Estimation of Level Set Trees Using Adaptive Partitions

Lasse Holmström*
Department of Mathematical Sciences, University of Oulu
Kyösti Karttunen
CEMIS Oulu, University of Oulu
and
Jussi Klemelä

October 12, 2016

Abstract

We present methods for the estimation of level sets, a level set tree, and a volume function of a multivariate density function. The methods are such that the computation is feasible and estimation is statistically efficient in moderate dimensional cases ($d \approx 8$) and for moderate sample sizes ($n \approx 50\ 000$). We apply kernel estimation together with an adaptive partition of the sample space. We illustrate how level set trees can be applied in cluster analysis and in flow cytometry.

Keywords: Cluster analysis, flow cytometry, kernel density estimation, mode detection, recursive partitioning.

*The authors gratefully acknowledge the TEKES funding under project 24301335.

1 Introduction

We consider estimation and computation of a level set tree and a volume function of a multivariate density function $f : \mathbf{R}^d \to \mathbf{R}$ using identically distributed observations $X_1, \ldots, X_n \in \mathbf{R}^d$ with density f.

A level set tree of a function $f : \mathbf{R}^d \to \mathbf{R}$ is a tree whose nodes correspond to the connected components of the level sets of f. A level set of f is defined as

$$\Lambda(f, \lambda) = \{x \in \mathbf{R}^d : f(x) \geq \lambda\},$$

where $\lambda \in \mathbf{R}$. The child-parent relations of a level set tree are defined by the set inclusion and the root of the tree is the support of the density. We obtain a tree with a finite number of nodes when we choose a finite number of levels λ.

A level set tree can be described using a recursive definition: (1) The root of the tree is the support of the density. We assume that the support is a connected set. (2) The level set which is one step above the support is decomposed into connected components. These connected components are the child nodes of the root node. (3) Given a node of the level set tree, we find the child nodes of the node by looking at the level set which is one step above the level of the node. We restrict ourselves on the part of the level set which is a subset of the node. This subset is decomposed into connected components and these connected components are the child nodes of the node. (4) The leaf nodes of a level set tree correspond to the local maxima (modes) of the function. The definition of a level set tree goes back at least to Reeb (1946), and level set trees were defined in Klemelä (2004b) for the purpose of visualization of density functions.

Applications of level set trees include the following. (1) Level set trees can be applied to visualize high dimensional functions, in particular, high dimensional density functions. (2) Level set trees can be applied to implement density based clustering. Indeed, a level set tree gives a useful description of the mode structure of a density function, which leads both to visualization and clustering. Level set trees help to detect such features of functions as the number, size, and location of modes. Level set trees do not only visualize modes but the complete tree structure of the level sets. To visualize functions we can use such enhancements of level set trees as volume functions and barycenter plots, as defined in

Klemelä (2004b). In density based clustering the observations are partitioned into clusters which consist of those observations that are inside a connected component of a level set (inside a node of a level set tree). In order to implement density based clustering it is useful to have an overview of the complete tree structure of level sets, instead of just looking at one level set and its connected components.

In this article we concentrate on one visualization tool for multivariate density functions: a volume function. We enhance a level set tree by adding information about the volumes of the connected components of the level sets. This leads to a volume function, which is a 1D function obtained from the original d-dimensional function. A volume function is a 1D function that has the same level set tree as the original d-dimensional function, and furthermore, the connected components of the level sets of the volume function have the lengths equal to the volumes of the corresponding connected components of the original d-dimensional function. The volume function shows the size of the modes of a density function in the sense of the excess mass, in addition to showing the mode structure.

To estimate and compute a level set tree and a volume function we apply a discretization of a kernel density estimator. First, we construct an adaptive partition of the sample space by using a greedy splitting algorithm. We create an adaptive partition whose size is typically equal to the sample size n, but it can be chosen to have a smaller size, too. Each member of the partition is a d-dimensional rectangle. Second, we evaluate a kernel density estimator at the centers of the members of the partition. This leads to a piecewise constant approximation of the kernel estimator.

Recursive partition algorithms have been used to construct classification and regression trees, as in Breiman et al. (1984), and they can be applied to construct histograms; see Klemelä (2009), Chapter 17, and the references there. In this article we apply a recursive partition algorithm to construct a partition of the sample space which discretizes a kernel density estimator. Note that an adaptive partition induces an adaptive grid.

In regressogram and histogram estimation recursive partition algorithms have been used to find adaptive partitions. The bins of a regressogram or a histogram are chosen using data so that the bins have different sizes and shapes. In regressogram estimation the aim is to obtain a partition in which the sets are large in those areas, where the regression function

does not change much, and the sets are small in those areas where the regression function changes rapidly. In histogram estimation the aim is to obtain a partition in which the sets are large in the areas with few observations, and the sets are small in the areas with many observations. The areas with few observations correspond to the areas of low density, and the areas with many observations correspond to the areas of high density. The areas of high density tend to be close to the areas where the density function changes rapidly.

An optimal partition is found by minimizing some criterion. In the case of regressograms a natural criterion is the sum of squared residuals. In the case of histograms a natural criterion is the negative log-likelihood. It is difficult to find the minimizer of these objective functions (the sum of squared residuals or the negative log-likelihood) among the collection of all partitions. Recursive partition algorithms apply stepwise minimization to find an approximate minimizer in a computationally efficient way. One starts with an initial set, which is splitted into two parts. Then each of the two parts is splitted into two additional parts. One continues in this way until a stopping criterion is satisfied.

Our purpose is not to estimate regression functions with regressograms or density functions with histograms. Instead, our purpose is to find a partition to discretize a continuous function, to make this continuous function piecewise constant. In our case the continuous function is a kernel density estimate. Some other density estimators could also be used, like Gaussian mixture density estimators, or k-nearest neighbor density estimators. We have applied the minimization of the negative log-likelihood to find the partition of the discretization, similarly as in histogram estimation. This choice is made because a good discretization is such that there are many grid points in the area where the function changes rapidly, and few grid points in the area where the function changes slowly.

We can compare our approach of using an adaptive grid for the computation of a level set tree with approaches that use a regular equispaced grid, a Voronoi partition, or a Delaunay partition.

Indeed, a level set tree of a function $f : \mathbf{R}^d \to \mathbf{R}$ can be computed by evaluating the function at suitable points $x_1, \ldots, x_m \in \mathbf{R}^d$. Evaluation of the function at the points of a regular equispaced grid (dense mesh) is a straightforward approach to start the computation of a level set tree, but this leads to an exponentially growing number of grid points when

the dimension d increases. Also, we are interested in the local maxima of the density, and not in the tail regions, and it is possible to get a more accurate estimate in the regions near local maxima if the grid there is denser. In the case of density estimation the data $X_1, \ldots, X_n$ themselves are such that there are more points around the local maxima, and fewer points in the tail areas, and therefore we use the data themselves to generate the grid.

Our method can be considered as an attempt to find a partition into rectangles that best approximates the Voronoi partition. The Voronoi partition is the collection of the Voronoi cells. For a sample of points $X_1, \ldots, X_n$ the Voronoi cell of X_i is the set of those $x \in \mathbf{R}^d$ which are closer to X_i than to the other observations X_j, $j \neq i$.[1] However, the sets in the Voronoi partition are polyhedrons with a large and varying number of edges, which makes the use of the Voronoi partition difficult in the computation of a level set tree.

As an alternative, the Delaunay triangulation could be proposed to represent the connected components of the level sets, since the Delaunay triangulation is in a sense dual to the Voronoi partition. The Delaunay triangulation is the collection of simpleces with $d+1$ vertices, where the vertices are the data points.[2] However, the number of simpleces in the Delaunay partition is grows exponentially with the dimension, which makes the Delaunay partition computationally even more expensive than the use of a regular partition. In fact, the number of simpleces is $O\left(n^{\lfloor (d+1)/2 \rfloor}\right)$; see McMullen (1970).

We propose an adaptive grid obtained with a recursive partition of the sample space which combines the good features of the Voronoi and the Delaunay partitions while avoiding the undesirable features of those partitions: as in the case of the Voronoi partition, the partition we obtain is small and, as in the case of the Delaunay partition, the members of the partition are simple sets.

[1] The Voronoi cell of a point $p \in S = \{X_1, \ldots, X_n\}$ is $V_p = \{x \in \mathbf{R}^d : D(x,p) \leq D(x,q) \text{ for all } q \in S\}$, where $D(x,y) = \|x - y\|$ is the Euclidean distance in $\mathbf{R}^d$. A Voronoi cell is a convex polyhedron: it is the intersection of the half-spaces of points at least as close to p as to q, taken over all $q \in S$.

[2] The Delaunay triangulation of $S = \{X_1, \ldots, X_n\}$ is $\text{Delaunay}(S) = \{\sigma \subset S : \bigcap_{p \in \sigma} V_p \neq \emptyset\}$; this is the collection of those tuples σ of $d+1$ points whose Voronoi cells touch each other. Sets $\sigma \in \text{Delaunay}(S)$ are considered as the vertices of a simplex. Thus, a Delaunay triangulation is a collection of d-simpleces. A d-simplex is the convex hull of its $d+1$ vertices. In the two dimensional case ($d = 2$) the simpleces are triangles and in the three dimensional case ($d = 3$) the simpleces are tetrahedrons.

The article gives the following contributions: (1) We provide an algorithm for efficient computation and estimation of level set trees and volume functions. We argue that the algorithm of this article is at least as fast as the previous algorithms while providing statistically accurate estimates. (2) We define a useful density estimator, which leads to an efficient plug-in-estimator of level sets of a density. (3) The methods lead to feasible density based clustering. (4) We analyze flow cytometry data in an inventive way.

