Dynamic Cluster Formation Using Level Set Methods

Andy M. Yip, Chris Ding, and Tony F. Chan

***Abstract*— Density-based clustering has the advantages for (i) allowing arbitrary shape of cluster and (ii) not requiring the number of clusters as input. However, when clusters touch each other, both the cluster centers and cluster boundaries (as the peaks and valleys of the density distribution) become fuzzy and difficult to determine. We introduce the notion of *cluster intensity function* (CIF) which captures the important characteristics of clusters. When clusters are well-separated, CIFs are similar to density functions. But when clusters become closed to each other, CIFs still clearly reveal cluster centers, cluster boundaries, and degree of membership of each data point to the cluster that it belongs. Clustering through bump hunting and valley seeking based on these functions are more robust than that based on density functions obtained by kernel density estimation, which are often oscillatory or over-smoothed. These problems of kernel density estimation are resolved using *Level Set Methods* and related techniques. Comparisons with two existing density-based methods, valley seeking and DBSCAN, are presented which illustrate the advantages of our approach.**

***Index Terms*— Dynamic clustering, level set methods, cluster intensity functions, kernel density estimation, cluster contours, partial differential equations.**

I. Introduction

RECENT computer, internet and hardware advances produce massive data which are accumulated rapidly. Applications include sky surveys, genomics, remote sensing, pharmacy, network security and web analysis. Undoubtedly, knowledge acquisition and discovery from such data become an important issue. One common technique to analyze data is *clustering* which aims at grouping entities with similar characteristics together so that main trends or unusual patterns may be discovered. Clustering is an example of unsupervised learning. There is no provision of any training examples to guide the grouping of the data. In the other words, cluster analysis can be applied without *a priori* knowledge of the class distribution. We refer the reader to [1], [2] for more detailed examples of the usage of clustering techniques in a variety of context.

A successful clustering task depends on a number of factors: collection of data, selection of variables, cleaning of data, choice of similarity measures, choice of a clustering algorithm, and interpretation of clustering results. In this paper, we focus on proposing a general-purpose clustering algorithm. We assume that we are given a set of data in an Euclidean space, i.e. each object is described by a set of numerical attributes, and pair-wise dissimilarity is measured by Euclidean distance. We also assume that preprocessing steps such as variable selection, data cleaning, and missing value imputation are treated separately. However, the proposed algorithm is robust to noise and outliers. Thus it can tolerate certain amount of imperfection in data cleaning.

There are a number of paradigms to define clusters. Our approach is density-based. The idea is that clusters are high density regions separated by low density regions. While this approach has a number of desirable properties (detailed in §II), a potential drawback common to all algorithms of this type is over-fitting of density. For example, density estimated from a set of samples drawn from a uniformly distribution is generally not uniform. A density-based clustering method must take this effect into account. Otherwise, the clustering results could be disastrous. We show experimentally that several well-known density-based methods fail to

This work has been partially supported by grants from DOE under contract DE-AC03-76SF00098, NSF under contracts DMS-9973341, ACI-0072112 and INT-0072863, ONR under contract N00014-03-1-0888, NIH under contract P20 MH65166, and the NIH Roadmap Initiative for Bioinformatics and Computational Biology U54 RR021813 funded by the NCRR, NCBC, and NIGMS.

A. Yip is with the Department of Mathematics, National University of Singapore, 2, Science Drive 2, Singapore 117543, Singapore. Email: matymha@nus.edu.sg

C. Ding is with the Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720. Email: chqding@lbl.gov

T. Chan is with the Department of Mathematics, University of California, Los Angeles, CA 90095-1555. Email: chan@math.ucla.edu

comply with such a robustness requirement. As a result, natural clusters are split into pieces due to the artifactitious differential densities.

To remedy such a problem, we propose a partial differential equation model to detect high density regions. The model has a built-in mechanism, which can be thought of adding surface tension to cluster boundaries, to overcome the localized roughness of the density landscape. To implement the dynamical evolution of cluster boundaries, we employ the Level Set Methods which allow moving boundaries to split or merge easily. We further introduce the concept of cluster intensity functions which clearly reveals cluster structures. Partitioning data according valleys of such functions provides an extra degree of robustness.

The organization of the rest of the paper is as follows. In the next subsection, we give a brief review of clustering algorithms. Then, we highlight some characteristics of density-based methods and some related concepts in §II. In §III, we outline the main steps of our methodologies. The initialization steps of our method is presented in §IV. Followed next in §V is the major step, which is to advance cluster boundaries robustly. A novel concept, clustering intensity function, for finalizing the clusters is introduced in §VI. Experimental results are then presented in §VII. Finally, some conclusion remarks are given in §VIII.

A. A Review of Clustering Algorithms

To facilitate a better understanding of our density-based method, we include a brief summary of various classes of clustering algorithms so as to contrast the different assumptions underlying each class of algorithms. Pointers to the literature are also given. A more thorough discussion of density-based methods and relevant concepts is presented in the next section. For more detailed reviews of clustering techniques, we refer the reader to [1], [2], [3], [4].

(i) Optimization-based methods. They seek for a partition of the dataset so as to optimize an objective. Usually, a measure of within-cluster similarity is maximized and/or a measure of between-cluster dissimilarity is maximized. Constraints such as number of clusters or minimum separation between clusters may be incorporated [5]. Perhaps the most well-known algorithm of this type is the Lloyd's Algorithm [6] for optimizing the k-means objective [7]. A generalization of the k-means objective and Lloyd's Algorithm to Bregman Divergence can be found in [8]. Algorithms for optimizing the k-medoids objective, a variant of k-means which is more robust to outliers, include PAM [3], CLARA [3] and CLARANS [9].

(ii) Hierarchical methods. They aim at producing a hierarchical tree (dendrogram) which depicts the successive merging (agglomerative methods) or splitting (divisive methods) of clusters. Examples of hierarchical methods include DIANA, Single Linkage, Average Linkage, Complete Linkage, Centroid and Ward's methods [3]. Different agglomerative methods differ by the way that the similarity between two clusters is updated. If the linkage satisfies a cluster aggregate inequality, then the algorithm can be implemented efficiently at a time complexity of $O(N^2)$ only [10]. A more recent method CHAMELEON [11] uses a sophisticated merging criterion which takes the clusters' internal structure into account as well.

(iii) Density-based methods. They are based on the idea that clusters are high density regions separated by low density regions. Methods of this type include Valley Seeking [12], DBSCAN [13], GDBSCAN [14], CLIQUE [15], DENCLUE [16] and OPTICS [17]. Our approach is density-based. We will further elaborate the pros and cons of this paradigm in the next section.

(iv) Grid-based methods. The feature space is projected onto a regular grid. Presumably, most non-empty grid cells are highly populated. Thus, by using a few representatives or summary statistics for each grid cell, a form of data compression is obtained. Such an approach is usually used for large databases. STING [18] and WaveCluster [19] fall into this category.

(v) Graph-based methods. Data points are represented by nodes and pair-wise similarity are depicted by the edge weights. Once a graph is constructed, graph partitioning methods can be used to obtain clusters of data points. Examples of graph-based methods are Spectral Min-Max Cut [20] and Shared Nearest Neighbors [21].

(vi) Model-based methods. They are application-specific. They assume the knowledge of a model which prescribes the nature of the data. CLICK [22] uses a finite mixture model [23] to model pair-wise similarity between points in the same cluster and points in different clusters.

II. Density-based Approaches and Level Set Methods

A. Density-based Approaches

Among various classes of clustering algorithms, density-based methods are of special interest for their connections to statistical models which are very useful in many applications. Density-based clustering has the advantages for (i) allowing arbitrary shape of cluster and (ii) not requiring the number of clusters as input, which is usually difficult to determine.

There are several basic approaches for density-based clustering:

(A1) A common approach is so-called bump-hunting: first find the density peaks or "hot spots" and then expand the cluster boundaries outward until they meet somewhere, presumably in the valley regions (local minimums) of density contours. The CLIQUE algorithm [15] adopted this methodology.

(A2) Another direction is to start from valley regions and gradually work uphill to connect data points in low-density regions to clusters defined by density peaks. This approach has been used in Valley Seeking [12] (see below) and DENCLUE [16].

(A3) A recent approach, DBSCAN [13], is to compute reachability from some seed data and then connect those "reachable" points to their corresponding seed. Here, a point p is reachable from a point q (with respect to $MinPts$ and Eps) if there is a chain of points $p_1 = q, p_2, \ldots, p_n = p$ such that, for each i, the Eps-neighborhood of p_i contains at least $MinPts$ points and contains p_{i+1}. A variant, called OPTICS, has been proposed in [17] which orders the data in such a way that clusterings at different density parameters are efficiently obtained.

When clusters are well-separated, density-based methods work well because the peak and valley regions are well-defined and easy to detect. When clusters are closed to each other, which is often the case in real situations, both the cluster centers and cluster boundaries (as the peaks and valleys of the density distribution) become fuzzy and difficult to determine. In higher dimension, the boundaries become wiggly and over-fitting often occurs.

