Research Project (PRe)

Major : STIC

Academic year : 2020

Fast estimation of convex density contours

Confidential Notice

Non Confidential Report

Author:
Marwen Bahri

Promotion:
2019-2022

Ensta Paris' Tutor:
Mr. Gianni Franchi

Télécom Paris' Tutor:
Mr. Pavlo Mozharovskyi

internship from 17/05/2021 to 13/08/2021

At: Télécom Paris

Address: 19 Place Marguerite Perey, 91120 Palaiseau

Abstract

A depth function can characterize a family of probability distributions if its central regions coincide with the density level contours of that family. We want to develop the correspondance between the depth function and the probability density for the family of quasi-concave probability distributions. Regardless of the depth function we may propose, we have to have a good estimate of the density level contours in order to be able to measure its performance. The exisiting procedures in the literature for the estimation of convex density contours are either not generalizable for high dimensions, or they are only efficient for a specific family of densities other than quasi-concave densities. We propose a new algorithm for the fast estimaton of convex density contours for the family of quasi-concave distributions. The designed algorithm relies on the geometry of the points and avoids operations that are non-scalable for the dimension d. We go through the different approaches we tested to better the procedure. We manage to make good estimations of the convex density contours and we address a possible path for future improvements.

Keywords: *data depth; convex density contours; quasi-concavity; central regions; convex hulls; estimation; optimization; linear programming*

Résumé

Une fonction de profondeur peut caractériser une famille de distributions de probabilités si ses régions centrales coïncident avec les contours du niveau de densité de cette famille. Nous voulons développer la correspondance entre la fonction de profondeur et la densité de probabilité pour la famille des distributions de probabilité quasi-concaves. Indépendamment de la fonction de profondeur que nous pouvons proposer, nous devons avoir une bonne estimation des contours de niveau de densité afin de pouvoir mesurer sa performance. Les procédures existantes dans la littérature pour l'estimation des contours de densité convexes sont soit non généralisables pour les dimensions élevées, ou elles ne sont efficaces que pour une famille spécifique de densités autres que les densités quasi-concaves. On propose un nouvel algorithme pour l'estimation rapide des contours de densité convexes pour la famille des distributions quasi-concaves. L'algorithme conçu s'appuie sur la géométrie des points et évite les opérations qui ne sont pas extensibles pour la dimension d. Nous passons par les différentes approches que nous avons testées pour améliorer la procédure. Nous parvenons à faire de bonnes estimations des contours de densité convexe et nous abordons une voie possible pour les améliorations futures.

mots clés: *profondeur des données; contours de densité convexes; quasi-concavité; régions de profondeurs; enveloppes convexes; estimation; optimisation; programmation linéaire*

Acknowledgments

I would like to express my deep and sincere thanks of gratitude to my research supervisor, Mr.Pavlo Mozharovskyi, for all his guidance and support throughout the internship, which have made this research project an inspiring and enriching experience for me. I would also like to thank Mr.Franchi Gianni, my ENSTA Paris's referring professor, for his helpfulness and assistance. Finally, I would like to thank all my friends and colleagues who have supported me throughout the internship.

Contents

List of Figures

Introduction

The lack of a natural definition of order for multivariate data has been the motive behind the development of many statistical tools that generalize univariate order notions such as medians, extremes and quantiles to the higher dimensions. For instance, a generalized definition for the median of a multivariate data set is the Geometric Median which is based on absolute distances between points. Another generalized definition is the Marjinal median which is defined as a vector whose components are univariate medians. In 1975, Tukey proposed a new defintion for the multivariate median. He proposed that the *deepest* point in a given data cloud represents the median. This was the first time the concept of *depth* has been used, and since then, a whole statistical methodology have been built around the notion of data depth.

Data depth provides a variety of non-parametric depth statistics that are useful in many applications such as nonparametric description of multivariate distributions, depth-based multivariate classification and clustering, and many other ones. The depth of a point relative to a data cloud is measured by a depth function. a depth function is built on a set of postulates that may be found to be slightly different from one paper to another. In this report we use the definition of data depth that is given in Mosler and Mozharovskyi (2020) [1], which is composed of 4 restrictions that we will explore in chapter 1. Usually a depth function is restricted to be affine invariant. This restriction is useful especially when the underlying probability distribution is *elliptical*. In this case, all affine invariant depth functions are equivalent and they have parallel upper level sets of elliptical shape. If in addition the distribution is unimodal, then the density level sets have the same elliptical shape and the density is a transformation of the depth (see Mosler 2013 [3]). These properties are very useful in many statistical applications in which we need the depth to be representative of the underlying probability distribution.

When it comes to the class of *quasi-concave* probability distributions, which represents a wider range of probabilities than the class of elliptical distributions, most of the depth functions that are available in the literature perform poorly in the characterization of the density. We want to extend the correspondance between depth and density to the family of quasi-concave densities by proposing a new depth function that can characterize this family of distributions, knowing that a depth function can characterize a family of distributions only if its central regions coincide with the density level contours of this family. Therefore, independantly of the definition of the depth function, we will need to have a good estimate of the density contour in order to be able to measure the

performance of the depth. But the estimation of the density level contours is in itself a hard task, especially for high dimensions $d \geq 2$ where most of the existing procedures for the estimation of density contours are inefficient. In this internship, we explore a new procedure for the fast estimation of convex density contours which relies on the geometry of the sample points and avoids operations that are non-scalable for the dimension d. The procedure is based on an estimation-re-estimation approach inspired by the Fast Minimum Covariance Determinant algorithm [2]. We estimate the convex density contour by the convex hull of the minimum volume set containing a certain proportion p of the sample points, i.e we search for a set $\mathcal{Y}$ such that $\mathcal{Y} = argmin_X(CH(X))$ such that X contains exactly m sample points, with $m = p \times n,\ p < 1$. Our idea to estimate such a set is the following:

> Construct an initial convex set containing m of sample points, we compute the convex hull of this set and then we scale this convex hull in a specific way that moves it in the direction of high density of sample point. We ensure this by including the mean of the working set in the scaling part. The idea is that the mean will guide us in the direction of higher density. We ensure that the new resulting convex set after the scaling contains the same number of points m of data points. Afterwards, we repeat this process on the newly obtained set until we meet a certain convergence criterion.