Algorithms of this article are implemented in R-packages "denpro" and "delt". These packages can be downloaded from the web page http://cran.r-project.org/ or from the page http://jussiklemela.com/denpro/ and the page http://jussiklemela.com/delt/. A tutorial for the application of the R-packages and some supplementary materials can be found at the home page of the article http://jussiklemela.com/art/volume/.

Section 2 contains a discussion of the previous related work. Section 3 describes the density estimators which are used to estimate level set trees and volume functions. Section 4 studies the properties of the discretized kernel estimator using simulations. Section 5 presents an application to a flow cytometry data. Section 6 offers a discussion.

2 Previous Work

Section 2.1 reviews level set estimation. Section 2.2 reviews level set tree based clustering because a large part of the literature related to level set trees has been written for the purpose of clustering. Section 2.3 reviews algorithms for the computation of level set trees.

2.1 Level Set Estimation

We apply the level sets of a discretized kernel estimator to construct level set trees. Other level set estimators that could be applied for our purposes should be such that they are able to estimate level sets with many connected components, which excludes the use of the convex hull estimator, and the piecewise polynomial estimator of the boundary function of a star shaped level set; see Korostelev and Tsybakov (1993) for a description of these classical level set estimators. An additional constraint for the level set estimators comes from the fact that for the purpose of visualization the level set estimator should be such that

the computation of the volumes, barycenters, and other characteristics of the connected components of the level sets is feasible.

For our purposes potentially useful level set estimators can be divided at least into four categories: (1) plug-in-estimators, where a level set estimator is a level set of a density estimator; (2) union of balls and related estimators; (3) union of polyhedrons and related estimators; (4) estimators defined as optimizers of the empirical excess mass.

Our level set estimator is a plug-in estimator, because the estimator is a level set of a discretized density estimator. Convergence rates of the plug-in-estimators when the density estimator is the kernel estimator has been studied for example in Baíllo et al. (2000), Baíllo et al. (2001), Cuevas et al. (2000), Cuevas et al. (2001), Cadre (2006), Cuevas et al. (2006), Biau et al. (2007), Burman and Polonik (2009), Rigollet and Vert (2009), Singh et al. (2009), and Rinaldo and Wasserman (2010).

It is natural to estimate a level set with a union of balls, centered at the observations whose estimated density is at least equal to the level λ of the level set. The support of a density can be considered as the zero-level level set, and the plug-in-estimator of the support is a union of balls when the density estimator is a kernel estimator whose kernel function has a compact ball-shaped support.[3] Properties of plug-in-estimators of the support were studied in Devroye and Wise (1980) and Cuevas and Fraiman (1997). A modified union-of-balls estimator of a level set has been studied in Walther (1997), where certain balls at the boundary are deleted. Although level set trees can be calculated when the level sets are estimated with a union of balls, the computation of the volume function requires an additional estimation step, because the computation of the volume of a union of balls has to be done numerically.

Delaunay partitions have been used in topological data analysis with point cloud data. The purpose has been to estimate topological properties of the support of a probability distribution when a sample of i.i.d. observations is available; see Zomorodian (2012). The direct application of Delaunay partitions in level set estimation is not feasible, because the

[3]When the kernel function is the standard normal density, then the support of the kernel estimator is the whole space $\mathbf{R}^d$. When the kernel function is the Bartlett density $C(1 - \|x\|^2)_+$, then the support is a union of balls, and when the kernel function is the Epanechnikov product density $C \prod_{i=1}^d (1 - x_i^2)_+$, then the support is a union of rectangles.

number of simpleces in the partition grows exponentially with the dimension of the data. Thus, Delaunay partitions do not seem to be better than regular grids. See Azzallini and Torelli (2007) for an application of Delaunay partitions in level set tree estimation. To make the Delaunay partition feasible it is necessary to restrict the number of sets in the partition. These kind of union of polyhedrons estimators were studied in Aaron (2013), where the size of the Delaunay partition was restricted by taking only those simpleces which fit inside a small ball, or only those simpleces which are such that the lengths of all edges are small.

The fourth class of potentially useful level set estimators consists of minimizers of an empirical risk. The excess mass criterion was proposed by Hartigan (1987) and Müller and Sawitzki (1991). See also Nolan (1991) and Polonik (1995), who derive rates of convergence for support estimation based on excess mass estimates. Mammen and Tsybakov (1995) study density support problem under a general setting of entropy conditions and Tsybakov (1997) extends the results to level set estimation. Klemelä (2004a) considers a method of support estimation using empirical risk minimization with an excess mass criterion, and uses a recursive splitting algorithm which is related to the splitting algorithm of this article.

2.2 Density Based Clustering

Density based clustering was proposed in (Hartigan, 1975, p. 205), where clusters were defined as regions of high density, separated from other such regions by regions of low density. Density based clustering is discussed in Fraley and Raftery (2002).

Stuetzle (2003) defines a cluster tree that can be defined as a level set tree with the levels such that the topology of the level set is changing at those levels. Density based clustering can be made feasible with the help of level set trees, as discussed in (Klemelä, 2009, Chapter 8.3). A level set tree can be used to define a cluster tree of observations whose nodes are associated with subsets of the observations. The cluster tree of observations has the same tree structure as the level set tree, but the node which is associated with set $A \subset \mathbf{R}^d$ in the level set tree is now associated with observations $X_i \in A$. The term likelihood tree was used for this cluster tree of observations in Klemelä (2009), because the observations are assigned to the nodes according to the estimated likelihood values $\hat{f}(X_i)$. Level set

trees generate high-density sample regions, whereas cluster trees of observations generate high-density clusters of data points.

In some cases it is the same thing to calculate the level set tree and the cluster tree of observations. This happens when the level sets are estimated as unions of balls, for example. However, there exist algorithms to derive a cluster tree of observations that do not lead to the corresponding level set tree.

On the other hand, we could construct a level set tree from a cluster tree of observations by associating each cluster of observations with the union of the Voronoi cells of the observations in the cluster. Other ad hoc constructions could also be used. In fact, the cluster of observations could be considered as a point cloud approximating the connected component of the level set. Level set trees obtained in this way may be not suitable for the computation of a volume function, for example due to the complexity of the Voronoi cells. Stuetzle and Nugent (2010) propose the estimation of the volume of a connected component A of a level set using

$$\text{volume}(A) = \int_A dx = E\left(\frac{I_A(X)}{f(X)}\right) \approx \frac{1}{n}\sum_{i=1}^{n}\frac{I_A(X_i)}{\hat{f}(X_i)},$$

where X is a random variable with density f, $X_1, \ldots, X_n$ is a sample from the distribution of f, and $I_A(x) = 1$ when $x \in A$, and $I_A(x) = 0$ otherwise. This estimator could be applied to calculate a volume function from a cluster tree of observations.

2.3 Algorithms for the Computation of a Level Set Tree

We do not develop new algorithms for the computation of level set trees in this article, but we show that the previously developed algorithms can be applied to calculate level set trees when the discretized kernel estimator is used. In particular, we argue that the Leafsfirst algorithm of Klemelä (2006) is a useful algorithm.

A level set tree is a concept closely related to a Reeb graph, whose definition is given in Reeb (1946). A Reeb graph is a graph whose nodes correspond to the local maxima, local minima, and to the points of the changing of the topology of the contours of the function, when a contour of a function is defined as $\{x \in \mathbf{R}^d : f(x) = \lambda\}$, where $\lambda \in \mathbf{R}$. First algorithms for the computation of a level set tree were given in the connection of the

computation of a Reeb graph. In fact, the computation of a Reeb graph can be reduced to the computation of a level set tree. Carr et al. (2003) suggest using the disjoint-set data structure (union-find data structure) of Tarjan (1976) to calculate a level set tree.[4]

We start with the descriptions of the algorithms for the computation of a cluster tree of observations, because these algorithms can sometimes be applied for the computation of a level set tree, too.

Stuetzle and Nugent (2010) present a three step algorithm. First, a density estimate is constructed and evaluated at the sample points. Second, a graph G is constructed whose vertices are the data points, and edges connect the data points. The edges are associated with weights. The weights of the edges are equal to the minimum value of the estimated density function on the line segment joining the observations. Third, for each density level λ a subgraph of G is constructed whose vertices are those data points whose estimated density value is larger or equal to λ, and there exists a connection between the two observations if the weight of the edge is larger or equal to λ. The connected components are searched in each subgraph. Chaudhuri and Dasgupta (2010) weight the edges according to the proximity of the sample points: observations are chosen to be inside a level set if their k-nearest neighbor estimate is high, and the graph contains an edge if the two observations are close in Euclidean distance. Menardi and Azzalini (2014) apply the method of Stuetzle and Nugent (2010) using a kernel density estimator with a locally adaptive smoothing parameter.

Kent et al. (2013) contains a review of the available algorithms for the computation of a cluster tree of observations. Kent et al. (2013) call methods where the edges are weighted edge iteration methods and those where the edges are unweighted point iteration methods. In the simplest form, there is an edge between the vertices for observations X_i and X_j if

[4] A Reeb graph of a function (or a contour tree of a function) is graph whose leaf vertices represent the local minima or maxima and each interior vertex represent the joining or splitting of the contours of the function. Carr et al. (2003) give the following procedure for the computation of a Reeb graph. First a level set tree and a lower level set tree are calculated. A lower level set tree is analogous to a level set tree, but its nodes correspond to the lower level sets $\{x \in \mathbf{R}^d : f(x) \leq \lambda\}$. Second, the level set tree and the lower level set tree are pruned so that only the leaf nodes and the nodes with more than one child are left. Third, the pruned level set tree (split tree) and the pruned lower level set tree (join tree) are combined to obtain the Reeb graph.

the distance between X_i and X_j is smaller than some threshold value, or if X_i and X_j are among each other's k-nearest neighbors. These algorithms were suggested in Maier et al. (2009) and Kpotufe and von Luxburg (2011), and implemented in Kent et al. (2013). Kent et al. (2013) use k-nearest neighbor density estimates and construct the k-nearest neighbor similarity graph on the set of observations.