In this paper, we adopt the framework of bump-hunting but with several new ingredients incorporated to overcome problems that many density-based algorithms share. The major steps of our method are as follows: (i) obtain a probability density function (PDF) by *Kernel Density Estimation*; (ii) identify peak regions of the density function using a surface evolution equation implemented by the *Level Set Methods* (LSM); (iii) construct a distance-based function called *Cluster Intensity Function* (CIF); (iv) apply *Valley Seeking* on the CIF. In the next subsections, we describe each of the above four notions.

B. Kernel Density Estimation (KDE)

In density-based approaches, one must need to estimate the density of data. We particularly consider the use of kernel density estimation [24], [25], a non-parametric technique to estimate the underlying probability density from samples. More precisely, given a set of data $\{\mathbf{x}_i\}_{i=1}^N \subset \mathbb{R}^p$, the probability density function (PDF) is defined to be

$$f(\mathbf{x}) := \frac{1}{(Nh^p)} \sum_{i=1}^{N} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \qquad (1)$$

where $K(\mathbf{x})$ is a positive kernel and h is a scale parameter. Clusters may then be obtained according to the partition defined by the valleys of f. An efficient valley seeking algorithm is reviewed below.

There are a number of important advantages of kernel density approach. Identifying high density regions is independent of the shape of the regions. Smoothing effects of kernels make density estimations robust to noise. Kernels are localized in space so that outliers do not affect the majority of the data. The number of clusters is automatically determined from the estimated density function, but one needs to adjust the scale parameter h to obtain a good estimate. When h is chosen too large, f will be over-smoothed and will become unimodal eventually as $h \to \infty$. On the other hand, when h is chosen too small, f may contain many spurious peaks and will eventually contain one peak for each data point as $h \to 0$. Theoretically, an optimal h is the one which minimizes the integrated squared error (ISE) $\int_{\mathbb{R}^d} (f(\mathbf{x}) - f_{\text{true}}(\mathbf{x}))^2 d\mathbf{x}$ or the expected value of ISE where f_{true} is the (unknown) underlying true density. A common practice is to apply heuristics

such as cross-validation to obtain a reasonably good estimate of the optimal h [26].

Despite the numerous advantages of kernel density methods, there are some fundamental drawbacks which deteriorate the quality of the resulting clusterings. PDFs obtained through KDE are very often oscillatory (uneven) since they are constructed by adding many kernels together. Such an oscillatory nature may lead to the problem of over-fitting, whereas a smooth cluster boundary between the clusters are usually preferred than an oscillatory one. Last but not least, valleys and peaks of PDFs are often very vague especially when clusters are closed together.

C. Level Set Methods (LSM)

We recognize that the key issue in density-based approach is how to advance the boundary either from peak regions outward towards valley regions, or the other way around. In this paper, we employ LSM, which are effective tools for computing boundaries in motion, to resolve the boundary advancing problem. LSM have well-established mathematical foundations and have been successfully applied to solve a variety of problems in image processing, computer vision, computational fluid dynamics and optimal design. LSM use implicit functions to represent complicated boundaries conveniently. While implicit representation of static surfaces have been widely used in computer graphics, LSM move one step further allowing the surfaces to dynamically evolve in an elegant and highly controllable way, see [27], [28] for details.

Advantages of LSM include: (i) the boundaries in motion can be made smooth conveniently and smoothness can be easily controlled by a parameter that characterizes surface tension; (ii) merging and splitting of boundaries can be easily done in a systematical way. Property (ii) is very important in data clustering as clusters can be merged or split in an automatic fashion. Furthermore, the advancing of boundaries is achieved naturally within the framework of partial differential equation (PDE) which governs the dynamics of the boundaries.

In LSM, a surface $\Gamma(t)$ at time t is represented by the zero level set of a Lipschitz function $\phi = \phi(\mathbf{x}, t)$, i.e., $\Gamma(t) = \{\mathbf{x} : \phi(\mathbf{x}, t) = 0\}$. The value of ϕ at non-zero level sets can be arbitrary, but a common practice is to choose ϕ to be the *signed distance function* $\psi_{\Gamma(t)}(\mathbf{x})$ for numerical accuracy reasons [27]. Our convention is that $\phi < 0$ inside $\Gamma(t)$ and $\phi > 0$ outside $\Gamma(t)$.

In general, the signed distance function with respect to a set of surfaces Γ is defined by

$$\psi_\Gamma(\mathbf{x}) = \begin{cases} -\min\limits_{\mathbf{y}\in\Gamma} \|\mathbf{x}-\mathbf{y}\|_2 & \text{if } \mathbf{x} \text{ lies inside } \Gamma \\ \min\limits_{\mathbf{y}\in\Gamma} \|\mathbf{x}-\mathbf{y}\|_2 & \text{if } \mathbf{x} \text{ lies outside } \Gamma, \end{cases} \tag{2}$$

where $\|\cdot\|_2$ denotes the Euclidean norm. To evolve $\Gamma(t)$ (where $\Gamma(0)$ is the initial data) with speed $\beta = \beta(\mathbf{x}, t)$, the equation is given by

$$\frac{\partial \phi}{\partial t} = \beta \|\nabla \phi\|_2 \tag{3}$$

which is known as the level set equation [28]. Our PDE also takes this form. The art is to design the speed function β effectively to achieve one's goal.

D. Cluster Intensity Functions (CIF)

We may use LSM strictly as an effective mechanism for advancing boundaries. For example, in the above approach (A1), once the density peaks are detected, we may advance cluster boundaries towards low-density regions using LSM. This would be a LSM-based bump hunting approach.

However, it turns out that utilizing LSM we can further develop a new and useful concept of *cluster intensity function*. A suitably modified version of LSM becomes an effective mechanism to formulate CIFs in a dynamic fashion. Therefore our approach goes beyond the three approaches (A1)–(A3) described earlier.

CIFs are effective to capture important characteristics of clusters. When clusters are well-separated, CIFs become similar to density functions. But when clusters become closed to each other, CIFs still clearly describe the cluster structure whereas density functions and hence cluster structure become blurred. In this sense, CIFs are a better representation of clusters than density functions.

CIFs resolve the problems of PDFs while advantages of PDFs are inherited. Although CIFs are also built on the top of PDFs, they are cluster-oriented so that only information contained in PDFs that is useful for clustering is kept while other irrelevant information is filtered out. We have shown that such a filtering process is very important in clustering especially when the clusters touch each

other. On the other hand, it is well-known that when the clusters are well-separated, then valley seeking on PDFs results in very good clusterings. Since the valleys of CIFs and PDFs are very similar, if not identical, when the clusters are well-separated, clustering based on CIFs is as good as that based on PDFs. However, advantages of CIFs over PDFs become very significant when the clusters are closed together.

E. Valley Seeking

In a graph-based version described in [12], the idea is to connect each point to another point nearby having a higher density. In this way, we obtain a forest where each tree is a cluster. Presumably, the root node of each tree is located in a density peak whereas most leaf nodes are in valley regions.

This method starts off with a density estimation evaluated at each data point. The density can be obtained using the PDF described in (1) or using the k-nearest neighbor method [25]. In the experiments below, we use the PDF f. For each point $\mathbf{x}_i$, denote by $N_r(\mathbf{x}_i)$ the set of neighboring points of $\mathbf{x}_i$ within a distance of r, excluding $\mathbf{x}_i$ itself. For each $\mathbf{x}_j \in N_r(\mathbf{x}_i)$, the directional derivation of f at $\mathbf{x}_i$ along $\mathbf{x}_j - \mathbf{x}_i$ is estimated by $s_i(j) = [f(\mathbf{x}_j) - f(\mathbf{x}_i)]/\|\mathbf{x}_j - \mathbf{x}_i\|_2$. Let $j_{\max} = \arg\max_j s_i(j)$. Next, if $s_i(j_{\max}) < 0$, then i is set to be a root node. If $s_i(j_{\max}) > 0$, then the point $\mathbf{x}_j$ that maximizes $s_i(j)$ is set to be the parent node of $\mathbf{x}_i$. If $s_i(j_{\max}) = 0$ and if $\mathbf{x}_i$ is connected to a root node, then the root node is assigned to be the parent of $\mathbf{x}_i$; otherwise, $\mathbf{x}_i$ becomes a root node. When all the points are visited, a forest will be constructed and each connected component (a tree) is a cluster.

In our method, once the CIF is obtained, cluster labels can be easily assigned by applying the above algorithm but with the density function replaced the distance-based CIF.

III. An Outline of Our Cluster Formation Strategies

Our clustering method consists of several major components. We here give a high-level description of the method in order to provide a global picture.

We start by introducing some terminologies. A *cluster core contour* (CCC) is a closed surface surrounding the core part/density peak of a cluster at which density is relatively high. A *cluster boundary* refers to the interface between two clusters, i.e., a surface separating two clusters. A CCC is usually located near a density peak while a cluster boundary is located at the valley regions of a density distribution. Here, a point $\mathbf{x}$ is said to belong to a valley region of f if there exists a direction along which f is a local minimum. The gradient and the Laplacian of a function g are denoted by ∇g and Δg respectively.