In chapter 4 we further detail and explain the algorithm. The major step in each iteration of the procedure is the scaling of the convex hull because each time we scale it we have to recalculate the number of points that are inside of it. Both the computation of the convex hull and the calculation of the number of points inside of it has been done by solving the linear programming problem described in 3.1.1. The usage of a linear programming approach is extremely useful as it allows us to do these computations for high dimensions d. The procedure was implemented in the programming language $c++$ in order to gain speed, and interfaced with the script language R for usability conveniance.

In chapter 1 we present some preliminary statistical tools. In chapter 2, we explore the different procedures exisiting in the literature that are implied in the estimation of convex density contours and the calculation of convex hulls. Before presenting the new algorithm, we develop the prerequestis in chapter 3 in order to understand each part of the procedure. Chapter 4 explains the algorithm as well as presents some numerical results that we obtained and the different approaches we tried to better the results.

Chapter 1

Preliminary statistical tools

1.1 Data depth

1.1.1 Depth function

A depth function $\mathcal{D}$ represents the depth of a point w.r.t a given data cloud or a distribution P from a class of distributionss $\mathcal{P}$ by a number between 0 and 1. Many different depth functions have been proposed in the literature, each of them satisfies the restrictions below and/or other special ones. Note that depth functions have been first proposed for empirical distributions only but many depth functions have their population versions. The following restrictions are given in Mosler and Mozharovskyi (2020) [1]. For $y \in \mathbb{R}^d$, X a random variable distributed as P, we write $D(y|X)$ in place of $D(y|P)$. A depth function satisfies:

- $\mathcal{T}-$**Invariance:** $D(T(y)|T(X)) = D(y|X)$ for all $T \in \mathcal{T}$ a class of $\mathbb{R}^d-transformations$
- **Null at infinity:** $\lim_{||y||\to\inf} D(y|X) = 0$
- **Monotone on rays:** if a point y^* has maximal depth, i.e $D(y^*|X) = max_{y\in\mathbb{R}^d} D(y|X)$, then for any $\mathbf{r}$ in the unit sphere of $\mathbb{R}^d$, the function $\alpha \to D(z^*+\alpha r|X)$ decreases with $\alpha > 0$
- **Upper semicontinuous:** The upper level sets $D_\alpha(X) = \{z \in E : D(z|X) > \alpha\}$ are closed for all $\alpha \in [0, 1]$

One special property that we need to mention is the quasi-concavity property. A depth function is called *quasi-concave* if its upper level sets are convex. An example of a quasi-concave depth function is the *Onion Depth*.

1.1.2 Central regions

Upper level sets D_α, or central regions, are regions with depth greater or equal to a certain threshold α. For a depth function D and $\alpha \in [0, 1]$, central regions are

defined by $D_\alpha(X) = \{z \in E : D(z|X) > \alpha\}$. They describe the ditribution with respect to location, dispersion and shape (see Mosler 2013 [3]). Central regions also provide multivariate ranks and quantiles and they are used in many applications such as description and comparison of multivariate distributions, multivariate classification and clustering, outlier detection, and many other applications. The following figure shows an example of central regions of bivariate macroeconomic data of the European Union countries, produced by the *convex hull peeling depth*. The figure was taken from Mosler and Mozharovskyi (2020) [1].

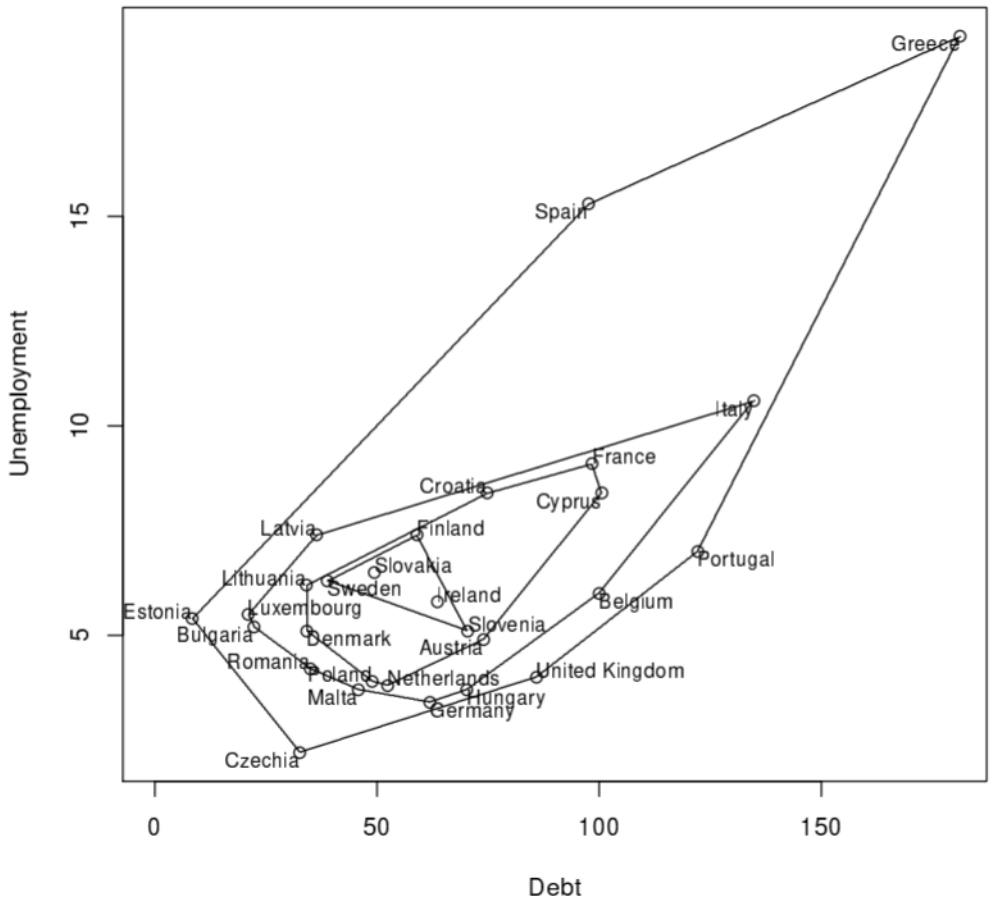

Figure 1.1: central regions of EU data on unemployment and public dept in 2018 (Source: EUROSTAT) for the Onion depth