The previously described algorithms for the computation of a cluster tree of observations can be applied for the computation of level set trees when the observations are replaced by regions of the sample space (for example, we can consider small balls centered at the observations). In fact, this idea was used in Ooi (2002), where the terminal rectangles of a recursive partitioning procedure are merged to make a binary tree, which is called a clustering tree. Ooi (2002) prunes the binary tree obtained from a recursive partitioning (which is called a density tree). An adjacency graph is constructed, where each vertex represents a terminal region, and the vertices for adjacent regions are connected by edges. A weight for each edge is defined. For example, a weight could be the arithmetic mean of the density values. The density values are taken to be empirical probabilities of the rectangles. The vertices are merged to get a spanning tree, which is called a clustering tree. A clustering tree is a binary tree obtained by merging the nodes of the density tree using the weighting function. The obtained clustering tree is conceptually close to a level set tree, but the levels of the nodes are not the values of the density but values of the weight function. The procedure leads to a clustering procedure but not to a level set tree, and the use of the histogram estimator as a density estimator seems not be statistically as efficient as the use of the kernel estimator.

The previously described algorithms require $O(dn^2)$ steps when the proximities for all pairs of n observations are calculated, because calculating a proximity takes at least d steps (for example, computation of the Euclidean distance takes d steps). Using a k-d-algorithm can reduce the number of steps to $O(dn \log n)$, but then the memory usage is $n^{O(d)}$; see Indyk (2004).[5] The algorithm of Stuetzle and Nugent (2010) can require $O(C_{n,d}n^2)$ steps, where $C_{n,d}$ is much larger than d, because the algorithm requires the

[5]Given a finite set $\mathcal{X}$ of points in $\mathbf{R}^d$ and a query point x, the k-d-algorithm finds the point in $\mathcal{X}$ closest to x. The k-d-algorithm uses a k-d-tree, which is a similar binary tree as we use to represent an adaptive partition.

estimation of the minimum value of a density estimate on the line segments joining the observations. For example, evaluation of a kernel density estimate at one point takes $O(nd)$ steps, and evaluation of of mixture of Gaussians takes $O(d)$ steps, but we need to evaluate the estimates at a large number of points. After the first proximity graph is calculated, the traveling of the graphs and the decomposition of the graphs into the connected components takes $O(n)$ steps, but this has to be done for each level set separately. Thus, the previous algorithms seem to work only for relatively small sample sizes, although they can work for high dimensional data.

We have applied the Leafsfirst algorithm of Klemelä (2006) to calculate level set trees. The algorithm was introduced in the connection of calculating level set trees for boundary functions of star shaped level sets, but it can be applied for the computation of level set trees of any functions. Unlike the previous algorithms, this algorithm does not start with a creation of a proximity graph. Instead, the algorithm starts at the rectangle with the highest density value, which is the first leaf node, merges the rectangles with the next highest density values if those rectangles are connected to the previous rectangles, and otherwise creates new leaf nodes. The worst case complexity is $O(dL^2)$, where L is the total number of rectangles. We choose typically $L = n$. However, the algorithm uses bounding boxes, which makes it possible to avoid making all pairwise comparisons between the rectangles (to find which rectangles touch each other), since we can first check whether the next rectangle touches the bounding boxes of the previously found connected components. Thus, the complexity of the algorithm can be much lower than $O(dL^2)$.

Leafsfirst algorithm can be used always when there is a collection of sets $A_1, \ldots, A_L \subset \mathbf{R}^d$ so that the level sets of $f : \mathbf{R}^d \to \mathbf{R}$ for levels $\lambda_1 < \cdots < \lambda_L$ can be written as

$$\Lambda(f, \lambda_l) = \bigcup_{j=l}^{L} A_j, \tag{1}$$

where $l = 1, \ldots, L$. Property (1) says that there exists a collection $A_1, \ldots, A_L$ of "elementary sets" so that all the level sets are unions of these elementary sets. The lowest level set is a union of all the elementary sets and higher level sets are unions of subsets of the elementary sets. Property (1) holds when level sets are estimated as unions of balls, unions of simpleces, and when using a regular or an adaptive partition of the sample space to rectangles.

A further algorithm was presented in Klemelä (2005). This algorithm assumes that the partition of the sample space is represented with a binary tree. This is the case for the adaptive partition of this article, because the construction of the adaptive partition is done using binary splits. Also, a regular partition can be represented with a binary tree. A level set can be divided into connected components with a dynamic programming algorithm that joins the connected sets by traveling the binary tree from the leaves to the root.

3 Density Estimators

We apply discretized kernel estimators to estimate the density function. The kernel density estimator $\hat{f}_0 : \mathbf{R}^d \to \mathbf{R}$, based on data $X_1, \ldots, X_n \in \mathbf{R}^d$, is defined as

$$\hat{f}_0(x) = \frac{1}{n} \sum_{i=1}^{n} K_h(x - X_i), \qquad x \in \mathbf{R}^d,$$

where $K_h(x_1, \ldots, x_d) = K(x_1/h_1, \ldots, x_d/h_d) / \prod_{i=1}^d h_i$, $K : \mathbf{R}^d \to \mathbf{R}$ is the kernel function, and $h = (h_1, \ldots, h_d)$ is the vector of positive smoothing parameters; see Silverman (1986).

Let $R \subset \mathbf{R}^d$ be a set containing the observations $X_1, \ldots, X_n$. We define a partition $\{A_1, \ldots, A_L\}$ of R. This means that $\cup_{l=1}^L A_l = R$ and $A_l \cap A_m = \emptyset$ for $l \neq m$. In our case the sets A_l are rectangles. Let x_l be the center point of A_l, $l = 1, \ldots, L$. The kernel density estimator $\hat{f}_0 : \mathbf{R}^d \to \mathbf{R}$ is evaluated at points x_l and we define the discretized kernel density estimator as

$$\hat{f}(x) = \sum_{l=1}^{L} \hat{f}_0(x_l)\, I_{A_l}(x), \tag{2}$$

where I_A is the indicator defined by $I_A(x) = 1$ when $x \in A$, and $I_A(x) = 0$ otherwise.

Besides kernel density estimators, similar discretized estimators could be defined for other density estimators, too. For example, we could take $\hat{f}_0$ to be the k-nearest neighbor density estimator, or a Gaussian mixture density estimator.

The smoothing parameter of the kernel density estimator will be chosen in this article using the the normal reference rule:

$$h_i = \left(\frac{4}{d+2} \right)^{1/(d+4)} n^{-1/(d+4)} \hat{\sigma}_i,$$

for $i = 1, \ldots, d$, where $\hat{\sigma}_i$ is the sample standard deviation for the $i$the variable; see (Silverman, 1986, page 45). The kernel function will always be the standard normal density.

There exists a large collection of methods for the choice of the smoothing parameter. These methods include the plug-in method, which chooses the smoothing parameter minimizing an estimate of the asymptotic mean integrated squared error, and the cross-validation method, which chooses the smoothing parameter minimizing a cross-validation estimate of the mean integrated squared error. These methods may improve the normal reference rule, but they increase computational complexity. A second possibility to improve kernel density estimation is to use a matrix of smoothing parameters, instead of using a scalar or a vector of smoothing parameters. However, the method of using a matrix of smoothing parameters seems to be implemented only for two or three dimensional data, whereas we are interested in higher dimensional data. A third possibility to improve kernel density estimation is to choose the smoothing parameter differently at each point $x \in \mathbf{R}^d$ of estimation, but we have not implemented this method. See Scott (1992) for more about kernel density estimation and smoothing parameter selection.

3.1 Regular and Delaunay Partitions

We call a partition regular if it is obtained as follows. First one finds the smallest rectangle with sides parallel to the coordinate axes that contains the observations. Then each side is divided into $[L^{1/d}]$ intervals to obtain a partition of cardinality approximately equal to L. The partition consists of the small rectangles obtained as a product of the intervals.

We compare the adaptive partition kernel estimator only with the regular partition kernel estimator, but it should be noted that definition (2) contains as special cases many other partitions. For example, computation of level set trees and volume functions of the Delaunay partition kernel estimator is implemented in the package "denpro", but we do not study this estimator in this article, because it seems to perform worse than the regular partition kernel estimator (the size of the Delaunay partition is $L = O\left(n^{\lfloor (d+1)/2 \rfloor}\right)$; see McMullen (1970)).

3.2 Adaptive Histogram Partitions

A greedy partition is a partition of the sample space which is found by a stepwise algorithm, that recursively splits the space into finer sets. This algorithm is called greedy, or stepwise, because it does not try to find a global minimum for the optimization problem but finds the optimizer one step at a time. Morgan and Sonquist (1963) presented this type of algorithm for regression function estimation, although they did not restrict themselves to binary splits but allowed a large number of splits to be made simultaneously. CART procedure for regression and classification is described in Breiman et al. (1984). The corresponding procedures for density estimation are described in (Klemelä, 2009, Chapter 17.2). Ooi (2002) defines a recursive partitioning similar to the one we apply here, but the partition is applied to calculate a histogram estimator and not a discretized kernel estimator, and the histogram estimator is used to calculate a clustering tree, where level sets are not estimated, although the clustering tree comes rather close to a level set tree.

