighted sum of individual cluster purities.

Table 2 summarizes the overall accuracy, along with the amount of time required by each algorithm we tested to cluster the sample of 15 data sets. Apart from DBSCAN, each of the other algorithms, due to their stochastic nature, were run several times and their averages were recorded. The results depicted in Table 2 clearly demonstrate LMCLUS's superiority over the other clustering algorithms. LMCLUS was the only algorithm able to cluster with over 0.85 accuracy of all the 15 data sets. ORCLUS ranked second, was able to cluster accurately 10 of the 15 data sets. HPPC ranked third, clustered accurately 8 of the 15 data sets. DBSCAN ranked last, was able to cluster accurately only three of the data sets. The inability of DBSCAN to cluster these data sets is no surprise for clustering algorithms that utilize distance metrics in the full space to cluster subspace or linear manifold clusters. Only LMCLUS and ORCLUS were able to handle the *star* clusters. In some cases ORCLUS slightly outperformed LMCLUS by a small margin. The fact that HPPC was not able to cluster the *star* data sets also comes as no surprise. This is because as a projection pursuit

algorithm HPPC searches for "interesting"-separations revealing 1D projections, and any 1D projection of these type of data sets, due to the star like arrangement of clusters in the space, will only reveal "uninteresting" unimodal distributions of projections that cannot be used to separate the data. In terms of running time, LMCLUS ranked second to HPPC. The remarkable low running times of HPPC can be attributed to the fact that it uses a stochastic "genetic" approach to select interesting 1D projections using a constant number of "generations", making it scale linearly with the size and dimensionality of the data. Nonetheless LMCLUS runs faster than the other algorithms on seven of the 15 data sets, and when compared to ORCLUS and DBSCAN only, demonstrates a significant gain in efficiency, especially when applied on large or high dimensional data sets.