Source: Choosing among notions of multivariate depth statistics, Mosler and Mozharovskyi (2020)

1.1.3 Convex hull peeling depth (Onion depth)

For an empirical distribution of X The Onion depth formes a nested set of convex layers by repeatedly contructing the convex hull of the data points then removing the points that belong to the peremiter of the covex layer and then repeating the same process

on the remaining sample points. the convex hull peeling depth of a point $x \in \mathbb{R}^d$ relative to d-variate data cloud (convex hull peeling depth is only defined for empirical distributions) is the level of the convex layer that it belongs to. The following definition has been taken from Mosler and Mozharovskyi (2020) [1]. For an empirically distributed X, consider the sequence of closed convex sets:

$$C_1(X) = CH(S_X), \quad C_{j+1}(X) = CH(S_X \cap intC_j(X)), \quad j = 1, 2... \tag{1.1}$$

where $CH(S)$ is convex hull of S, $int(S)$ is the interior of S, and S_X is the finite set on which X is defined. The Onion depth of y is

$$D_{Onion}(y|X) = \sum_{j \geq 1} 1_{intC_j(X)}(y), \quad \textit{with } 1_S \textit{ denoting the indicator function of a set } S \tag{1.2}$$

A study of the properties such as robustness and computational feasibility of the Onion depth can be found in Mosler and Mozharovskyi (2020) [1].

1.2 Density estimation

Based on observed data we want to build an estimation of the underlying density distribution. There are two main approaches for this, one is *parametric* estimation and the other is *non parametric* estimation. The former assumes that the underlying distribution comes from a specific family of distributions, for example Normal distributions, and then tries to estimate the parameters of that distribution, the latter does not make any assumptions on the family of the underlying distribution. In this internship we used a non parametric approach to estimate the density, called *Kernel* density estimation. In this section we provide a brief definition of *Kernel* estimation and in section 5 of chapter 4 we will explain exactly how we used it in the algorithm.

A *Kernel* is a non-negative real-valued integrable function K that satisfies the following property:

$$\int_{-\infty}^{+\infty} K(x)dx = 1 \tag{1.3}$$

Let $(X_1, ..., X_n)$ be a sample of n observations. The *kernel estimator* with kernel K is defined by:

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{x - X_i}{h}), \text{ where } h \text{ is the bandwidth} \tag{1.4}$$

The kernel estimator places a 'bump' over every observation and then adds them up, the width of the bumps is determined by the parameter h. In this intership we use the *Gaussian* Kernel defined by

$$G(x, h) = \frac{1}{(\sqrt{2\pi}h)^d} \exp -\frac{||x||^2}{2h^2} \tag{1.5}$$

A good explanation of kernel estimation can be found in B.W. Silverman (1986) [4]. The following figure is taken from there, and it showcases the effect of the choice of the parameter h on the final estimation.

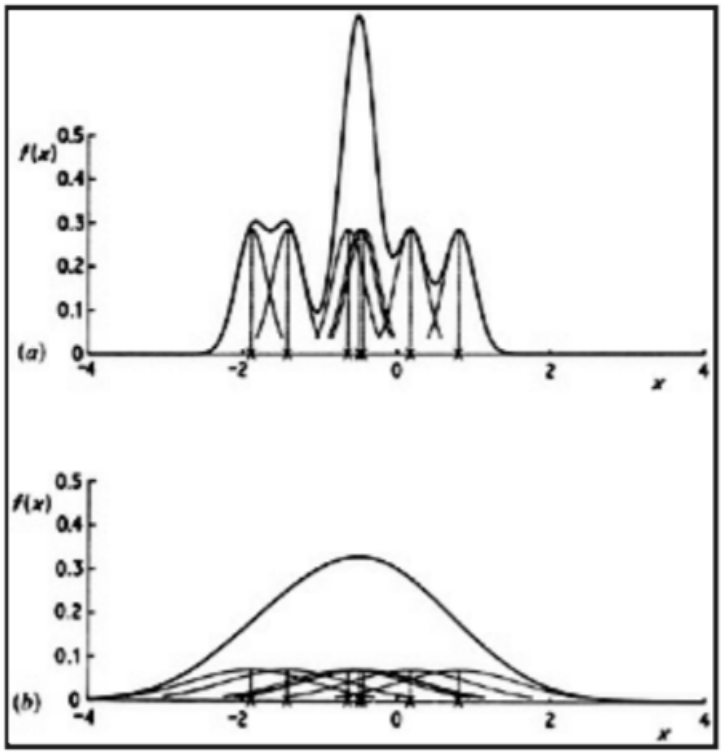

Figure 1.2: Kernel estimates showing individual kernels.
(a) h=0.2; (b) h=0.8

Source: Density estimation for statistics and data analysis, B.W Silvermann

Chapter 2

Exploration of existing procedures

In this chapter we will explore some of the exisiting procedures in the literature for the estimation of convex density contours and minimum volume sets.

2.1 Estimation of a convex density contour by Hartigan

In 1987, Hartigan [5] proposed an algorithm for the estimation of convex density contours for bivariate data clouds only. It is not generalizable to higher dimensions. Hartigan's algorithm estimates the convex density contour containing probability α by contructing a closed convex set S_n containing a proportion α of sample points and that maximizes a certain function, the boundary of S_n is shown to be a consistent estimate of the convex density contour. The time complexity of Hartigan's procedure is $\mathcal{O}(n^3)$. The consistency of the estimation makes the algorithm suited for benchmarking purposes, we can compare our method's estimations to Hartigan's procedure results for bivariate sample data cloud for which we can't compute the exact density level contour.