We do not use a histogram estimator but a discretized kernel estimator; only the partition of the discretization is constructed with the help of a histogram. We define the histogram corresponding to the partition $\mathcal{P} = \{A_1, \ldots, A_L\}$ as

$$\hat{f}(x, \mathcal{P}) = \sum_{l=1}^{L} \frac{n_l/n}{\text{volume}(A_l)} I_{A_l}(x), \tag{3}$$

where n_l is the number of observations in A_l.

First we define the split points over which we search the best splits. The splits are made parallel to the coordinate axes and thus we have to define a grid of possible split points for each direction. Let us denote the sets of possible split points by

$$\mathcal{G}_1, \ldots, \mathcal{G}_d, \tag{4}$$

where $\mathcal{G}_k \subset \mathbf{R}$ is a finite grid of split points in direction k. It is natural to choose $\mathcal{G}_k$ to be set of the midpoints of the coordinates of the observations: $\mathcal{G}_k = \{Z_1^k, \ldots, Z_{n-1}^k\}$, $k = 1, \ldots, d$, where Z_i^k is the midpoint of $X_{(i)}^k$ and $X_{(i+1)}^k$:

$$Z_i^k = \frac{1}{2}\left(X_{(i)}^k + X_{(i+1)}^k\right),$$

where $X_{(1)}^k, \ldots, X_{(n)}^k$ is the order statistic of the kth coordinate of the observations $X_1, \ldots, X_n$.

When a rectangle $R \subset \mathbf{R}^d$ is split through the point $s \in \mathbf{R}$ in direction $k = 1, \ldots, d$, we obtain sets

$$R_{k,s}^{(0)} = \{(x_1, \ldots, x_d) \in R : x_k \leq s\} \tag{5}$$

and

$$R_{k,s}^{(1)} = \{(x_1, \ldots, x_d) \in R : x_k > s\}. \tag{6}$$

The split point s satisfies

$$s \in S_{R,k} \stackrel{def}{=} \mathcal{G}_k \cap \mathrm{proj}_k(R), \tag{7}$$

where $\mathrm{proj}_k(R) = R_k$, when $R = R_1 \times \cdots \times R_d$. We say that partition $\mathcal{P}$ is grown if it is replaced by partition

$$\mathcal{P}_{R,k,s} = \mathcal{P} \setminus \{R\} \cup \left\{R_{k,s}^{(0)}, R_{k,s}^{(1)}\right\}, \tag{8}$$

where rectangle $R \in \mathcal{P}$ is split in direction $k \in \{1, \ldots, d\}$ through the point $s \in S_{R,k}$.

We choose partition $\mathcal{P}^*$ using the following procedure. Let the minimal observation number be $m \geq 1$.

1. Start with the partition $\mathcal{P} = \{R\}$, where R is the smallest rectangle containing the observations and whose sides are parallel to the coordinate axes.

2. Suppose we have constructed a partition $\mathcal{P}$. If all $R \in \mathcal{P}$ satisfy $\#\{X_i \in R\} \leq m$, then splitting is finished. Otherwise, we choose $R \in \mathcal{P}$ with $\#\{X_i \in R\} > m$ and construct the new partition $\mathcal{P}_{R,\hat{k},\hat{s}}$, where

$$\left(\hat{k}, \hat{s}\right) = \mathrm{argmax}_{(k,s) \in I_R} \sum_{i=1}^{n} \log \hat{f}(X_i, \mathcal{P}_{R,k,s}),$$

   where $I_R = \{(k, s) : k = 1, \ldots, d,\ s \in S_{R,k}\}$, $S_{R,k}$ is the set of split points defined in (7), $\mathcal{P}_{R,k,s}$ is the partition defined in (8), and $\hat{f}(\cdot, \mathcal{P})$ is the histogram defined in (3).

In the worst case the number of the steps of the algorithm is

$$O\left(dn^2\right). \tag{9}$$

In fact, to find the first split requires dn steps. The worst case happens when one of the children contains one observation and the other child contains $n - 1$ observations. Then,

to find the second split requires $d(n-1)$ steps. To find the third split requires in the worst case $d(n-2)$ steps, and so on. The complexity in (9) holds because $\sum_{i=1}^{n} i = n(n+1)/2$.

In the best case the number of the steps of the algorithm is

$$O\left(dn \log n\right). \tag{10}$$

Again, to find the first split requires dn steps. The best case occurs when the two child rectangles contain the same number of observations. Then, to find the next two splits takes $d(n/2)$ steps. In the best case, to find the next four splits takes $d(n/4)$ steps, and so on. The complexity in (10) holds because in the best case we need $\log_2 n$ resolution levels to obtain rectangles which contain only one observation.

We can define a faster algorithm where only dyadic splits are made, so that splits are such that the number of observations in the child rectangles is about the half of the number of observations in the parent rectangle. This algorithm would take $O\left(dn \log n\right)$ steps. Holmström et al. (2014) shows that this estimator can have a significantly larger mean integrated squared error than the estimator that allows a larger number of possible split points.

When the parameter m of the algorithm is chosen large, then the computation is faster. We can handle large sample sizes by using large values of m. Holmström et al. (2014) studies how the mean integrated squared error of the estimator increases when m increases.

Figure 1 illustrates the construction of the partition. In panel (a) we show the collection of the elementary sets when there are 10 observations. The four of the 10 observations define the boundary, and the rest 6 observations define the possible splitting points along the coordinate axes.[6] Panel (b) shows the partition which is obtained when $m = 1$, so that there is exactly one observation inside each rectangle.

Figure 2 shows examples of partitions. In panel (a) we have generated 50 observations from the standard normal distribution. In panel (b) we have simulated 500 observations from an equal mixture of standard normal distributions. In both cases $m = 1$, so that each bin contains exactly one observation.

[6]Note that when observations are in $\mathbf{R}^d$ it is possible that only d of the observations are on the boundary, which happens when d observations are on the corners of the smallest rectangle containing the observations.

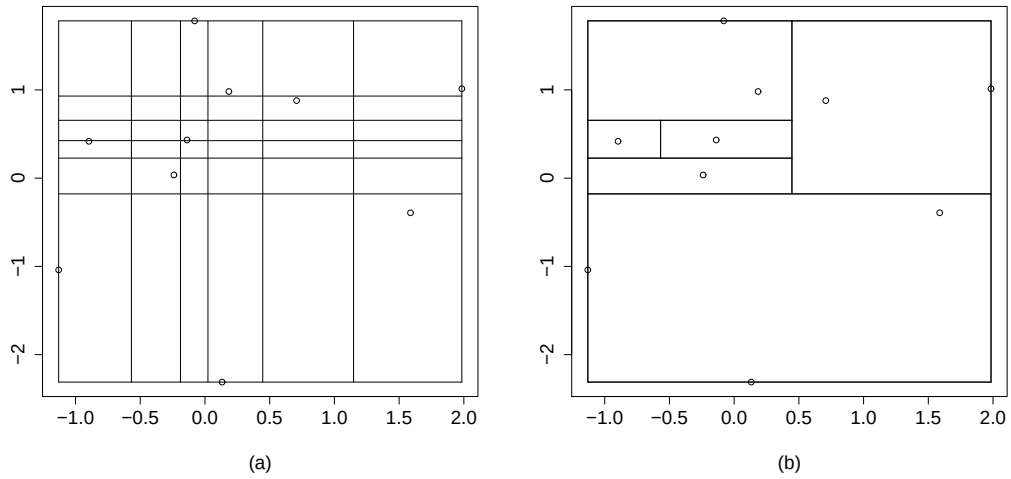

Figure 1: *A partition in 2D.* (a) The collection of split points when there are 10 observations. (b) An adaptive grid obtained from the split points in frame (a).

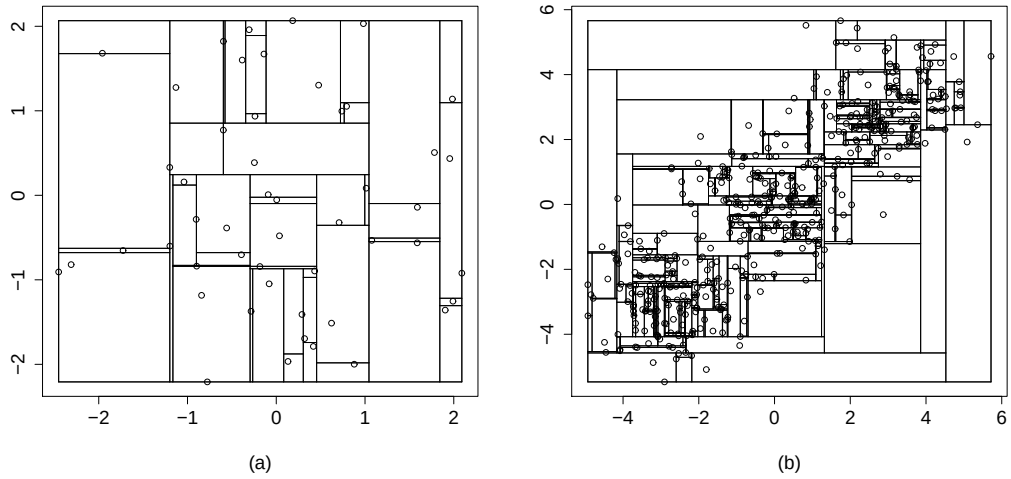

Figure 2: *Partitions in 2D.* (a) An adaptive grid obtained from 50 observations generated from the standard normal distribution. (b) An adaptive grid obtained from 500 observations which were simulated from a mixture of three Gaussians.