Our method consists of the following main steps which will be elaborated in details in the next sections:

(1) initialize CCCs to surround high density regions;
(2) advance the CCCs using LSM to find density peaks;
(3) apply valley seeking algorithm on the CIF constructed from the final CCCs to obtain clusters.

IV. Initialization of Cluster Core Contours (CCC)

We now describe how to construct an initial cluster core contours Γ_0 effectively. The basic idea is to locate the contours at which f has a relatively large (norm of) gradient. In this way, regions inside Γ_0 would contain most of the data points — we refer these regions as *cluster regions*. Similarly, regions outside Γ_0 would contain no data point at all and we refer them as *non-cluster regions*. Such an interface Γ_0 is constructed as follows.

Definition 1: An initial set of CCCs Γ_0 is the set of zero crossings of Δf, the Laplacian of f. Here, a point $\mathbf{x}$ is a zero crossing if $\Delta f(\mathbf{x}) = 0$ and within any arbitrarily small neighborhood of $\mathbf{x}$, there exist $\mathbf{x}^+$ and $\mathbf{x}^-$ such that $\Delta f(\mathbf{x}^+) > 0$ and $\Delta f(\mathbf{x}^-) < 0$.

We note that Γ_0 often contains several closed surfaces, denoted by $\{\Gamma_{0,i}\}$. The idea of using zero crossings of Δf is that it outlines the shape of datasets very well and that for many commonly used kernels (e.g. Gaussian and cubic B-spline) the sign of $\Delta f(\mathbf{x})$ indicates whether $\mathbf{x}$ is inside ($\Delta f(\mathbf{x}) < 0$) or outside ($\Delta f(\mathbf{x}) > 0$) Γ_0.

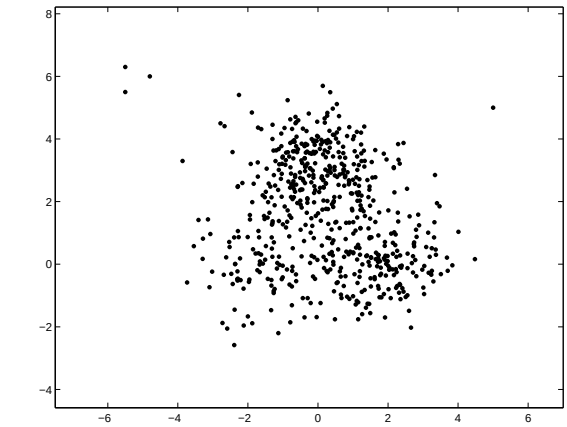
Complete reasons for using zero crossings of Δf to outline the shape of datasets are several folds: (a) the solution is a set of surfaces at which $\|\nabla f\|_2$ is relatively large; (b) the resulting Γ_0 is a set of closed surfaces; (c) Γ_0 well captures the shape of clusters; (d) the Laplacian operator is an isotropic operator which does not bias towards certain directions; (e)

the equation is simple and easy to solve; (f) it coincides with the "definition" of edges in the case of image processing. In fact, a particular application of zero crossings of Laplacian in image processing is to detect edges to outline objects [29]; (g) the sign of $\Delta f(\mathbf{x})$ indicate whether $\mathbf{x}$ is inside (negative) or outside (positive) of a cluster region.

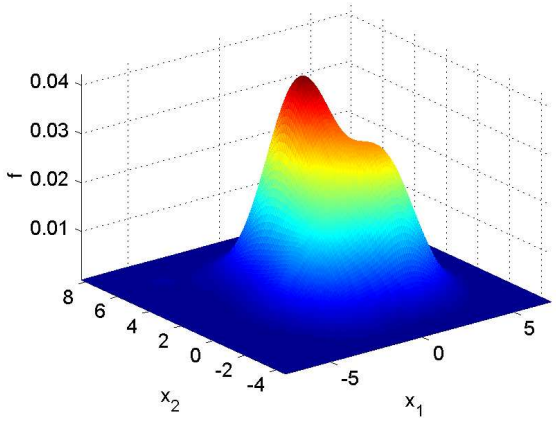
Analytic formula for Δf is often available. In case of Gaussian kernels, we have

$$\Delta f(\mathbf{x}) = \sum_{i=1}^{N} \frac{\|\mathbf{x}-\mathbf{x}_i\|_2^2 - h^2 p}{N h^{p+4} (2\pi)^{p/2}} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right).$$

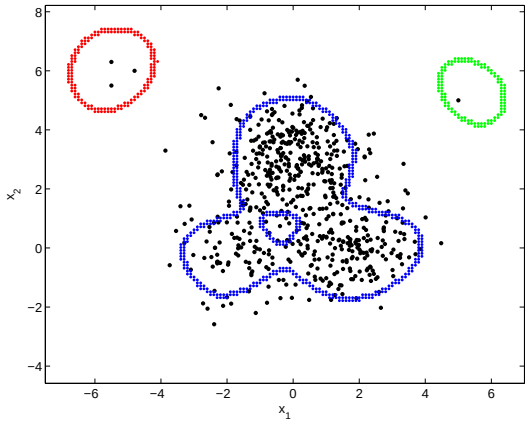
In Fig. 1(a) and (b), we show a dataset drawn from a mixture of three Gaussian components and the PDF f obtained by KDE respectively. The dataset is generated so that the three clusters are closed to each other whilst the Gaussian mixture still has three distinct peaks theoretically. Such a dataset is expected to be tough for most clustering algorithms. We observe that the valleys and peaks correspond to the two smaller large clusters of the PDF are very vague or may even not exist. Thus, the performance of PDF-based bump-hunting and/or valley seeking could be poor. In Fig. 1(c), we show the initial CCCs juxtaposed with the dataset. We observe that the CCCs capture the shape of the dataset very well.



(a)



(b)



(c)

Fig. 1. (a) A mixture of three Gaussian distributions. (b) The PDF f using Gaussian kernel with window size $h = 1$. (c) The initial CCC. In (b), peaks and valleys corresponding to the two smaller large clusters are very vague that the performance of applying bump-hunting and/or valley seeking algorithm based on the PDF is expected to be poor. Clusters obtained from our method are shown in Fig. 3. In (c), we observe that the initial CCCs capture the shape of the dataset and that the resulting boundaries capture the hot spots of the dataset very well.

V. Advancing Cluster Core Contours

Next, we discuss how to advance the initial CCCs to obtain peak regions through hill climbing in a smooth way. We found that this is a key issue in density-based approaches and is also how ideas from LSM come into play. More precisely, we employ PDE techniques to advance contours in an elegant way.

A. Evolution Equation

Since each initial CCC $\Gamma_{0,i}$ in Γ_0 changes its shape as evolution goes on, we parameterize such a family of CCCs by a time variable t, i.e., the i-th CCC at time t is denoted by $\Gamma_i(t)$. Moreover, $\Gamma(0) \equiv \Gamma_0$.

Using a level set representation, the *mean curvature* $\kappa = \kappa(\mathbf{x}, t)$ (see [28]) of $\Gamma(t)$ at $\mathbf{x}$ is given by

$$\kappa(\mathbf{x}, t) = \nabla \cdot \left(\frac{\nabla \phi(\mathbf{x}, t)}{\|\nabla \phi(\mathbf{x}, t)\|_2}\right).$$

Roughly speaking, the value of κ at $\mathbf{x}$ indicates how curve the surface $\{\mathbf{x} : \phi(\mathbf{x}, t) = 0\}$ is at $\mathbf{x}$. Moreover, if the surface is convex (respectively concave) at $\mathbf{x}$, then $\kappa > 0$ (respectively $\kappa < 0$).

Given the initial CCCs $\Gamma(0)$ represented by the zero level set of $\phi(\mathbf{x}, 0)$ (which is chosen to be the signed distance function $\psi_{\Gamma(0)}(\mathbf{x})$), the time dependent PDE that we employ for hill climbing on density functions is given by

$$\begin{aligned} \frac{\partial \phi}{\partial t} &= \left(\frac{1}{1 + \|\nabla f\|_2} + \alpha\kappa \right) \|\nabla \phi\|_2 \quad (4) \\ \phi(\mathbf{x}, 0) &= \psi_{\Gamma(0)}(\mathbf{x}). \end{aligned}$$

This equation is solved independently for each cluster region defined according to $\Gamma(t)$. During evolution, each contour and hence each cluster region may split. For example, if an initial CCC encloses two density peaks, then the CCC will eventually split into two as it climbs uphill. Evolution is stopped when no further splitting occurs.

The aim of the factor $1/(1+\|\nabla f\|_2)$ is to perform hill climbing to look for density peaks. Moreover, the factor also adjusts the speed of each point on the CCCs in such a way that the speed is lower if $\|\nabla f\|_2$ is larger. Thus the CCCs stay in steep regions of f where peak regions are defined better. In the limiting case where f has a sharp jump ($\|\nabla f\|_2 \to \infty$), the CCCs actually stop moving at the jump. We remark that in traditional steepest descent methods for solving minimization problems, the speed (step size) is usually higher if $\|\nabla f\|_2$ if larger, which is opposite to what we do. This is because our goal is to locate steep regions of f rather than local minimums.