2.2 Maximum likelihood estimation of a multi-dimensional log-concave density

The *log-concave* maximum likelihood estimator is a fully automatic non-parametric estimator, with no tuning parameter to be chosen unlike the case of the Kernel density estimator. It frees us from the hardships of choosing an appropriate bandwidth for kernel density estimation, which becomes a difficult task when working with high dimensional data. The log-concave maximum likelihood estimator is defined to be the log-concave density function that maximizes the likelihood function defined for a family of $i.i.d$ random vectors $X_1, ..., X_n$ by

$$L(f) = \Pi_{i=1}^{n} f(X_i) \tag{2.1}$$

Cule, Samworth and Stewart (2010) [8] showed that such an estimator is unique and exists with probability 1. The maximum likelihood log-concave estimator is useful for many tasks. It can be used for visualization purposes, for classification purposes in problems where we want to classify an observation as coming from one of multiple populations, in clustering problems, for the estimation of a functional of the true underlying density, as well as many other applications. A study of the theoretical properties of the log-concave maximum likelihood estimator has been conducted in Cule and Samworth (2010) [8]. Cule, Samworth and Steward (2010) [8] proposed an efficient iterative algorithm for the computation of the maximum likelihood log-concave density estimator, and they proved its advantages when working with log-concave densities compared to the Kernel estimator. They conducted a study of the performance of the algorithm on 6 different densities for the cases of $d = 2$ and $d = 3$ and for a variety of sample sizes ($n = 100, 200, 500, 1000,\ and 2000$). The following table is taken from there work and it summarizes the different properties of the 6 used densities (a)-(f).

Density	*Log-concave*	*Dependent*	*Normal*	*Mixture*	*Skewed*	*Bounded*
(a)	Yes	No	Yes	No	No	No
(b)	Yes	Yes	Yes	No	No	No
(c)	Yes	No	No	No	Yes	Yes
(d)	Yes	No	Yes	Yes	No	No
(e)	Yes	No	Yes	Yes	No	No
(f)	No	No	Yes	Yes	No	No

Table 2.1: Summary of features of the example densities

Source: Maximum likelihood estimation of a multi-dimentional log-concave density. Cule, Samworth and Stewart (2010) [8]

The simulation study has proven that for the 5 examples where the density is *log-concave*, the maximum likelihood log-concave density estimator has a smaller *mean iterative square error* than the Kernel estimator for moderate and large sample size n, and ragardless of the bandwidth used for the Kernel estimator (even when the theoretical optimal bandwith is used). However, when the density is not log-concave (case (f)) the Kernel estimator outperforms the maximum likelihood estimator.

2.3 Minimum volume peeling estimator

The *Minimum Volume Peeling* algorithm was proposed by T.Kirschtein, S.Liebscher, G.C. Porzio, G.Ragozini [6] as an efficient way to compute the minimum volume subset for Sager's procedure for the estimation of the multivariate mode. The main idea of the Minimum volume peeling algorithm is to progressively build the minimum volume set by succesively adding a number $m1$ of the closest data points to the working set's convex hull and then removing the $m2$ data points incident to the longest facet of the

new convex hull, having $m_1 > m_2$. This process of adding and removing data points is repeated until the set contains exactly a number n_k of points. The procedure applies the process to l different intial subsets of size $d+1$ (d is the dimension) and then picks the final set with the minimum volume. The running time of the procedure depends heavily on the choice of m_1 and m_2.

Chapter 3

Prerequesits

In order to understand what we are doing in the algorithm, this chapter will introduce a set of different notions and procedures that we used in the algorithm.

3.1 Linear optimization

Linear optimization, often called linear programming, is an optimization method used to find optimal solution of a minimization or a maximization problem defined by *linear* relationships and subjected to linear constraints. A linear programming problem is expressed in the following standard form:

$$\begin{aligned} &Find\ a\ vector\ x \in \mathbb{R}^n \\ &that\ miminizes\ c^T x \\ &subject\ to\ Ax \geq\ B \\ &and\ x \geq \mathbb{0}_d \end{aligned} \tag{3.1}$$

$c^T x$ is called the objective. Each minimization problem has a maximization dual problem and vice versa. The most common algorithm for solving linear programming problems is the *Simplex* algorithm.

3.1.1 Computation of the convex hull

The computation of the convex hull plays a major part in our procedure. There are various algorithm proposed in the literature for the computation of the convex hull with various computational complexities, with most of them being dedicated for bivariate points only. In order to increase the computational feasibility of the computation of the convex hull for higer dimensions, we formulate the convex hull problem into a linear programming problem which is solvable for dimensions reaching up to $d = 20$. Suppose we want to find the convex hull $CH(\mathcal{Y})$ of a set of m points in $\mathbb{R}^d$ represented by the set $\mathcal{Y}$. A point belonging to $CH(\mathcal{Y})$ is called an extreme point of $\mathcal{Y}$. We use the fact

that an extreme point of $\mathcal{Y}$ is not a convex combination of points of $\mathcal{Y}$ to formulate the following linear programming problem:

$$\begin{aligned} &for\ each\ y_i \in \mathcal{Y}, \\ &find\ max\ \lambda_1\ s.t: \\ &\mathcal{Y}_{-i}^T \Lambda = y_i, \\ &\Lambda^T \mathbb{1}_{m-1} = 1, \\ &\Lambda = \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_{m-1} \end{pmatrix} \geq \mathbb{0}_{m-1} \end{aligned} \tag{3.2}$$

$\mathcal{Y}_{-i}$ is a $(m-1) \times d$ matrix formed by the elements of $\mathcal{Y}$ excluding y_i. If no solution exists for a given point y_i, then this point is an extreme point of CH, i.e $y_i \in CH(\mathcal{Y})$. Note that the objective $max\ \lambda_1$ is arbitrary because we only need to know if a solution exists or not and the value of this solution doesn't matter. In our procedure we used the *No license linear programming kit* developped by Pavlo Mozharovksyi to solve the previous linear programming problem using the *Simplex* algorithm.