4 Simulations

Section 4.1 compares the mean integrated squared errors of the regular grid kernel estimator and the adaptive grid kernel estimator. Section 4.2 shows an example of a volume function of a 7-dimensional density estimate, and Section 4.3 shows an example of a volume function of a 4-dimensional density estimate. These examples also illustrate how level set trees can be used in density based clustering.

4.1 MISE Comparison

The mean integrated squared error (MISE) is defined as

$$\mathrm{MISE}(f, \hat{f}) = E \int_{\mathbf{R}^d} \left(f - \hat{f} \right)^2,$$

where $f : \mathbf{R}^d \to \mathbf{R}$ is the true density function and $\hat{f} : \mathbf{R}^d \to \mathbf{R}$ is its estimator. For a piecewise constant estimator $\hat{f}(x) = \sum_{l=1}^{L} a_l I_{A_l}(x)$ the integrated squared error can be easily calculated. Indeed, we have that $\mathrm{ISE}(f, \hat{f}) = \int_{\mathbf{R}^d} f^2 + \int_{\mathbf{R}^d} \hat{f}^2 - 2 \int_{\mathbf{R}^d} f\hat{f}$, and

$$\int_{\mathbf{R}^d} \hat{f}^2 = \sum_{l=1}^{L} a_l^2 \cdot \mathrm{volume}(A_l), \qquad \int_{\mathbf{R}^d} \hat{f} f = \sum_{l=1}^{L} a_l \int_{A_l} f.$$

The probability of a rectangle $A = [a_1, b_1] \times \cdots \times [a_d, b_d]$ is

$$\int_A f = \sum_{i_1=1}^{2} \cdots \sum_{i_d=1}^{2} (-1)^{i_1 + \cdots + i_d} F(u_{1,i_1}, \ldots, u_{d,i_d}),$$

where $u_{i,1} = a_i$ and $u_{i,2} = b_i$ for $i = 1, \ldots, d$, and $F : \mathbf{R}^d \to \mathbf{R}$ is the distribution function. The MISE can be estimated by generating $M \geq 1$ Monte Carlo samples of size n from the distribution with density f, calculating ISE for each sample, and taking the arithmetic mean of the M values of ISE. We will use $M = 100$ Monte Carlo samples.

Figure 3 shows the MISE of discretized kernel estimators as a function of the sample size when the true density is the standard normal density. In panel (a) $d = 2$ and in panel (b) $d = 3$. The black curves with labels "o" show the MISE of the discretized kernel estimator with an adaptive grid. The partition of the adaptive grid is such that there is exactly one observation in each cell. The red curves with labels "1" show the case of the regular grid with 10^d cells. The blue curves with labels "2" show the case of 20^d cells. The green curve

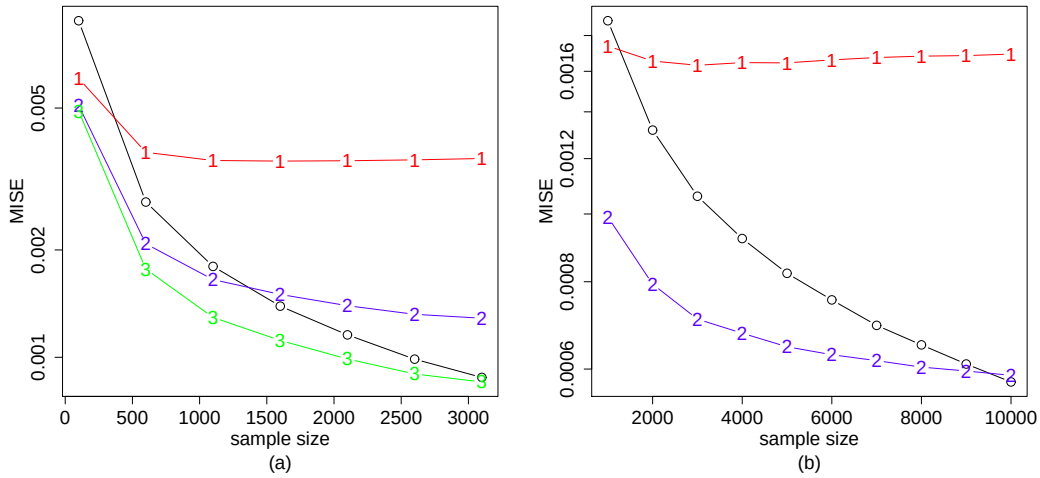

Figure 3: *MISE as a function of the sample size for the standard normal density.* (a) $d = 2$; (b) $d = 3$. The black curves show the MISE of the adaptive grid kernel estimator. The red curves show the regular grid kernel estimator with 10^d cells, the blue curves show the case of 20^d cells, and the green curve shows the case of 30^d cells.

with labels "3" in panel (a) shows the case of 30^d cells. When $d = 2$, the adaptive grid is better than regular grid with 10^d cells when the sample size is about 500, the adaptive grid is better than regular grid with 20^d cells when the sample size is about 1200, and the adaptive grid is better than regular grid with 30^d cells when the sample size is about 3000. When $d = 3$, then the adaptive grid is better than regular grid with 10^d cells when the sample size is about 500, the adaptive grid is better than regular grid with 20^d cells when the sample size is about 10 000.

Figure 4 shows the MISE of discretized kernel estimators as a function of the sample size when the true density is the equally weighted mixture of the standard normal density $\phi(x)$ and the density $\phi(x/\sigma)/\sigma^d$, where $\sigma = 0.1$. This density is spatially inhomogeneous because it has a sharp peak at the origin. In panel (a) $d = 2$ and in panel (b) $d = 3$. The black curves with labels "o" show the MISE of the discretized kernel estimator with the adaptive grid. The partition of the adaptive grid is again such that there is exactly one observation in each cell. The red curves with labels "1" show the case of the regular grid with 10^d cells. The blue curves with labels "2" show the case of 20^d cells. The green

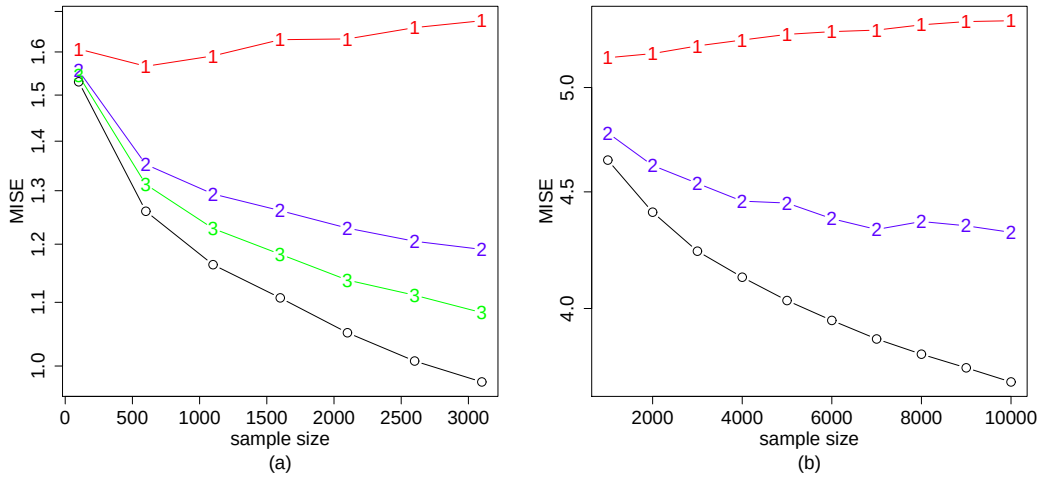

Figure 4: *MISE as a function of the sample size for an inhomogeneous density.* (a) $d = 2$; (b) $d = 3$. The black curves show the MISE of the adaptive grid kernel estimator. The red curves show the regular grid kernel estimator with 10^d cells, the blue curves show the case of 20^d cells, and the green curve shows the case of 30^d cells.

curve with labels "3" in panel (a) shows the case of 30^d cells. The adaptive grid leads to uniformly better MISE values than the regular grid.

Note that the computational complexity of the estimator with an adaptive grid of size n is roughly equal to the computational complexity of the estimator with a regular grid when the size of the grid is N^d, where $N \approx n^{1/d}$. Thus, the MISE computations can tell which of the two estimators with equal computational complexity has a better accuracy.

4.2 7D Example

Figure 5 considers a sample of 10 000 observations in dimension $d = 7$. The data are generated from the mixture of eight normal distributions with the marginal standard deviations 0.25 and uncorrelated components. The modes are located at the eight vertices of the unit simplex.[7] Panel (a) shows the volume function of a discretized kernel estimator with an

[7]The unit simplex is defined as the convex hull of the origin and the vertices $e_1, \ldots, e_d$, where $e_i \in \mathbf{R}^d$ has 1 in the ith position and 0 in the other positions. Klemelä (2004b) used a simulation example where the simplex was such that all edges have the same length. The current simulation example was used in