The curvature κ exerts surface tension to smooth out the CCCs[30]. In contrast, without exerting surface tension, the CCCs could become wiggly which may lead to the common problem of overfitting of PDFs. Therefore, we employ the term κ to resolve such a problem. In fact, if ϕ is kept to be a signed distance function for all t, i.e., $\|\nabla \phi\|_2 \equiv 1$, then $\kappa = \Delta\phi$ so that ϕ is smoothed out by Gaussian filtering. In the variational point of view, the curvature term exactly corresponds to the steepest descent of the length (in 2-dimensional case) and surface area (in n-dimensional case where $n \geq 3$) of the CCCs. More precisely, under the level set representation, the length/surface area of the CCCs at time t can be expressed as (see [31]):

$$\int |\nabla H(\phi(\mathbf{x}, t))| d\mathbf{x}$$

where $H(x)$ is the Heaviside function defined by

$$H(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0. \end{cases}$$

Then the derivative of the length/surface area with respect to ϕ gives the curvature of ϕ.

B. Dynamic Adjustment of α

The scalar $\alpha \geq 0$ controls the amount of tension added to the surface and will be adjusted dynamically during the course of evolution. At the beginning of evolution of each $\Gamma_i(0)$, we set $\alpha = 0$ in order to prevent smoothing out of important features. After a CCC is split into pieces, tension is added and is gradually decreased to 0. In this way, spurious oscillations can be removed without destroying other useful features. Such a mechanism is similar to cooling in simulated annealing. In our implementation, the PDE (4) is solved at discrete time $t_k = k\Delta t$. The α for each component contour Γ_i is dynamically adjusted as follows:

- Set $\alpha = 0$ for each Γ_i initially.
- At each time point t_k and for each component Γ_i, if Γ_i is split into two contours Γ_{i_1} and Γ_{i_2}, then set the α for both Γ_{i_1} and Γ_{i_2} to be $\alpha_{\max}$. Otherwise, replace the α for Γ_i by $\gamma\alpha$ where $0 < \gamma < 1$ is a fixed constant.

In our experiments, we fix $\alpha_{\max} = 1$ and $\gamma = 0.99$. Empirically, we found that the evolution is quite robust to the choice of γ. For example, one may set γ to be any number between 0.8 and 1.0. The parameter α should be chosen to be small so that the dominated motion is hill climbing. The purpose of the curvature term is to regularize the way that the contours climb up the hill. If α is set to be too large, then the motion of the contours will have nothing to do with the PDF f at all.

C. Termination of Evolution

If we do not stop the evolution, then the contours will reach the peak regions, presumably one contour for each peak. Eventually, since $\|\nabla f\|$ is never infinity in practice (and hence the speed of the contours is never 0), each contour will shrink into a

point and disappear in finite time. Ideally, we want to stop the evolution when each contour encloses one density peak. In this case, there will be no further splitting of the contours. However, it is generally difficult to determine in advance whether the contours will split at a later time or not.

Our strategy is to let the contours evolve until they disappear. But we keep in record the contours right after each splitting. More precisely, if Γ_i is split into two contours Γ_{i_1} and Γ_{i_2}, then we store Γ_{i_1} and Γ_{i_2} and delete Γ_i from our record. In this way, we can retain the earliest contours which will not be split any further. Let t' be the earliest time so that no splitting will occur at a later time and let t'' be the time where all contours disappear. In most situations, we have $t' \gg t'' - t'$ so that the computation spent on the time interval $[t', t'']$ is small compared with that on $[0, t']$.

D. A Summary of the Evolution Equation

In summary, the PDE simply (i) moves the initial CCCs uphill in order to locate peak regions; (ii) adjusts the speed according to the slope of the PDF; (iii) removes small oscillations of the CCCs by adding tension so that hill climbing is more robust to the unevenness of the PDF (c.f. Examples 1 and 2 in §VII). In addition to these, the use of LSM allows the CCCs to change their topology easily.

In Fig. 3(a)–(c), we show the CCCs during the course of evolution governed by (4). We observe that the contours are attracted to density peaks. When a contour is split into several contours, the pieces are not very smooth near the splitting points. Since tension is added in such cases, the contours are straightened quickly.

Numerical implementation of our method is given in the Appendix.

E. Some Remarks

Remark 1: Our evolution equation (4) resembles some features of the *geometric active contours* for image segmentation [32]:

$$\frac{\partial \phi(\mathbf{x}, t)}{\partial t} = g(\|\nabla u(\mathbf{x})\|_2)\,[c + \kappa(\mathbf{x}, t)]\,\|\nabla \phi(\mathbf{x}, t)\|_2.$$

Here c is a nonnegative constant, $u(\mathbf{x})$ is the given image and $g(\|\nabla u(\mathbf{x})\|_2)$ is an edge-detector which is a decreasing function of $\|\nabla u(\mathbf{x})\|_2$ and is zero at an edge. If we define g by

$$g(\|\nabla u(\mathbf{x})\|_2) = \frac{1}{1 + \|\nabla u(\mathbf{x})\|_2},$$

then we have

$$\begin{aligned} &\frac{\partial \phi(\mathbf{x}, t)}{\partial t} \\ = \;& \frac{1}{1 + \|\nabla u(\mathbf{x})\|_2}\,[c + \kappa(\mathbf{x}, t)]\,\|\nabla \phi(\mathbf{x}, t)\|_2. \end{aligned}$$

This equation is similar to our PDE (4) except for the coefficient of κ. Presumably, contours will stop at the edges of the objects in the image. The curvature term is used to regularize the motion so that it is robust to the noise in the image.

Remark 2: While our approach dynamically evolves a set of contours to detect density peaks, another approach called *support vector clustering* [33] obtains a set of static surfaces where each of them encloses one cluster. The idea is to first use kernel methods to map the data to a high dimensional feature space. Then, find the sphere with the smallest radius which encloses most of the data points in the feature space (a few outliers are allowed). Finally, apply the inverse mapping to map the sphere back to the original space. The sphere in the original space becomes a set of surfaces where each of them encloses one cluster. In this method, there is no control over the smoothness of the back transformed surfaces in the original space. Thus, it can be clearly seen from their experiments that the surfaces are very wiggly (for example, see Fig. 3 in [33]).

VI. Cluster Intensity Functions

In non-parametric modelling, one may obtain clusters by employing valley seeking on PDFs. However, as mentioned above, such methods perform well only when the clusters are well-separated and of approximately the same density in which case peaks and valleys of the PDF are clearly defined. On the other hand, even though we use the density peaks identified by our PDE (4) as a starting point, if we expand the CCCs outward according to the PDF, we still have to face the problems of the PDF; we may still get stuck in local optimum due to its oscillatory nature.

In this section, we further explore cluster intensity functions which are a better representation of

clusters than that by PDFs. Due to the advantages of CIFs, we propose to perform valley seeking on CIFs to construct clusters, rather than on PDFs. Here, CIFs are constructed based on the final CCCs obtained by solving the PDE (4).

CIFs capture the essential features of clusters and inherit advantages of PDFs while information irrelevant to clustering contained in PDFs is filtered out. Moreover, peaks and valleys of CIFs stand out clearly which is not the case for PDFs. The principle behind is that clustering should not be done solely based on density. Instead, it is better done based on both density and distance. For example, it is well-known that the density-based algorithm DBSCAN [13] cannot separate clusters that are closed together even though their densities are different and the density-based algorithm OPTICS [17] cannot separate the clusters when they have similar densities.

CIFs, however, are constructed by calculating *signed distance* from CCCs (which are constructed based on density). Thus, CIFs combine both density and distance information about the dataset. This is a form of regularization to avoid over-specification of density peaks.

The definition of a CIF is as follows.

Definition 2: Given a set of closed hypersurfaces Γ (the final CCCs), the CIF φ with respect to Γ is defined to be

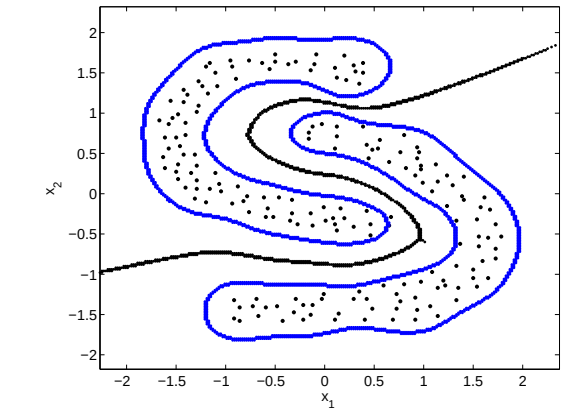
$$\varphi(\mathbf{x}) = -\psi_\Gamma(\mathbf{x})$$

where ψ_Γ is the signed distance function in (2).