3.2 The Quickhull algorithm by Barber

The Quickhull algorithm proposed by Barber in 1996 [7] is a combination between the two-dimensional Quickhull algorithm and the general-dimension Beneath-Beyond Algorithm, it efficiently computes the convex hull of general-dimension data points. Its complexity is of the order of $\mathcal{O}(n \log(n))$ for $d \leq 3$, with n is the number of data points for which we are computing the convex hull, and of the order of $\mathcal{O}(n\frac{f_r}{r})$ for $d \geq 4$, with r being the number of processed points and f_r the maximum number of facets for r vertices. Note f_r increases exponentially with the dimension d and it can take really high values for high values of d. The Quickhull algorithm has been implemented in c programming language and made available in a program called *qhull*. The program computes convex hulls, Delaunay triangulations, Voronoi vertices, further-site Voronoir vertices, and halfspace intersections. It also provides an option for the estimation of the volume of the convex hull, which, for high values of d, becomes a computationally expensive and inefficient task. We used this option in our method for benchamarking purposes, where the tests have been conducted on bivariate data clouds only. we kept track of the evolution of the volume of the convex hulls between the iterations of the procedure inorder to analyse the variation of the volume as well as to compare our final estimation's volume to the volume of the real convex density contour.

Chapter 4

Fast estimation of convex density contours algorithm

4.0.1 Explanation of the algorithm

Notations:
$\mathcal{X}$: *set of n points in* $\mathbb{R}^d$,
n : *number of data points*,
d : *the dimenion of the data points*,
p : *the proportion of data to be inside CH*,
m : $m < n$ *points to be inside the central region*,
$\mathcal{Y}$: *set of m points inside of the current convex hull*,
CH : *convex hull of* $\mathcal{Y}$,
$\mathcal{V}$: *vertices of CH*,
I : *size of V*,
$\mathbf{c}_0$: *the mean of points inside the previous convex hull*,
$\mathbf{c}_n$: *the mean of points inside the current convex hull*,
λ : *parameter used to scale the convex hull*

Our procedure is based on an estimation-re-estimation approach, so we need an initial estimate of the central region. We use the convex hull of the m closest points to the center c_0 of the total set of points $\mathcal{X}$ as an initial estimate of the central region. We represent the data points inside this initial estimate by the set $\mathcal{Y}$ and the vertices of its convex hull by the set $\mathcal{V}$. In the next step we will change the shape of the convex hull of the set $\mathcal{Y}$ in order to move in the direction of high densities. We calculate the mean c_n of the set $\mathcal{Y}$ and search for the parameter λ_{opti} such that the convex $\{c_n + \lambda_{opti}(V_1 - c_0), ..., c_n + \lambda_{opti}(V_j - c_0), ..., c_n + \lambda_{opti}(V_I - c_0)\}$ contains exactly m points, with V_i is a vertex of the convex hull CH and I is the total number of vertices of CH. After finding the optimal parameter λ_{opti} we update the sets $\mathcal{Y}$ and $\mathcal{V}$ and the center c_0. We repeat this process until the absolute difference between the centers c_n and c_0 is less then a parameter ϵ. Following is the pseudo-code for the procedure:

convex hull estimation procedure

procedure ConvexHullEstimation($\mathcal{X}, p$)
  $n \leftarrow$ *#rows of* $\mathcal{X}$, $d \leftarrow$ *#columns of* $\mathcal{X}$
  $m \leftarrow ceil(n \times p)$
  $c_0 \leftarrow$ **mean**($\mathcal{X}$)
  $\mathcal{Y} \leftarrow m$ *closest points to* c_0
  $\mathcal{V} \leftarrow$ **chull**($\mathcal{Y}$)
  $c_n \leftarrow$ **mean**($\mathcal{Y}$)
  do:
    $\lambda_{max} \leftarrow \max_{i,j} \left(\frac{||\mathcal{X}_i - c_0||}{||\mathcal{V}_j - c_0||}\right) \times 10$
    $c_n \leftarrow$ **mean**($\mathcal{Y}$)
    $\lambda_{opti} \leftarrow$ **Find**λ $(\mathcal{Y}, \mathcal{V}, c_0, c_n, m, \lambda_{max})$
    $\mathcal{Y} \leftarrow$ **UpdateY** $(\mathcal{Y}, \mathcal{V}, c_0, c_n, \lambda_{opti})$
    $\mathcal{V} \leftarrow$ **chull**($\mathcal{Y}$)
    $c_0 \leftarrow c_n$
    $c_n \leftarrow$ **mean**($\mathcal{Y}$)
  while: $|c_n - c_0| > \epsilon$
  return $(\mathcal{V}, \mathcal{Y}, c_0)$
end procedure

The **chull** method computes the convex hull of a set of points. This method makes use of the open source library *NlLpk* to solve the linear programming problem described in 3.1.1 using the *Simplex* algorithm. The complexity of the **chull** method depends on the complexity of the Simplex algorithm, which in theory can be exponential to the size of the linear system, but in practice it is shown that it doesn't reach exponential running times. Lets denote the complexity of the Simplex algorithm by s. Since in **chull** we solve the linear programming problem for each $y_i \in \mathcal{Y}$, the complexity will be $\mathcal{O}(m \times s \times \mu)$, whith μ being the time complexity for building $\mathcal{Y}_{-i}$ for each y_i. But there is a solution to avoid building $\mathcal{Y}_{-i}$ form zero on each iteration. First, note that in *NlLpk*, the matrix $\mathcal{Y}_{-i}$ of the system 3.1.1 is represented by a vector of length $d \times m$, where the lines of the matrix $\mathcal{Y}_{-i}$ are stacked horizontally, so each succesive d coeffiecients represent the coeffiecient of a point y_i. Now, start by constructing the vector for the first element of $\mathcal{Y}$, keep track of the previous processed point and each time you process a new point you just replace the d coefficients in $\mathcal{Y}$ corresponding to the current element by the coefficients of the previously processed one. This is possible since we can determine the position corresponding to the coefficients for any $y_i \in \mathcal{Y}$ because the elements of $\mathcal{Y}$ are saved in a *c++* vector and we can iterate over them in an ordered manner. By this, the complexity of **chull** becomes $\mathcal{O}(m \times s)$.

The **Find**λ method runs a **binary search** algorithm in order to find the optimal value for the parameter λ. It makes use of the same linear programming approach used in the **chull** function but instead of using it to find the vertices of the convex hull, it uses it to calculate the number of points in its interior.