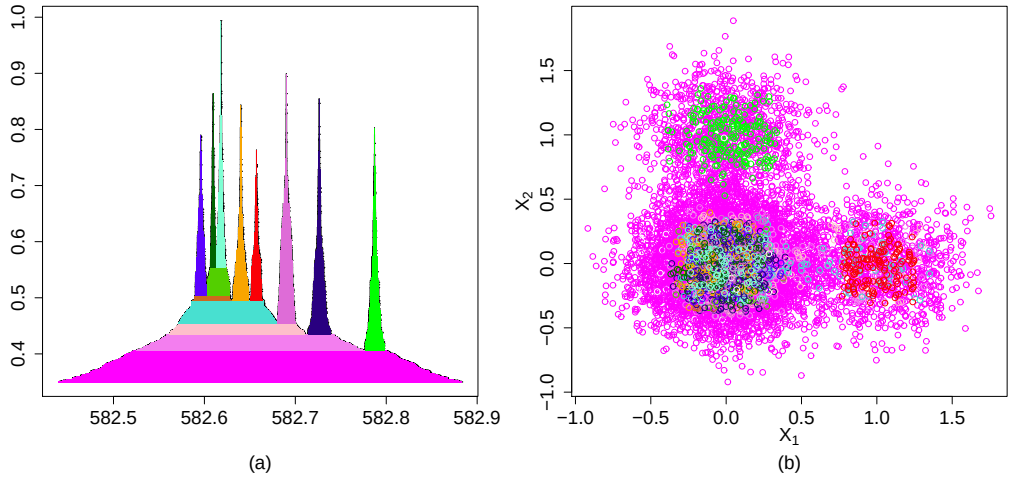

Figure 5: *A volume function and a scatter plot for 7D data.* (a) A volume function drawn from a density estimate of a 8-modal density. Only the part of the volume function above level 0.35 is shown. (b) A scatter plot of the 1st and the 2nd coordinates of the observations.

adaptive partition. Panel (b) shows a scatter plot of the 1st and the 2nd coordinates of the data. The complete scatter plot matrix is shown in Holmström et al. (2014). We show only the part of the graph of the volume function which is above level 0.35. The cells with 5 or less observations were not split anymore. The volume function and the scatter plot are colored so that the observations have the same color as the part of the region in the graph of the volume function that corresponds to the smallest connected component of the level set where the observation is included. To make the coloring easier to interpret, we have pruned the modes with small excess mass from the estimate, so that only the 8 largest modes were left.

Colored volume functions can be combined with the colored scatter plot matrices to provide a visual and a computational tool to make density based clustering easier to apply. Kent et al. (2013) use similar coloring techniques. They calculate cluster trees of observations instead of level set trees, and do not draw volume functions. They enhance basic colored cluster trees of observations with the frequencies of the clusters, instead of the volumes of the connected parts of level sets.

Stuetzle and Nugent (2010) and Menardi and Azzalini (2014) with the sample size about 500.

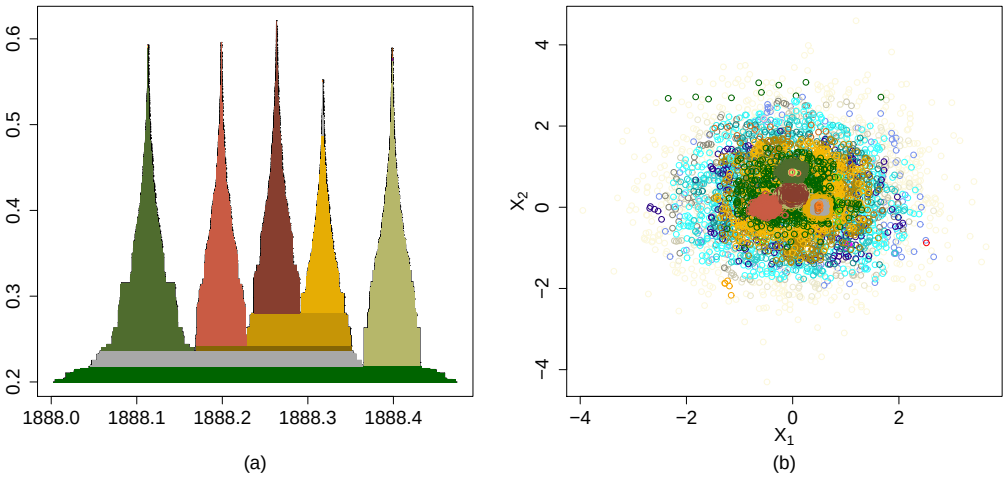

Figure 6: *A volume function and a scatter plot for 4D data.* (a) A volume function drawn from a density estimate of a 5-modal density. Only the part of the volume function above level 0.2 is shown. (b) A scatter plot of the 1st and the 2nd coordinates of the observations.

4.3 4D Example

Figure 6 considers a sample of 10 000 observations in dimension $d = 4$. The data are generated from the equally weighted mixture of five peaked distributions. Each peaked distribution is, in turn, defined as the equally weighted mixture of the standard normal density $\phi(x)$ and the density $\phi(x/\sigma)/\sigma^d$, where $\sigma = 0.1$. These peaked distributions were used in the MISE computation of Figure 4. (Thus, the underlying density is a mixture of ten normal distributions.) The modes are located at the five vertices of a simplex.[8] Panel (a) shows the volume function of a discretized kernel estimator with an adaptive partition. Panel (b) shows a scatter plot of the 1st and the 2nd coordinates of the data. The complete scatter plot matrix is shown in Holmström et al. (2014). We show only the part of the graph of the volume function which is above level 0.2. The cells with 5 or less observations were not further split.

[8]The modes are located at $(1/2, 0, 0, 0)$, $(-1/2, 0, 0, 0)$, $(0, \sqrt{3}/2, 0, 0)$, $(0, 1/(2\sqrt{3}), \sqrt{2/3}, 0)$, and $(0, 1/(2\sqrt{3}), 1/(2\sqrt{6}), \sqrt{15/24})$.

5 Application to Flow Cytometry Data

Section 5.1 describes the basics of flow cytometry (FCM), the nature of FCM data and how they are traditionally analyzed. Section 5.2 presents our level set tree based approach to FCM data analysis and reports results obtained from the analysis of FCM data previously discussed in Aghaeepour et al. (2011).

5.1 Flow Cytometry

A flow cytometer is a laser-based instrument that measures several biophysical and chemical characteristics of thousands of cellular particles per second. Optical excitation of the biological cells takes place in a small flow chamber that hydrodynamically aligns the cells so that they pass single file through the laser beam. Each interaction between the laser beam and a cell creates scattered light and fluorescence. These signals are then turned into multichannel data used to classify the cells.

The history of flow cytometry goes back to the 1960s and the continuous refinement of the technique during the last 50 years has produced an extremely sensitive, fast, and versatile methodology for clinical cell analyses. Biochemical methods offer a variety of possibilities to stain cells in a FCM sample tube. FCM protocols can be used for example to diagnose many cancers and diseases. For a comprehensive review of all aspects of flow cytometry, see for example Melamed et al. (1990) and Shapiro (2003).

Based on light scattering and laser-induced fluorescence, and employing simultaneously several laser light sources in state-of-the-art instruments, as many as 20 different spectroscopic channels can be used in the measurements, resulting in as many features used to characterize the properties of a cell. The number of analyzed cells may well exceed 10^5 per sample. In addition to relevant cellular data, the sample may also include dead cells and other debris.

The FCM measurements and the associated statistical data analyses aim to produce a useful clustering of the cells based on their measured features. The clustering thus produced can also be incomplete leaving some cells unclassified. Clusters of cells can indicate for example the balance of various cell types in the sample, an incipient disease or the stage of a disease already in progress. Cancer represents one of the most important uses of FCM

in clinical use.

Clustering has traditionally been carried out manually by examining one or two-dimensional marginal plots of the feature vectors. This process, usually referred to as 'gating', often proceeds as follows. First a pair of variables is selected and clusters are defined in the 2D scatter plot. Then a next pair of variables is considered, the clusters are refined and the process continues until a satisfactory classification of the sample has been reached. When used by experts such a method can produce good results although there are obvious caveats related to the use of a sequence of 2D projections in the analysis of high dimensional data. In addition, the rapid increase in the use of flow cytometry analyses has begun to generate large numbers of data sets that are difficult or impossible to analyze by such traditional methods, cf. Bashashati and Brinkman (2009) and O'Neill et al. (2013).

5.2 Level Set Tree Clustering of FCM Data

Many automated procedures have been developed for clustering of FCM data. The performance of various algorithms was reviewed in Aghaeepour et al. (2013). The k-means algorithm has found the most applications. Model-based clustering using finite Gaussian mixture models, or other mixture models, has also been popular. Lo et al. (2008) study both Gaussian mixture models and t-mixture models. Ge and Sealfon (2012) introduce the flowPeaks algorithm that combines k-means clustering and Gaussian mixture density estimation. The Gaussian mixture is based on the cluster centers defined by the k-means algorithm and the mixture is then used in a cluster merging procedure to produce the final classification of the data.

Both k-means and mixture models lead typically to convex clusters, whereas clustering with nonparametric density estimation can find clusters of any shape. Walther et al. (2009) constructs a regular grid with associated weights that are derived by binning the data. The density of a grid point is estimated using a kernel density estimator. Bins are clustered by assigning them to the local maximum of the density estimate that is at the end of a chain of bins of progressively higher estimated density. The use of a regular grid limits the use of the method to lower dimensional cases. Walther et al. (2009) suggest to use the method sequentially for two-dimensional projections, and to use two-dimensional scatter plots for

visualization. The curvHDR method of Naumann et al. (2010) mimics manual gating by applying kernel density estimation in a two-stage procedure that first finds regions of high negative curvature using the Hessian of the density estimate; see also Duong et al. (2008). A kernel density estimate of the data in a neighborhood of each negative curvature region is then computed and clusters are defined by their highest density regions.

We propose to apply level set trees to gate FCM data. Karttunen et al. (2014) used such an approach in FCM data analysis. The example data considered here are six dimensional and they were generated in FCM analysis of graft-versus-host disease (GvHD), a common complication in tissue transplants. Two of the variables measure forward and side scattering (FSC.H and SSC.H) and four are associated with fluorescence channels (FL1.H, FL2.H, FL3.H and FL4.H). The sample size is 17 640. The data set has been analyzed earlier in Aghaeepour et al. (2011) and it is available as a part of the flowMeans R-package Aghaeepour (2010). GvHD data are analyzed also in Lo et al. (2008).