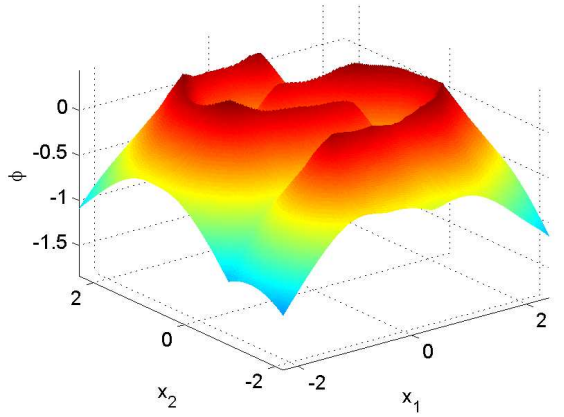
The value of a CIF at $\mathbf{x}$ is simply the distance between $\mathbf{x}$ and Γ with its sign being positive if $\mathbf{x}$ lies inside Γ and negative if $\mathbf{x}$ lies outside Γ. Roughly speaking, a large positive (respectively negative) value indicates that the point is deep inside (respectively outside) Γ while a small absolute value indicates that the point lies close to the interface Γ. To illustrate the expressive power of CIFs, an example based on the "C"-shape clusters is shown in Fig. 2.

In Fig. 3(d), the CIF constructed from the CCCs in Fig. 3(c) is shown. The peaks correspond to the three large clusters can be clearly seen which shows that our PDE is able to find cluster cores effectively.

Based on the CIF, valley seeking (c.f. §II) can be easily done in a very robust way. In Fig. 3(e), we show the valleys of the CIF juxtaposed with the dataset and the final CCCs.



(a)



(b)

Fig. 2. (a) Two "C" shape clusters juxtaposed with the zero crossings of Δf and the valleys of the CIF. (b) The CIF constructed from the zero crossings of Δf. In (a), the valleys of the CIF clearly separate the two clusters. This cannot be done effectively by distance-based methods such as k-means.

We remark that the use of signed distance as CIFs has a property that their valleys are nothing but the equidistant surfaces between the CCCs. Moreover, cluster core contours play a similar role as cluster centers in the k-means algorithm [7]. Thus, our method may be treated as a generalization of the k-means algorithm in the sense that a "cluster center" may be of arbitrary shape instead of just a point.

Under LSM framework, valleys and peaks are easily obtained. The valleys are just the singularities of the level set function (i.e. CIF) having negative values. On the other hand, the singularities of the level set function having positive values are the peaks or ridges of the CIF (also known as skeleton).

VII. Experiments

In addition to the examples shown in Figs. 1–3, we give more examples to further illustrate the

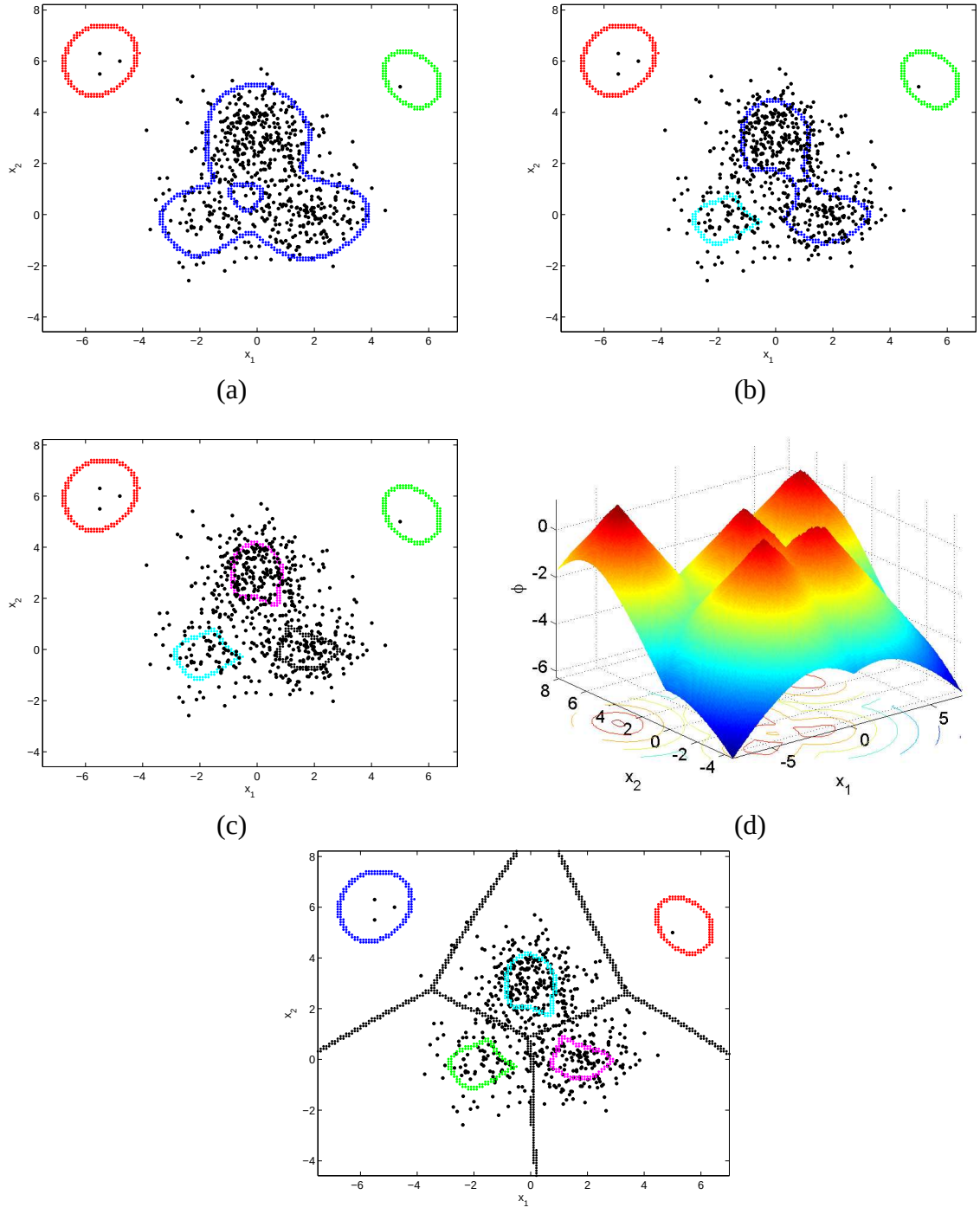


Fig. 3. Evolution of cluster core contours (CCC) using the bump hunting PDEs (4). The dataset is the one used in Fig. 1. (a) Initial CCC. (b) Intermediate CCCs. (c) Final CCCs. (d) CIF constructed from the contours in (c). Peaks corresponding to the three large clusters are clearly seen. (e) Valleys of the CIF. In (e), the three cluster cores are well-discovered.

usefulness of the concepts introduced. Comparisons with valley seeking [12] and DBSCAN [13] algorithms (c.f. §II) are given in Examples 1 and 2. Clustering results of two real datasets are also presented. For visualization of CIFs which is one dimension higher than the datasets, two dimensional datasets are used while the theories presented above apply to any number of dimensions.

When applying DBSCAN, the parameter $MinPts$ is fixed at 4 as suggested by the authors in [13].

Example 1: We illustrate how the problem of over-fitting (or under-fitting) of PDFs is resolved using our method. In Fig. 4, we compare the clustering results of valley seeking using the scale parameter $h = 0.6, 0.7$ and the DBSCAN algorithm using $Eps = 0.28, 0.29$. The best result is observed in Fig. 4(a) but it still contains several small clusters due to the spurious oscillations of the PDF. For other cases, a mixture of many small clusters and some oversized clusters are present. In contrast, our method (shown in Fig. 3) resolves these problems by (i) outlining the shape of the dataset well by keeping the CCCs smooth; (ii) using curvature motion to smooth out oscillations due to unevenness of PDFs.

Example 2: A dataset with 4000 uniformly distributed points lying in two touching circles is considered. The dataset together with the zero crossings of Δf are shown in Fig. 5(a). The result of our method is in Fig. 5(b). We observe that the final CCCs adapt to the size of the clusters suitably. The results of valley seeking on PDFs ($h = 0.05, 0.06$) are shown in Fig. 5(c) and (d) where the unevenness of the PDFs result in either 2 or 4 large clusters. The results of DBSCAN with $Eps = 0.010, 0.011$ are in Fig. 5(e) and (f) which contain many small clusters. In addition, this example also makes it clear that density functions must be regularized which is done implicitly by adding surface tension in our method.

Example 3: This example uses a dataset constructed from the co-expression patterns of the genes in yeast during cell cycle [34]. Clusters of the data are expected to reflect functional modules. The results are shown in Fig. 6. We observe that the valleys of the CIF in Fig. 6(c) are right on the low density regions and thus a reasonable clustering is obtained. We also notice that the clusters are very close, if not overlapped, to each other, especially the two clusters at the bottom.

Example 4: Our next example uses a real dataset from text documents in three newsgroups. For the ease of visualization, the dataset is first projected to a 2-D space using principle component analysis [35]. The results in Fig. 7 show that the clustering results agree with the true clustering very well.

VIII. Concluding Remarks

In the paper, we introduced level set methods to identify density peaks and valleys in density landscape for data clustering. The method relies on advancing contours to form cluster cores. One key point is that during contour advancement, smoothness is enforced via LSM. Another point is that important features of clusters are captured by cluster intensity functions which serve as a form of regularization. The usual problem of roughness of density functions is overcome. The method is shown to be more robust and reliable than traditional methods that perform bump hunting or valley seeking on density functions.