The **UpdateY** method updates the set $\mathcal{Y}$ by the m points which are inside the convex hull of $\{c_n + \lambda_{opti}(V_1 - c_0), ..., c_n + \lambda_{opti}(V_j - c_0), ..., c_n + \lambda_{opti}(V_I - c_0)\}$. The mean c_n will always be close to the location of high density, so having it in the optimization equation guides us in right direction. But this was not enough to get a good prediction, as after numerical exprimentation we noticed that we manage to move the working set to a high density location but we fail to have the correct shape for *CH*. This led us to think of other ways to do the scaling part of the procedure, ways that may reflect more the underlying density distribution. In the next sections we will go through the different approaches we tested in the scaling part of the procedure and explain the motive behind each one of them as well as present the simulation results we obtained for each approach.

4.1 Scaling

4.1.1 Uniform scaling

Uniform scaling was the first approache we used. In this approache we multiply all the vertices V_i of *CH* by the same constant λ. With this approache we succefuly moved the initial convex hull to a relatively good position, but the shape of *CH* was still not good enough as it shows figure 4.1. The following data cloud is consisted of 5000 data points (n=5000) and generated by a skew-normal distribution using the *R* package ***sn***. The red zone represents the true convex density level set for $p = 0.3$. We ran the procedure with the parameters $p = 0.3$ *and* $\epsilon = 0.01$. The estimation of the convex hull area is computed using the qhull program.

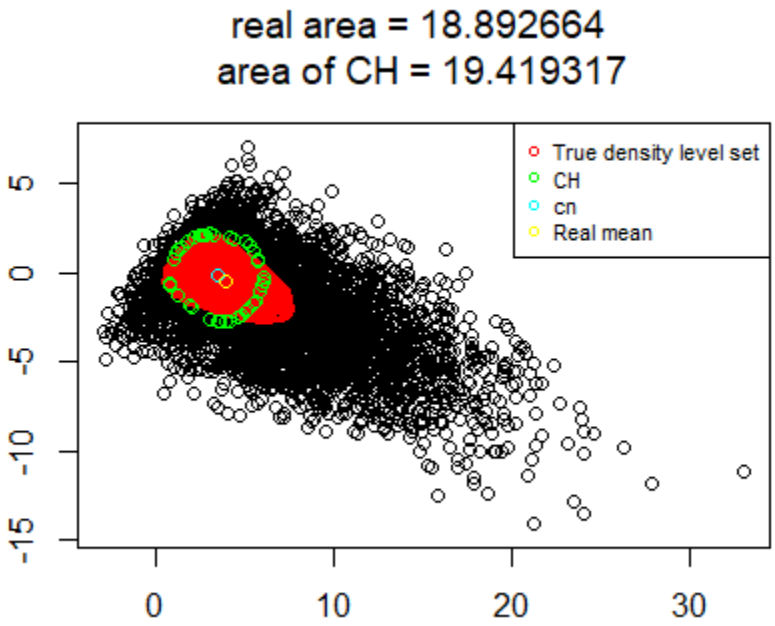

Figure 4.1: Rsults of the uniform approach on skew-normal data cloud
n=5000, p=0.3

We ran the procedure on $n = 500$ sample points produced by the same distribution, we got the following result (we changed the background color for visibilty purposes):

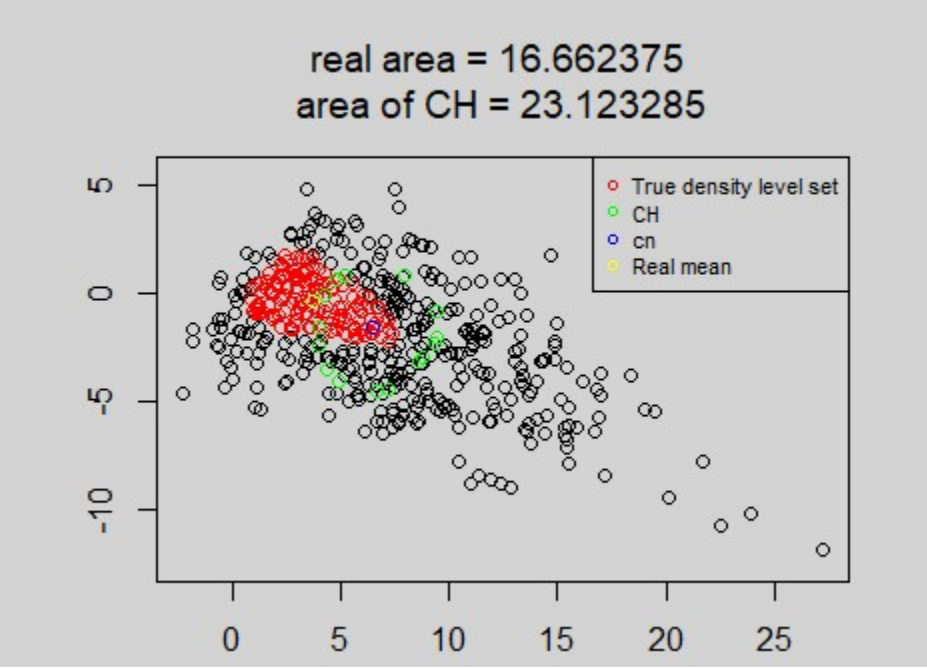

Figure 4.2: Rsults of the uniform approach on skew-normal data cloud
n=500, p=0.3

The estimation quality has detriorated significantly.

4.1.2 Non-uniform scaling

In order to ensure that the shape of the convex hull changes in a way that reflects the underlying density distribution, we need to favor the scaling in the directions of high density instead of uniformly scaling the shape. We tried to do this by adding new parameters P_i to the optimization problem and we thought of a variety of ways of choosing the values for these new parameters. The new optimization problem is find λ such as $\{c_n + \lambda P_1 (V_1 - c_0), ..., c_n + \lambda P_j (V_j - c_0), ..., c_n + \lambda P_I (V_I - c_0)\}$ contains exacly m points.