The data were first scaled and edge values removed leaving sample size of 12 160. Then the discretized kernel estimator with an adaptive histogram partition was applied. The estimator is described in Section 3. The normal reference rule was used to choose the smoothing parameters of the kernel estimator. The volume function of the discretized kernel estimate is shown in Figure 7. It shows that the density function has several modes of different shapes and sizes. The volume function suggests the presence of dense, sparse, narrow, and broad clusters of data with varying degree of overlap in the six dimensional data space.

The clusters identified by the volume function are presented in Figure 8 using traditional scatter plots of pairs of FCM variables; only four of the variables are considered and the on-line supplement to this article includes a scatter plot matrix of all six variables used in our example. The coloring schemes are the same in Figure 7 and in the lower left part of Figure 8. The dense prominent clusters are clearly visible in both figures but the relative size and locations of smaller clusters, such as those displayed in lighter green and blue, can be inferred much more effectively from the volume function. A small and potentially interesting cluster, such as the one shown in red in Figure 7, can be totally obscured in a scatter plot. The scatter plots of the clusters produced by the application of the flowMeans

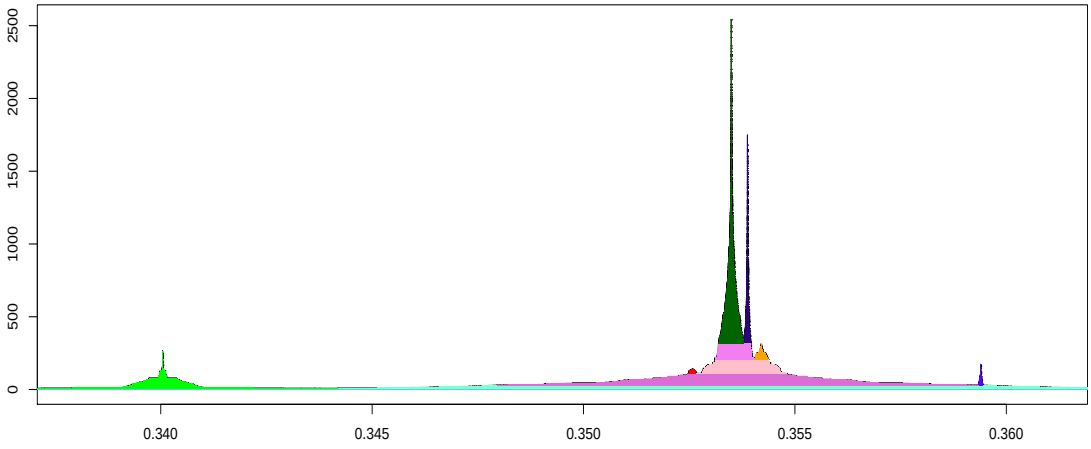

Figure 7: *Volume function of FCM data.* The six modes (partial clusters) are colored individually. Some of the modes rise from common supporting level sets that are also colored individually.

algorithm to these data are shown on page 10 of the supplement to Aghaeepour et al. (2011).

Yet another level set tree based visualization technique is a barycenter plot where each variable is considered separately and the centers of mass of the connected components of its level sets are displayed as a function of the level λ in a tree-like structure; see Klemelä (2004b). A barycenter plot allows one to track the locations of the emerging modes when λ is increased. The on-line supplement includes barycenter plots of all six FCM variables considered here.

6 Discussion

We have developed an algorithm for the computation of a level set tree whose worst case complexity is $O(dn^2)$, where n is the sample size and d is the dimension of the observations, but the typical complexity can be much lower.

We calculate a "genuine level set tree", whose nodes are associated with connected

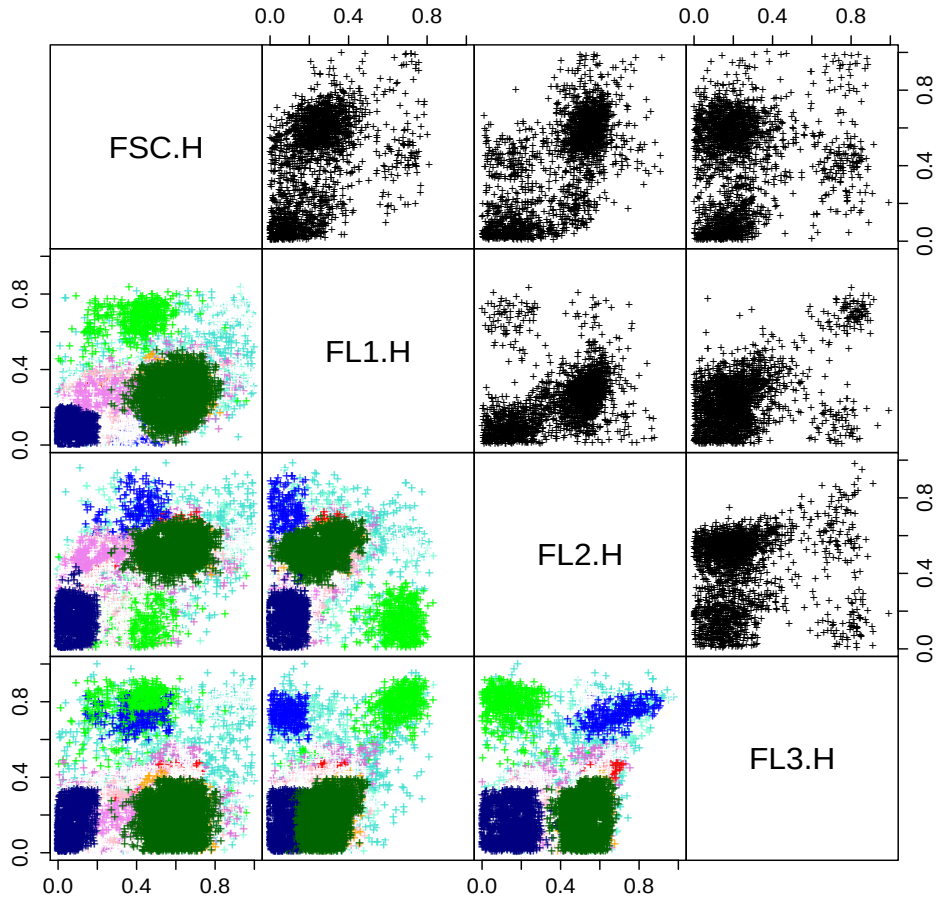

Figure 8: *Scatter plots of FCM data.* The coloring scheme applied in the lower left part of the plot is the same as in Figure 7. To emphasize the clusters, the densest regions are plotted last, possibly covering sparser cluster regions. In the upper right panels the sample size is 2000 to diminish the problem of overplotting, but the lower left panels show all observations.

components of the level sets of a density function, whereas most previous algorithms calculate a "cluster tree of observations", whose nodes are annotated with subsets of the observations. The genuine level set tree is needed to make statistical inference about the existence of modes and clusters, and in the visualization of the density function using volume functions, barycenter plots, and other characteristics of the connected components of the level sets. Our algorithm works for moderate dimensional cases and for large sample sizes whereas the algorithms for the computation of a cluster tree of observations seem to work in higher dimensional cases but for small sample sizes.

We have studied the computation of a level set tree of a density function. However, a level set tree can be useful for other functions than just densities. For example, it is of interest to compute level set trees of regression function estimates, to study which x-values lead to large y-values. A regression function is estimated using data $(X_1, Y_1), \ldots, (X_n, Y_n)$, where $X_i \in \mathbf{R}^d$ are observations of an explanatory variable, and $Y_i \in \mathbf{R}$ are observations of a response variable. The observations $X_1, \ldots, X_n$ can be used to generate an adaptive partition just as in the case of density estimation. However, the grid does not have the property that there are more observations in the neighborhoods of the local maxima of the regression function, and this can make the estimation of the level set tree less accurate. Thus, it may be reasonable to generate another set of points $\tilde{X}_1, \ldots, \tilde{X}_N$, which is such that there are more points around those X_i for which $\hat{f}(X_i)$ is large. When we want to calculate a level set tree for an arbitrary function $f : \mathbf{R}^d \to \mathbf{R}$, which can be evaluated at the points $x_1, \ldots, x_n$ of our choice, then we conjecture that the points $x_1, \ldots, x_n$ could be generated by an algorithm that searches local maxima, which leads to a grid where there are more points around the local maxima.

Our analysis of flow cytometry data has shown that using level set tree based techniques, such as volume functions and barycenter plots, we can detect and visualize the sizes (in the sense of excess mass) and relative locations of the clusters in a way that would not be possible using only scatter plots or for example tree plots of the level set tree. We conjecture that the additional information provided by a volume function can be important in helping the practitioner to detect abnormal clustering of cells, for example. In addition, barycenter plots can usefully complement data analyses by helping to track modes and their locations

through simple and easy-to-interpret graphics. We therefore believe that visualization techniques derived from level set tree analysis can provide insights into complex data, such as those generated by FCM, that traditional approaches based on pairwise scatter plots cannot offer.

To our knowledge, the R-packages "denpro" and "delt" are the only available software-packages to calculate level set trees and volume functions. For the computation of cluster trees of observations there exists R-package "gslclust", described in Stuetzle and Nugent (2010), Python-package "DeBaCl", described in Kent et al. (2013), and R-package "pdfCluster", described in Menardi and Azzalini (2014).

References

Aaron, C. (2013). Estimation of the support of the density and its boundary using random polyhedrons. Technical report, Université Blaise Pascal.

Aghaeepour, N. (2010). *FlowMeans: Non-parametric flow cytometry data gating*. R package version 1.16.0.