Our method can also identify outliers effectively. After the initial cluster core contours are constructed, outliers are clearly revealed and can be easily identified. In this method, different contours evolve independently. Thus outliers do not affect normal cluster formation via contour advancing. Such a nice property does not hold for clustering algorithms such as the k-means where several outliers could skew the clustering.

Our method for contour advancement given by (4) is based on the dynamics of interface propagation in LSM. A more elegant approach is to recast the cluster core formation as a minimization problem where the boundary advancement can be derived from first principles which will be presented in a later paper.

Appendix
Numerical Implementations

To solve the PDE (4) numerically, we apply standard finite difference schemes for level set equations. We refer the readers to [27], [28], [36] for excellent overviews of level set methods, evolutionary equations and numerical implementations.

Consider a grid $(i_1\Delta x, i_2\Delta x, \ldots, i_p\Delta x)$ where $i_1, i_2, \ldots, i_p$ are integers, Δx is the spatial step size. We recall that p is the dimensionality of the data. The time variable is discretized to $k\Delta t$ where k is an integer and Δt is the temporal step size. Let

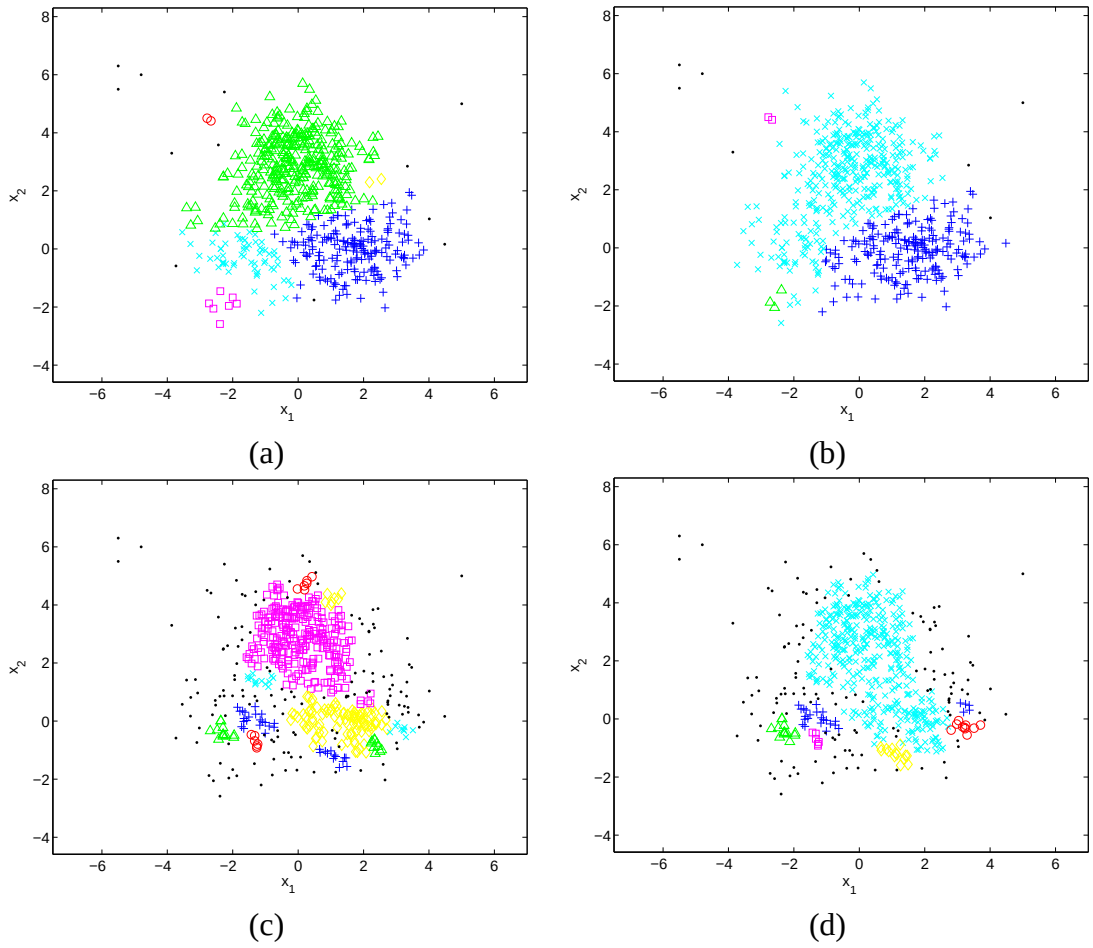


Fig. 4. Clusters obtained from applying valley seeking and DBSCAN to the dataset in Fig. 1. (a) Valley seeking with $h = 0.6$. (b) Valley seeking with $h = 0.7$. (c) DBSCAN with $Eps = 0.28$. (d) DBSCAN with $Eps = 0.29$. In (a) and (c), many small clusters are present due to unevenness of the density functions. These results are not as good as the results using our method as shown in Fig. 3

$\mathbf{i} = (i_1, i_2, \ldots, i_p)$. The sample of $\phi(\mathbf{x}, t)$ at the grid point $(i_1\Delta x, i_2\Delta x, \ldots, i_p\Delta x)$ at time $k\Delta t$ is denoted by $\phi^k_{\mathbf{i}}$.

To obtain the initial data $\phi(\mathbf{x}, 0)$ in (4), we need to compute the signed distance from the initial cluster core contours. This requires solving the Eikonal equation

$$\|\nabla\psi\|_2 = 1$$

with boundary conditions $\psi(\mathbf{x}) = 0$ on $\Gamma(0)$ which can be done fast by the Fast Marching Method [27] or Fast Sweeping Method [37]. Since the implementation is straightforward and has been detailed in [27], we omit the details.

Let $\mathbf{e}_j$ be the vector $(0, \ldots, 0, 1, 0, \ldots, 0)$ whose j-th entry is an 1 and the other entries are 0. Define the forward difference operator in the j-th spatial dimension D^+_j by:

$$D^+_j\phi^k_{\mathbf{i}} := \frac{\phi^k_{\mathbf{i}+\mathbf{e}_j} - \phi^k_{\mathbf{i}}}{\Delta x}.$$

Similarity, define the backward difference operator in the j-th spatial dimension D^-_j by:

$$D^-_j\phi^k_{\mathbf{i}} := \frac{\phi^k_{\mathbf{i}} - \phi^k_{\mathbf{i}-\mathbf{e}_j}}{\Delta x}.$$

A first order accurate upwind scheme for the level set equation

$$\frac{\partial\phi(\mathbf{x}, t)}{\partial t} = \beta(\mathbf{x}, t)\|\nabla\phi(\mathbf{x}, t)\|_2$$

is given by

$$\frac{\phi^{k+1}_{\mathbf{i}} - \phi^k_{\mathbf{i}}}{\Delta t} = -[\max(-\beta^k_{\mathbf{i}}, 0)\nabla^+ + \min(-\beta^k_{\mathbf{i}}, 0)\nabla^-]$$

where

$$\begin{aligned}\nabla^+ &= [\max(D^-_1, 0) + \min(D^+_1, 0) + \ldots + \\ &\quad \max(D^-_p, 0) + \min(D^+_p, 0)]^{1/2} \\ \nabla^- &= [\max(D^+_1, 0) + \min(D^-_1, 0) + \ldots + \\ &\quad \max(D^+_p, 0) + \min(D^-_p, 0)]^{1/2},\end{aligned}$$