5.4. Scalability

Scalability was assessed in terms of the size and dimensionality of the data. In the first set of tests, we fixed the number of dimensions at 10, and the number of clusters to 3, each of same size, and embedded in a 3D manifold. We then increased the number of points from 1000 to 1,000,000. In the second set of tests we fixed the number of points and clusters as before, but increased the number of dimensions in the data set from 10 to 120. As before, LMCLUS and ORCLUS were run several times on each data set, reporting the averages. Figs. 6 and 7 are plots of the running time for the two sets of tests. Although LMCLUS's asymptotic time complexity indicates that it is quadratic in the size of the data, these set of tests confirm our suggestion from Section 4 that in practice LMCLUS will scale much better. Fig. 6(a) shows that in practice, for data sets with a small number of clusters which are embedded in low dimensionality manifolds, LMCLUS like ORCLUS scales linearly with the size of the data. We note, however, that as the dimensionality of the cluster manifolds increases, performance will degrade. Fig. 7(a) validates our analytical complexity analysis that LMCLUS, like DBSCAN, will scale linearly
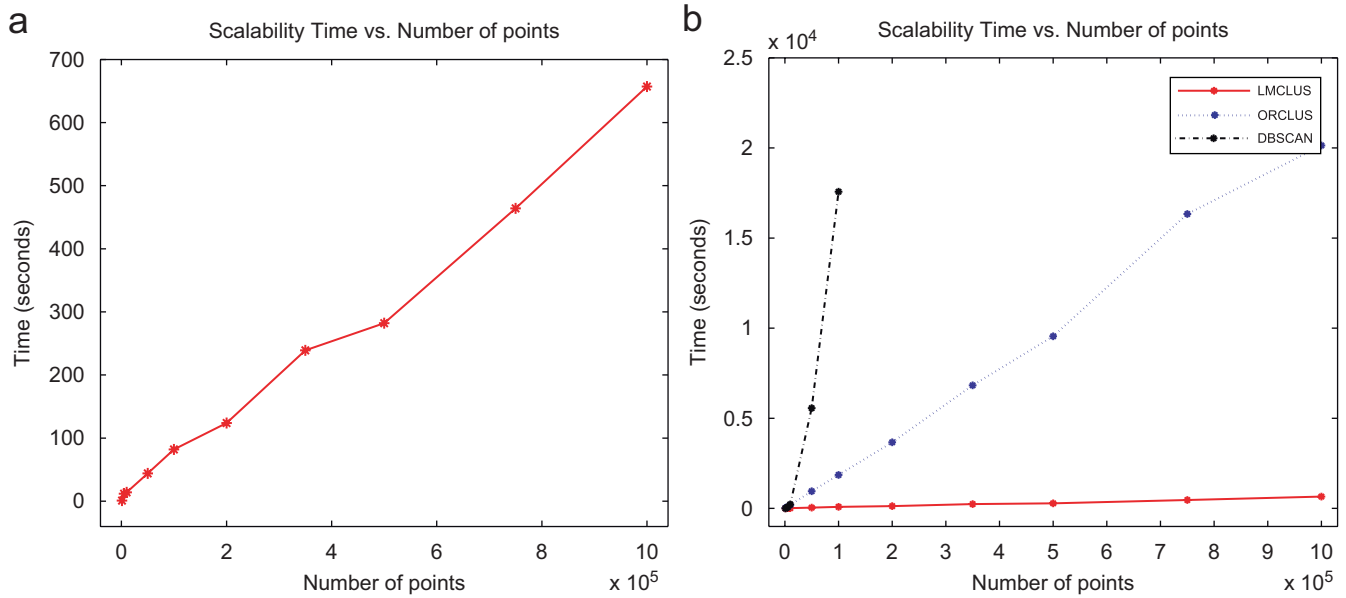
Fig. 6. Scalability, running time vs. data size: (a) LMCLUS's scalability; (b) LMCLUS's scalability relative to ORCLUS and DBSCAN.
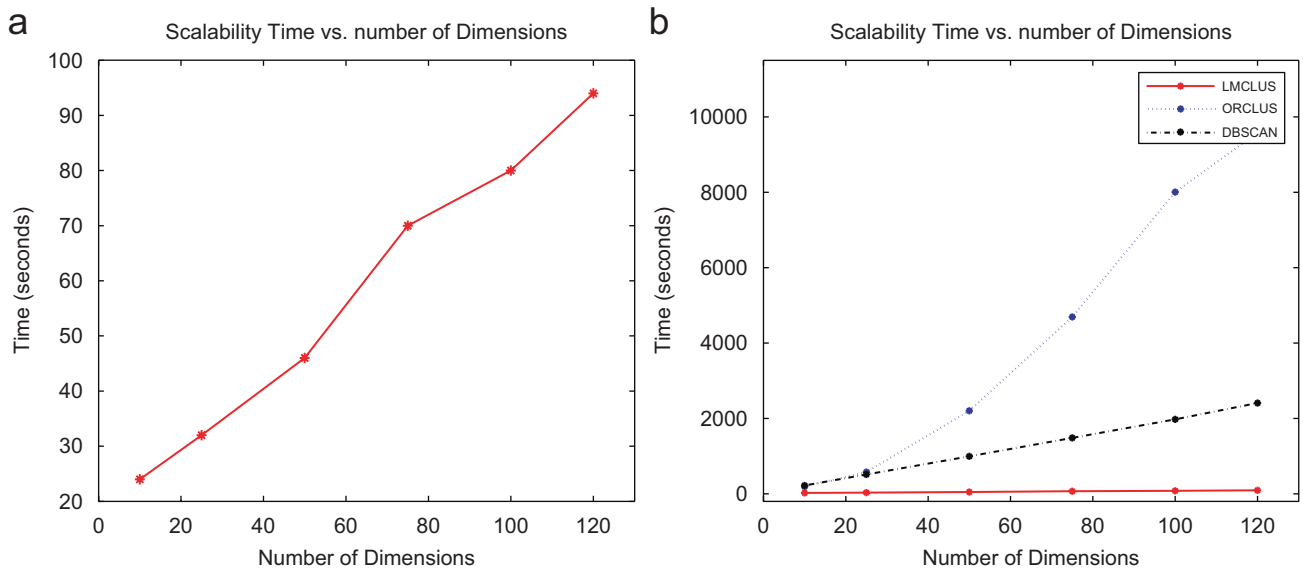

Fig. 7. Scalability, running time vs. dimensionality of the data: (a) LMCLUS's scalability; (b) LMCLUS's scalability relative to ORCLUS and DBSCAN.

with the dimensionality of the data set. Combined together, linearity in both the size and dimensionality of the data set makes LMCLUS one of the fastest algorithms in its class, and far superior to ORCLUS and DBSCAN in terms of running time as depicted by Figs. 6(b) and 7(b).