4.2 Non-uniform scaling based on distances

We start by computing the distances l_i of the vertices V_i, $i \in \{1..I\}$, of CH to the center c_0. Let $\bar{l}$ be the mean of these distances. In this approach we fix the parameter

P_i to be equal to a constant $g > 1$ if the distance l_i of the corresponding vertice V_i is less than $\bar{l}$, i.e $P_i = g,\ if\ l_i < \bar{l}$, else we fix it to a constant $s < 1$.

$$P_i = \begin{cases} g & \text{if } l_i < \bar{l},\ with\ g > 1 \\ s & \text{if } l_i > \bar{l},\ with\ s < 1 \end{cases} \tag{4.1}$$

The computation of the distances l_i adds a time complexity of $\mathcal{O}(I)$ to the procedure. The choice of the parameters g and s has a big impact on the final result. We tuned these parameters manually to see if we can get a good estimation, and if so, we could add another optimization problem to the algorithm where we search for good values for g and s. The following figure shows the results obtained on the same data cloud used in 4.1.1.

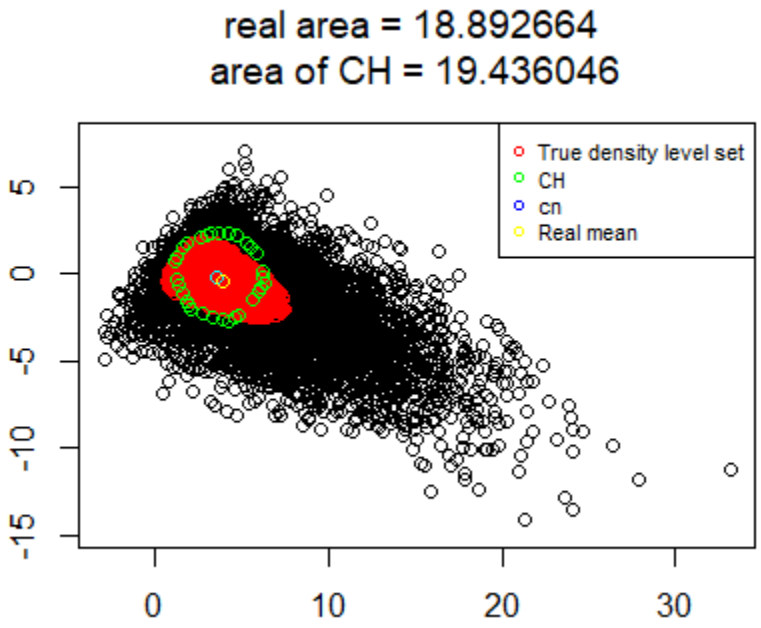

Figure 4.3: Non-uniform scaling based on distances
g=1.07, s=0.9
n=5000, p=0.3

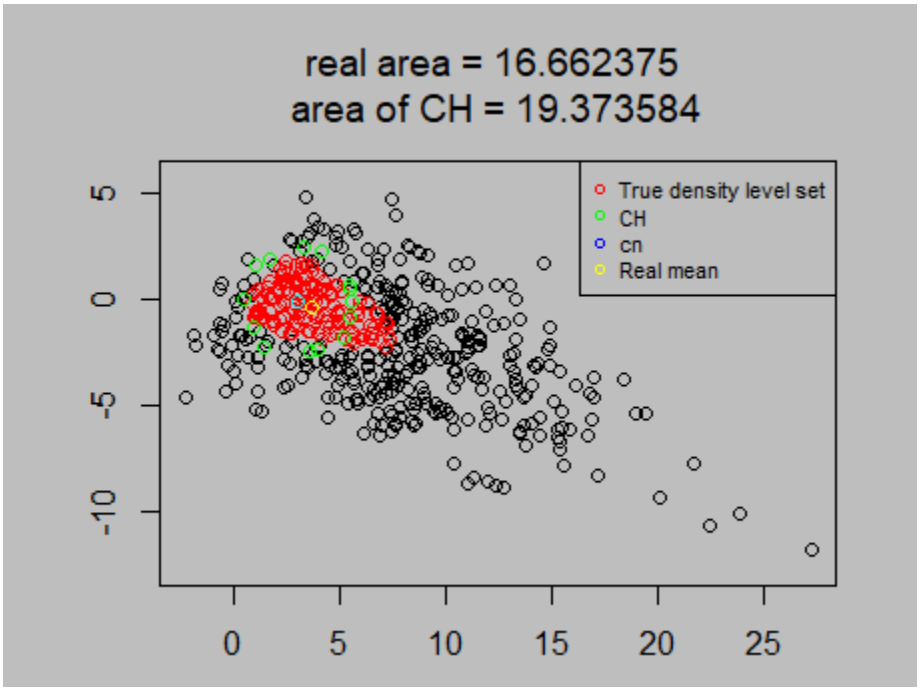

Figure 4.4: Non-uniform scaling based on distances
g=1.07, s=0.9
n=500, p=0.3

As we can see, the estimation did not get much better even after trying multiple values for g and l, except that for the case of $n = 500$ the location of the final estimation is significantly better than for the case of uniform scaling. Another important thing to mention is that for most values of g and l, the procedure did not meet the stoppage criterion, i.e the absolute difference between c_0 and c_n never became lesser than ϵ. This leads us to consider changing the stoppage criterion, but we decided to just fix a maximum number of iterations for the procedure for now and keep the problem of stoppage criterion to possible future improvements.
The next idea is still based on distances, but this time rather than using only two fixed values for g and l, we let P_i take values based on the following equation:

$$P_i = (1 - \gamma \times \frac{l_i - \bar{l}}{\bar{l}}) \tag{4.2}$$

The paramater γ is a constant used as a sensibility parameter. We can see that the results have improved, but they are still not good enough. Since we are working on dimension $d = 2$ and with sample size of $n = 5000$, we should reach a nearly perfect estimation. However, we see a big improvement of the estimation for the case of $n = 500$.