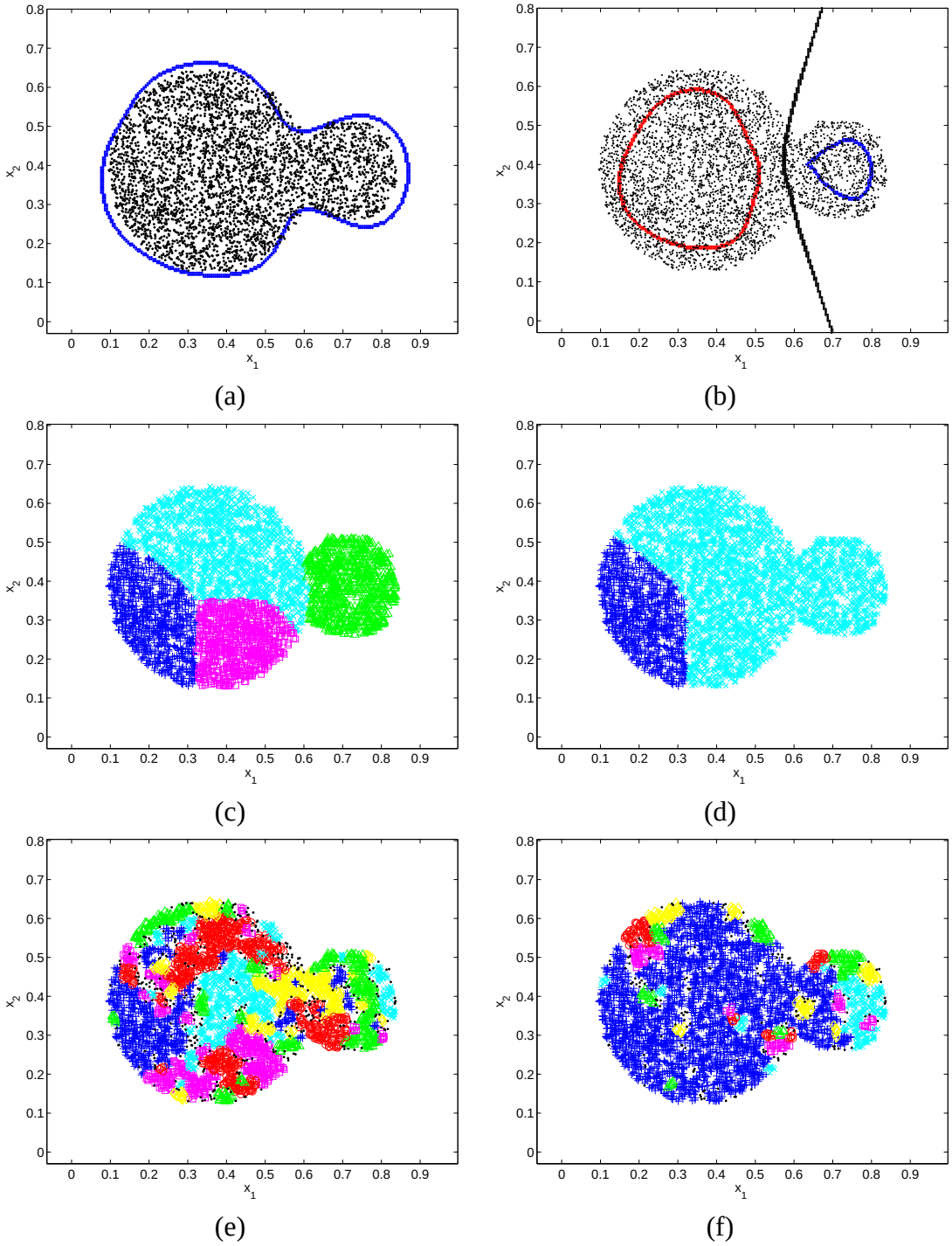


Fig. 5. Comparisons of our method with valley seeking and DBSCAN. (a) Dataset with the zero crossing of Δf superimposed. (b) Final CCCs (the closed curves in red and blue) and the valleys of the CIF (the line in black). (c) Valley seeking with $h = 0.05$. (d) Valley seeking with $h = 0.06$. (e) DBSCAN with $Eps = 0.010$. (f) DBSCAN with $Eps = 0.011$. The CCCs are able to capture the cluster shape while valley seeking and DBSCAN seem to suffer from over-fitting and result in many small spurious clusters.

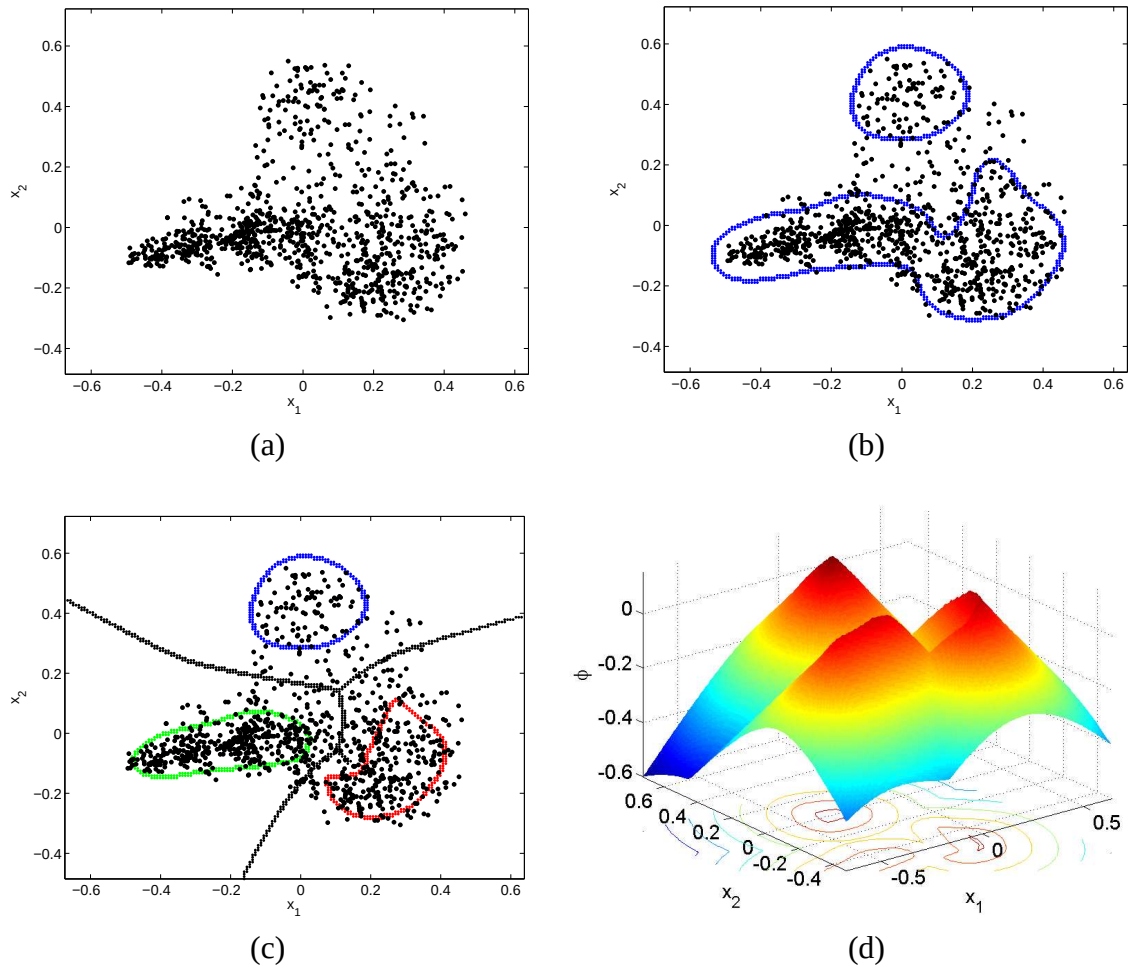


Fig. 6. (a) DNA gene expression dataset. (b) Zero crossings of Δf. (c) The final CCCs and the valleys of the CIF. (d) The CIF. The cluster cores are well-retrived and the valleys successfully separate the data into clusters of relatively high density.

see [36, p.80] and also [27, p.65]. In our PDE (4), the speed is given by

$$\begin{aligned}\beta(\mathbf{x},t) &= \frac{1}{1+\|\nabla f\|_2}+\alpha\kappa \\ &= \frac{1}{1+\|\nabla f\|_2}+\alpha\nabla\cdot\left(\frac{\nabla\phi}{\|\nabla\phi\|_2}\right)\end{aligned}$$

which is approximated by central difference to obtain $\beta_{\mathbf{i}}^k$. For higher order accurate schemes, see [27, pp.66–67].

In level set methods, we are often only interested in the evolving contours, which are located at the zero-level set of ϕ. Thus, there is no need to solve the PDE on the whole grid. We apply the Narrow Band Level Set Method [27] which updates the grid points near the moving contours. In this way, computational costs are greatly reduced.

Sparse grids techniques [38], [39] may be used to further reduce the computational complexity. In the original full grid, suppose each dimension has n grid points, then the number of grid points (in space) is n^p. Using sparse grids, which are optimally chosen subsets of grid points, we can reduce the number of grid points to $O(n(\log n)^{p-1})$ with a fairly small loss in accuracy. Indeed, sparse grids techniques have already been successfully applied to a number of data mining problems [40]–[42]. We have yet to explore this direction.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for the detailed, valuable suggestions and for bringing into our attention the support vector clustering algorithm.