5.5. Stability

A stochastic algorithm such as LMCLUS will produce results that differ with each run depending on the random number generator's seed. For the stability set of tests we used two data sets on which LMCLUS was applied 500 times, and for each

run we recorded the accuracy. The first data set, was the sample data depicted in Fig. 1. We chose this data set because of the relative proximity of its clusters, which in turn requires the sampled manifolds to be fairly accurate approximations of the true manifolds in which the clusters are embedded. The second data set was a *star* type data set consisting of 3000 points in a 10D space, with four clusters embedded in 3D manifolds. The same set of tests were also applied on ORCLUS. The results of this experiment are shown in Table 3. For each data set we measured the mean, median and standard deviation (stdev.) of the algorithm's accuracy. The table shows that LMCLUS is able to maintain high accuracy. Applied on the first data set

Table 3
Stability of an algorithm in terms of accuracy

	LMCLUS	ORCLUS
1st data set		
Mean	0.991	0.921
Median	0.999	0.955
Stdev.	0.047	0.1056
2nd data set		
Mean	0.974	0.993
Median	0.974	0.995
Stdev.	0.000053	0.000049

LMCLUS was able to maintain an average of 0.991 accuracy with only 4 runs which yielded below 0.50 accuracy. Whereas ORCLUS had an average of 0.921 with more than 20 runs below 0.50 accuracy as indicated by the larger standard deviation. Applied on the second data set both algorithms maintained high accuracy, but ORCLUS performed slightly better, a result consistent with the accuracy experiments in Section 5.3 where ORCLUS slightly outperformed LMCLUS on the *star* type data sets.

5.6. Real data

5.6.1. Time series clustering and classification

Multivariate time series classification has become increasingly important due to new applications in biomedical signal analysis, DNA microarray analysis, and datamining in temporal databases. We applied LMCLUS on a data set obtained from the UCI KDD Archive [21] consisting of 600 points with 60 attributes each, divided into six different classes: decreasing trend, cyclic, normal, upward shift, increasing trend, and downward shift. The donors of this data set claim that this is a good data set to test time series clustering because Euclidean distance measures will not be able to achieve good accuracy. Setting $\Gamma = 0.4$ (the sensitivity input parameter) LMCLUS was able to achieve 0.87 accuracy by discovering eight clusters all embedded in 1D linear manifolds, where the cyclic class was divided into three clusters, and where most of the misclassified points came from the decreasing trend class. ORCLUS was only able to achieve 0.50 accuracy, while DBSCAN with extensive tuning of its parameters yielded 0.68 accuracy, and HPPC 0.64 accuracy.

5.6.2. Handwritten digit recognition

A second real data experiment consisted of preprocessed handwritten digit bitmaps obtained from the UCI Machine Learning Repository [22]. A 32×32 bitmap representing each digit was divided into non-overlapping 4×4 blocks, and the number of "on" pixels in each block was counted. The reduced 8×8 matrix was then converted into a 64-dimensional feature vector whose entries contained integers in the range [0, 16]. We then partitioned this data set of 3823 feature vectors into two smaller sets of even and odd digits. Applied on the set of even digits LMCLUS was able to achieve an average of 0.95 accuracy, discovering five clusters all embedded in one

and 2D manifolds. DBSCAN was able to achieve 0.82 accuracy, while ORCLUS achieved 0.85, and HPPC achieved 0.50 accuracy. Applied on the set of odd digits LMCLUS achieved an average of 0.82 accuracy, discovering seven clusters (two clusters for each of the digits 1 and 9) all embedded in one and 2D manifolds. DBSCAN achieved 0.58 accuracy, whereas ORCLUS achieved 0.83, and HPPC 0.93 accuracy. We set $\Gamma = 0.4$ for both data sets as input to LMCLUS. We note that the donors of this data set reported they were able to achieve an average of 0.97 classification accuracy by using a supervised K-nearest neighbors classification scheme, with a Euclidean distance metric.