Aghaeepour, N., G. Finak, H. Hoos, T. R. Mosmann, R. Brinkman, R. Gottardo, and R. H. Scheuermann (2013). Critical assessment of automated flow cytometry data analysis techniques. *Nature Methods 10*(3), 228–238.

Aghaeepour, N., R. Nikolic, H. H. Hoos, and R. R. Brinkman (2011). Rapid cell population identification in flow cytometry data. *Cytometry. Part A : J. Int. Soc. Analytical Cytology 79*(1), 6–13.

Azzallini, A. and N. Torelli (2007). Clustering via nonparametric density estimation. *Statistics and Computing 17*, 71–80.

Baíllo, A., J. A. Cuesta-Albertos, and A. Cuevas (2001). Convergence rates in nonparametric estimation of level sets. *Statist. Probab. Lett. 53*, 27–35.

Baíllo, A., A. Cuevas, and A. Justel (2000). Set estimation and nonparametric detection. *Canadian J. Statist. 28*, 765–782.

Bashashati, A. and R. R. Brinkman (2009). A survey of flow cytometry data analysis methods. *Advances in Bioinformatics*, 584–603.

Biau, G., B. Cadre, and B. Pelletier (2007). A graph-based estimator of the number of clusters. *ESAIM Probab. Stat. 11*, 272–280.

Breiman, L., J. Friedman, R. Olshen, and C. J. Stone (1984). *Classification and Regression Trees*. New York: Chapman and Hall.

Burman, P. and W. Polonik (2009). Multivariate mode hunting: Data analytic tools with measures of significance. *J. Multivariate Anal. 100*, 1198–1218.

Cadre, B. (2006). Kernel estimation of density level sets. *J. Multivariate Anal. 97*(4), 999–1023.

Carr, H., J. Snoeyink, and U. Axen (2003). Computing contour trees in any dimension. *Comput. Geometry: Theory Appl. 24*(2), 75–94.

Chaudhuri, K. and S. Dasgupta (2010). Rates of convergence for the cluster tree. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta (Eds.), *Advances in Neural Information Processing Systems 23*, pp. 343–351. Vancouver, BC: Curran Associates.

Cuevas, A., M. Febreiro, and R. Fraiman (2000). Estimating the number of clusters. *Canad. J. Statist. 28*, 367–382.

Cuevas, A., M. Febreiro, and R. Fraiman (2001). Cluster analysis: A further approach based on density estimation. *Comput. Stat. Data Anal. 36*, 441–459.

Cuevas, A., M. Febreiro, and R. Fraiman (2006). Plug-in estimation of general level sets. *Aust. N. Z. J. Stat. 48*, 7–19.

Cuevas, A. and R. Fraiman (1997). A plug-in approach to support estimation. *Ann. Statist. 25*, 2300–2312.

Devroye, L. and G. L. Wise (1980). Detection of abnormal behavior via nonparametric estimation of the support. *SIAM J. Appl. Math. 38*, 480–488.

Duong, T., A. Cowling, I. Koch, and M. P. Wand (2008). Feature significance for multivariate kernel density estimation. *Comput. Statist. Data Anal. 52*(9), 4225–4242.

Fraley, C. and A. E. Raftery (2002). Model-based clustering, discriminant analysis and density estimation. *J. Am. Statist. Assoc. 97*, 611–631.

Ge, Y. and S. C. Sealfon (2012). Flowpeaks: a fast unsupervised clustering for flow cytometry data via k-means and density peak finding. *Bioinformatics 28*(15), 2052–2058.

Hartigan, J. A. (1975). *Clustering Algorithms*. New York: Wiley.

Hartigan, J. A. (1987). Estimation of a convex density cluster in two dimensions. *J. Am. Statist. Assoc. 82*, 267–270.

Holmström, L., K. Karttunen, and J. Klemelä (2014). Estimation of level set trees with adaptive partitions: Supplementary material. Technical report, University of Oulu.

Indyk, P. (2004). Nearest neighbors in high-dimensional spaces. In J. E. Goodman and J. O'Rourke (Eds.), *Handbook of Discrete and Computational Geometry*, pp. 877–892. Boca Raton, FL: Chapman & Hall/CRC.

Karttunen, K., L. Holmström, and J. Klemelä (2014). Level set trees with enhanced marginal density visualization. *In Proceedings of the 5th International Conference on Information Visualization Theory and Applications, (IVAPP 2014), Lisbon, Portugal*, 210–217.

Kent, B. P., A. Rinaldo, and V. Timothy (2013). DeBaCl: A Python package for interactive DEnsity-BAsed CLustering. *J. Statist. Software*. To appear.

Klemelä, J. (2004a). Complexity penalized support estimation. *J. Multivariate Anal. 88*, 274–297.

Klemelä, J. (2004b). Visualization of multivariate density estimates with level set trees. *J. Comput. Graph. Statist. 13*(3), 599–620.

Klemelä, J. (2005). Algorithms for the manipulation of level sets of nonparametric density estimates. *Comput. Statist. 20*, 349–368.

Klemelä, J. (2006). Visualization of multivariate density estimates with shape trees. *J. Comput. Graph. Statist. 15*(2), 372–397.

Klemelä, J. (2009). *Smoothing of Multivariate Data: Density Estimation and Visualization*. New York: Wiley.

Korostelev, A. P. and A. B. Tsybakov (1993). *Minimax Theory of Image Reconstruction*, Volume 82 of *Lecture Notes in Statistics*. Berlin: Springer.

Kpotufe, S. and U. von Luxburg (2011). Pruning nearest neighbor cluster trees. In *Proceedings of the 28th International Conference on Machine Learning*, Volume 105, pp. 225–232.

Lo, K., R. R. Brinkman, and R. Gottardo (2008). Automated gating of flow cytometry data via robust model-based clustering. *Cytometry. Part A : J. Int. Soc. Analytical Cytology 73*, 321–332.

Maier, M., M. Hein, and U. von Luxburg (2009). Optimal construction of k-nearest-neighbor graphs for identifying noisy clusters. *Theoretical Computer Science 410*(19), 1749–1764.

Mammen, E. and A. B. Tsybakov (1995). Asymptotical minimax recovery of sets with smooth boundaries. *Ann. Statist. 23*, 502–524.

McMullen, P. (1970). The maximum numbers of faces of a convex polytope. *Mathematika 17*, 179–184.

Melamed, M. R., T. Lindmo, and M. L. Mendelsohn (1990). *Flow Cytometry and Sorting* (2nd ed.). New York: Wiley.

Menardi, G. and A. Azzalini (2014). An advacement in clustering via nonparametric density estimation. *Stat. Comput. 24*, 753–767.

Morgan, J. N. and J. A. Sonquist (1963). Problems in the analysis of survey data, and a proposal. *J. Am. Statist. Assoc. 58*, 415–434.

Müller, D. W. and G. Sawitzki (1991). Excess mass estimates and tests of multimodality. *J. Am. Statist. Assoc. 86*, 738–746.

Naumann, U., G. Luta, and M. P. Wand (2010). The curvHDR method for gating flow cytometry samples. *BMC Bioinformatics 11*(44).

Nolan, D. (1991). The excess-mass ellipsoid. *J. Multivariate Anal. 39*, 348–371.

O'Neill, K., N. Aghaeepour, J. Spidlen, and R. Brinkman (2013). Flow cytometry bioinformatics. *PLOS Computational Biology 9*(12).

Ooi, H. (2002). Density visualization and mode hunting using trees. *J. Comput. Graph. Statist. 11*, 328–347.

Polonik, W. (1995). Measuring mass concentration and estimating density contour clusters - an excess mass approach. *Ann. Statist. 23*, 855–881.

Reeb, G. (1946). Sur les points singuliers d'une forme de pfaff completement integrable ou d'une fonction numerique. *Comptes Rend. Acad. Sci. Paris 222*, 847–849.

Rigollet, P. and R. Vert (2009). Optimal rates for plug-in estimators of density level sets. *Bernoulli 15*, 1154–1178.

Rinaldo, A. and L. Wasserman (2010). Generalized density clustering. *Ann. Statist. 38*, 2678–2722.

Scott, D. W. (1992). *Multivariate Density Estimation: Theory, Practice, and Visualization*. New York: Wiley.

Shapiro, H. M. (2003). *Practical Flow Cytometry* (4th ed.). New York: Wiley.

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. London: Chapman and Hall.

Singh, A., C. Scott, and R. Nowak (2009). Adaptive Hausdorff estimation of density level sets. *Ann. Statist. 37*, 2760–2782.

Stuetzle, W. (2003). Estimating the cluster tree of a density by analyzing the minimal spanning tree of a sample. *J. Classification 20*(5), 25–47.

Stuetzle, W. and R. Nugent (2010). A generalized single linkage method for estimating the cluster tree of a density. *J. Comput. Graph. Statist. 19*, 397–418.

Tarjan, R. E. (1976). Efficiency of a good but not linear set union algorithm. *J. ACM 22*, 215–225.

Tsybakov, A. B. (1997). On nonparametric estimation of density level sets. *Ann. Statist. 25*, 948–969.

Walther, G. (1997). Granulometric smoothing. *Ann. Statist. 25*, 2273–2299.

Walther, G., N. Zimmerman, W. Moore, D. Parks, S. Meehan, I. Belitskaya, J. Pan, and L. Herzenberg (2009). Automatic clustering of flow cytometry data with density-based merging. *Advances in Bioinformatics 2009*, 686–759.

Zomorodian, A. (2012). Topological data analysis. In A. Zomorodian (Ed.), *Advances in Applied and Computational Topology*, Volume 70, pp. 1–40. American Mathematical Society.