REFERENCES

[1] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufman, 2001.
[2] A. K. Jain, M. N. Murty, and P. J. Flyn, "Data clustering: A review," *ACM Comp. Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

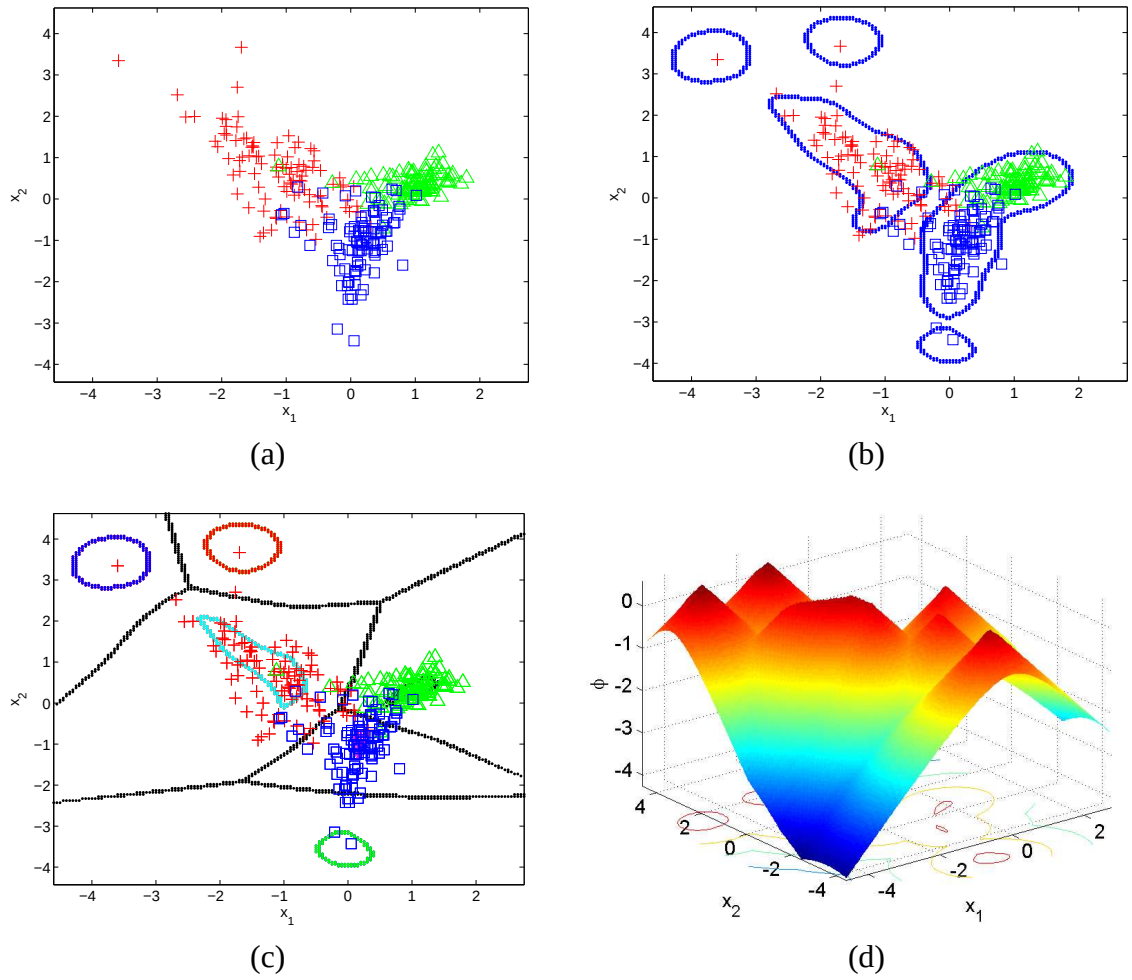


Fig. 7. (a) A dataset of 3 internet newsgroups with 100 news items in each group (news items in the same group are displayed with the same symbol). (b) The zero crossings of Δf. (c) Clustering results (lines are valleys of the final CIF and closed curves are the cluster cores). (d) The CIF.

[3] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. New York: John Wiley & Sons, Inc., 1990.

[4] P. Berkhin, "Survey of clustering data mining techniques," Accrue Software, Tech. Rep., 2002.

[5] P. Hansen and B. Jaumard, "Cluster analysis and mathematical programming," *Mathematical Programming*, vol. 79, pp. 191–215, 1997.

[6] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[7] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. I: Statistics, 1967, pp. 281–297.

[8] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh, "Clustering with bregman divergences," in *Proc. 4th SIAM Int. Conf. on Data Mining*, 2004, pp. 234–245.

[9] R. T. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining," in *Proc. 20th Int. Conf. on Very Large Data Bases*. Santiago, Chile: Morgan Kaufmann, 1994, pp. 144–155.

[10] C. Ding and X. He, "Cluster aggregate inequality and multi-level hierarchical clustering," in *Proc. 9th European Conf. Principles of Data Mining and Knowledge Discovery*, Porto, Portugal, 2005, pp. 71–83.

[11] G. Karypis, E. Han, and V. Kumar, "CHAMELEON: A hierarchical clustering algorithm using dynamic modeling," *Computer*, vol. 32, pp. 68–75, 1999.

[12] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed. Boston Academic Press, 1990.

[13] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Int. Conf. Knowledge Discovery and Data Mining*. Portland, OR: AAAI Press, 1996, pp. 226–231.

[14] J. Sander, M. Ester, H. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 169–194, 1998.

[15] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," in *Proc. ACM-SIGMOD Int. Conf. Management of Data*. Seattle, WA: ACM Press, 1998, pp. 94–105.

[16] A. Hinneburg and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *Int. Conf. Knowledge Discovery and Data Mining*. New York City, NY: AAAI Press, 1998, pp. 58–65.

[17] M. Ankerst, M. Breunig, H. P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," in *Proc. ACM-SIGMOD Int. Conf. Management of Data*. Philadelphia,

PA: ACM Press, 1999, pp. 49–60.

[18] W. Wang, J. Yang, and R. Muntz, "STING: A statistical information grid approach to spatial data mining," in *Proceedings of the 23rd VLDB Conference*, Athens, Greece, 1997, pp. 186–195.

[19] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "WaveCluster: A wavelet-based clustering approach for spatial data in very large databases," *The VLDB Journal*, vol. 8, pp. 289–304, 2000.

[20] C. Ding, X. He, H. Zha, M. Gu, and H. Simon, "Spectral min-max cut for graph partitioning and data clustering," in *Proc. 1st IEEE Int. Conf. on Data Mining*, San Jose, CA, 2001, pp. 107–114.

[21] L. Ertoz, M. Steinbach, and V. Kumar, "Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data," in *Proc. of the 3th SIAM Int. Conf. on Data Mining*, D. Barbara and C. Kamath, Eds., San Francisco, CA, 2003, pp. 47–58.

[22] R. Sharan and R. Shamir, "CLICK: A clustering algorithm with applications to gene expression analysis," in *Proc. ISMB '00*. Menlo Park, CA: AAAI Press, 2000, pp. 307–316.

[23] G. J. McLachlan and D. Peel, *Finite Mixture Models*. New York: Wiley, 2001.

[24] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Stat.*, vol. 33, pp. 1065–1076, 1962.

[25] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. New York: Wiley-Interscience, 2001.

[26] S. R. Sain, K. A. Baggerly, and D. W. Scott, "Cross-validation of multivariate densities," *Journal of the American Statisitcal Association*, vol. 89, pp. 807–817, 1992.

[27] J. A. Sethian, *Level Set Methods and Fast Marching Methods*, 2nd ed. New York: Cambridge Univ. Press, 1999.

[28] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*. New York: Spring Verlag, 2003.

[29] A. K. Jain, *Fundamentals of Digital Image Processing*. Englewood Cliffs, NJ: Prentice Hall, 1988.

[30] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on Hamiton-Jacobi formulations," *Journal of Computational Physics*, vol. 79, pp. 12–49, 1988.

[31] H. K. Zhao, T. Chan, B. Merriman, and S. Osher, "A variational level set approach to multiphase motion," *Journal of Computational Physics*, vol. 127, pp. 179–195, 1996.

[32] V. Caselles, F. Catté, T. Coll, and F. Dibos, "A geometric model for active contours in image processing," *Numerische Mathematik*, vol. 66, pp. 1–31, 1993.

[33] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik, "Support vector clustering," *Journal of Machine Learning Research*, vol. 2, pp. 125–137, 2001.

[34] P. Spellman, G. Sherlock, M. Zhang, V. Iyer, and K. Anders, "Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization," *Mol. Biol. Cell*, vol. 9, no. 12, pp. 3273–3297, 1998.

[35] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed. Springer, 2002.

[36] G. Sapiro, *Geometric Partial Differential Equations*. New York: Cambridge Univ. Press, 2001.

[37] Y. R. Tsai, L. Cheng, S. Osher, and H. Zhao, "Fast sweeping algorithms for a class of Hamilton-Jacobi equations," *SIAM J. Numer. Anal.*, vol. 41, no. 2, pp. 673–694, 2003.

[38] C. Zenger, "Sparse grids," in *Parallel Algorithms for Partial Differntial Equations*, ser. Notes on Numerical Fluid Mechanics, W. Hackbusch, Ed. Braunschweig/Wiesbaden: Vieweg, 1991, vol. 31.

[39] H. Bungartz and M. Griebel, "Sparse grids," *Acta Numerica*, pp. 147–269, 2004.

[40] J. Garcke and M. Griebel, "Classification with sparse grids using simplicial basis functions," in *Proc. 7th ACM SIGKDD*, F. Provost and R. Srikant, Eds. ACM, 2001, pp. 87–96.

[41] J. Garcke, M. Griebel, and M. Thess, "Data mining with sparse grids," *Computing*, vol. 67, no. 3, pp. 225–253, 2001.

[42] J. Garcke, M. Hegland, and O. Nielsen, "Parallelisation of sparse grids for large scale data analysis," in *Proc. of the Int. Conf. on Computational Science 2003*, P. M. A. Sloot, D. Abramson, A. Bogdanov, J. Dongarra, A. Zomaya, and Y. Gorbachev, Eds., 2003, pp. 683–692.