6. Conclusion

A new clustering paradigm which is based on the concept of linear manifolds was presented in this paper. Unlike conventional cluster models a linear manifold cluster is a set of points compact around a linear manifold. The linear manifold cluster model presented in this paper can be viewed as a generalization of other more recent models such as the subspace and pattern cluster models. In addition linear manifolds are able to capture correlations in the data, making the concept of linear manifold clustering applicable to a wide range of applications. An efficient stochastic algorithm called LMCLUS for detecting groups of points embedded in low dimensionality linear manifolds was presented. The algorithm was evaluated on real and synthetic data sets and shown to outperform three competing methods in terms of accuracy and computation time. The algorithm scales well with the size and dimensionality of the data, but not with the dimensionality of the linear manifolds in which clusters are embedded. We believe that clusters embedded in higher dimensionality linear manifolds need to be treated differently and plan to investigate possible approaches in future work. Likewise we plan to extend the concept of linear manifold clustering to non-linear manifolds.

References

[1] J.H. Friedman, An overview of predictive learning and function approximation, in: V. Cherkassky, J.H. Friedman, H. Wechsler, (Eds.), From Statistics to Neural Networks, Springer, Berlin, NATO/ASI, 1994, pp. 1–61.

[2] J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, University of California Press, Berkeley, 1967, pp. 281–297.

[3] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, OR, 1996, pp. 226–231.

[4] L. Parsons, E. Haque, H. Liu, Subspace clustering for high dimensional data: a review, SIGKDD Explorations, Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining, vol. 6(1), 2004, pp. 90–105.

[5] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan, Automatic subspace clustering of high dimensional data for data mining applications, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 1998, pp. 94–105.

[6] S. Goil, H. Nagesh, A. Choudhary, Mafia: efficient and scalable subspace clustering for very large data sets, Technical Report 9906-010, Northwestern University, June 1999.

[7] C.C. Aggarwal, J.L. Wolf, P.S. Yu, C. Procopiuc, J.S. Park, Fast algorithms for projected clustering, in: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, ACM Press, 1999, pp. 61–72.

[8] C.C. Aggarwal, P.S. Yu, Finding generalized projected clusters in high dimensional spaces, in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, 2000, pp. 70–81.

[9] J.M. Huband, J.C. Bezdek, R.J. Hathaway, BigVAT: visual assessment of cluster tendency for large data sets, Pattern Recognition 38 (11) (2005) 1875–1886.

[10] Y. Cheng, G.M. Church, Biclustering of expression data, in: Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology, 2000, pp. 93–103.

[11] J. Yang, W. Wang, H. Wang, P.S. Yu, δ-clusters: capturing subspace correlation in a large data set, in: ICDE '02: Proceedings of the 18th International Conference on Data Engineering, 2002, pp. 517–528.

[12] R. Harpaz, R.M. Haralick, Exploiting the geometry of gene expression patterns for unsupervised learning, in: 18th International Conference on Pattern Recognition (ICPR 2006), vol. 2, 2006, pp. 670–674.

[13] S. Dasgupta, Learning mixtures of gaussians, in: Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, 1999.

[14] A.D. Miasnikov, J.E. Rome, R.M. Haralick, A hierarchical projection pursuit clustering algorithm, in: 17th International Conference on Pattern Recognition (ICPR 2004), 2004, pp. 268–271.

[15] J. Kittler, J. Illingworth, Minimum error thresholding, Pattern Recognition 19 (1) (1986) 41–47.

[16] P. Diaconis, D. Freedman, Asymptotics of graphical projection pursuit, Ann. Stat. 12 (3) (1984) 793–815.

[17] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, second ed., Wiley, New York, 2000.

[18] S. Cho, R.M. Haralick, S. Yi, Improvement of Kittler and Illingworth's minimum error thresholding, Pattern Recognition 22 (5) (1989) 609–617.

[19] S. Dasgupta, A. Gupta, An elementary proof of the Johnson–Lindenstrauss lemma, Technical Report TR-99-006, Berkeley, CA, 1999.

[20] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Addison-Wesley, Reading, MA, 2005.

[21] S. Hettich, S.D. Bay, The UCI KDD archive, 1999, ⟨http://kdd.ics.uci.edu⟩.

[22] C.L. Blake, C.J. Merz, UCI Repository of machine learning databases, 1998, ⟨http://www.ics.uci.edu/mlearn/MLRepository.html⟩.