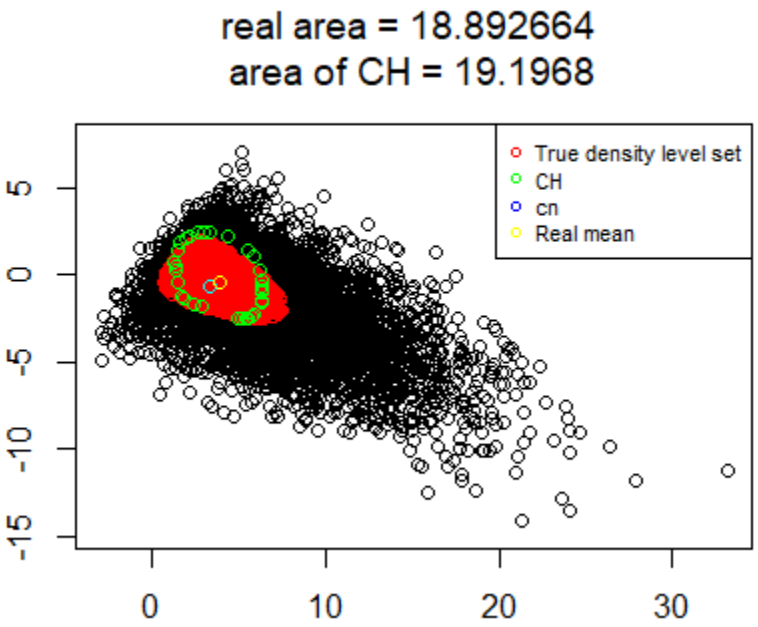

Figure 4.5: Non-uniform scaling based on distances second approach gamma = 2.5, n=5000 and p=0.3

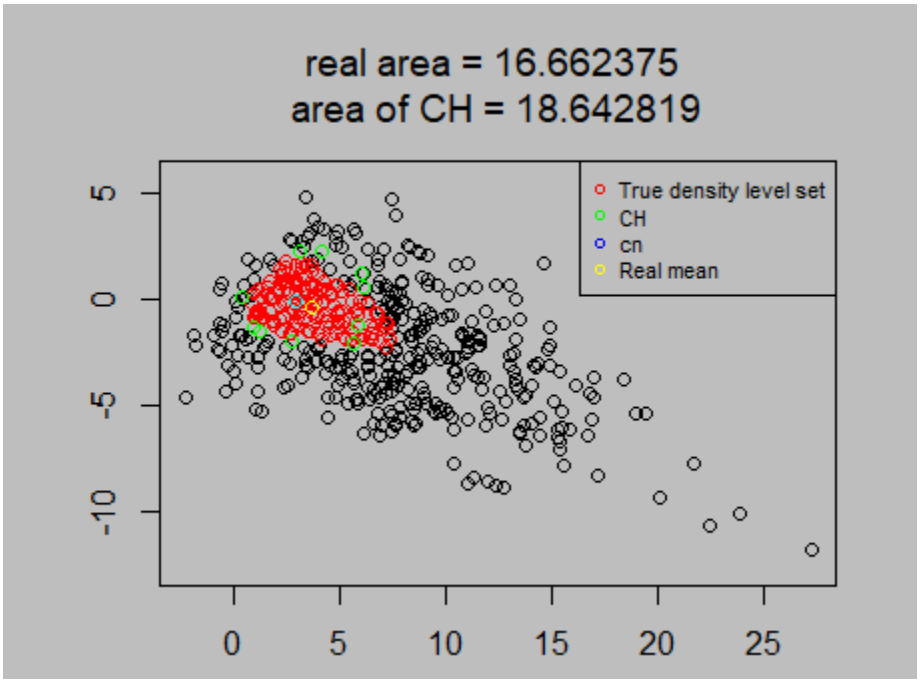

Figure 4.6: Non-uniform scaling based on distances second approach gamma = 2.7, n=500 and p=0.3

4.3 Non-uniform scaling based on standard deviation

The next idea we tried was based on standard deviation. For every vector $(V_i - c_0)$, $V_i \in \mathcal{V}$, we project all sample points inside of $\mathcal{Y}$ on this vector. Using these projection we then calculate the empirical standard deviation

$$s_i = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(p_j - \bar{p}_i)^2} \tag{4.3}$$

with $p_j,\ j \in [1..m]$ being the projections of the points of $\mathcal{Y}$ on $(V_i - c_0)$ and $\bar{p}_i$ is the mean of these projections. After computing the s_i value for each vertex $V_i \in \mathcal{V}$ we calculate the mean standard deviation $\bar{s}$ and use a similar formulation for P_i as in 4.2. The computation of these parameters has a time complexity of $\mathcal{O}(m \times I)$.

$$P_i = (1 - \gamma \times \frac{s_i - \bar{s}}{\bar{s}}) \tag{4.4}$$

At first, we tried values ranging from 0.5 to 4 for the parameter γ. For this range of values we didn't get any noticeably good estimation, so we had to pass the next idea which is based on kernel density estimation. But later on we came back to this approach and tried smaller values for γ ranging from 0.01 to 0.05, and the method actually produced a good estimation of the density level set as we can see in the following figure.

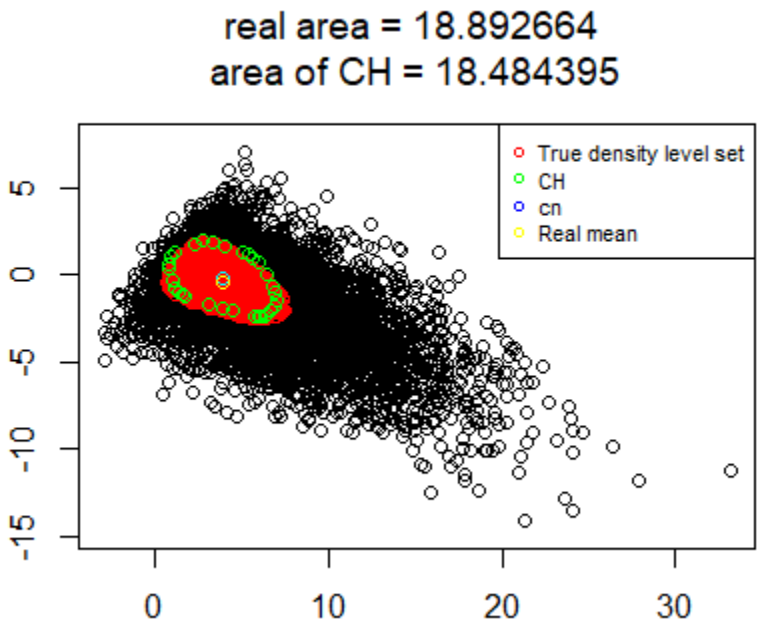

Figure 4.7: Non-uniform scaling based on standard deviation
gamma = 0.05, n=5000 and p=0.3

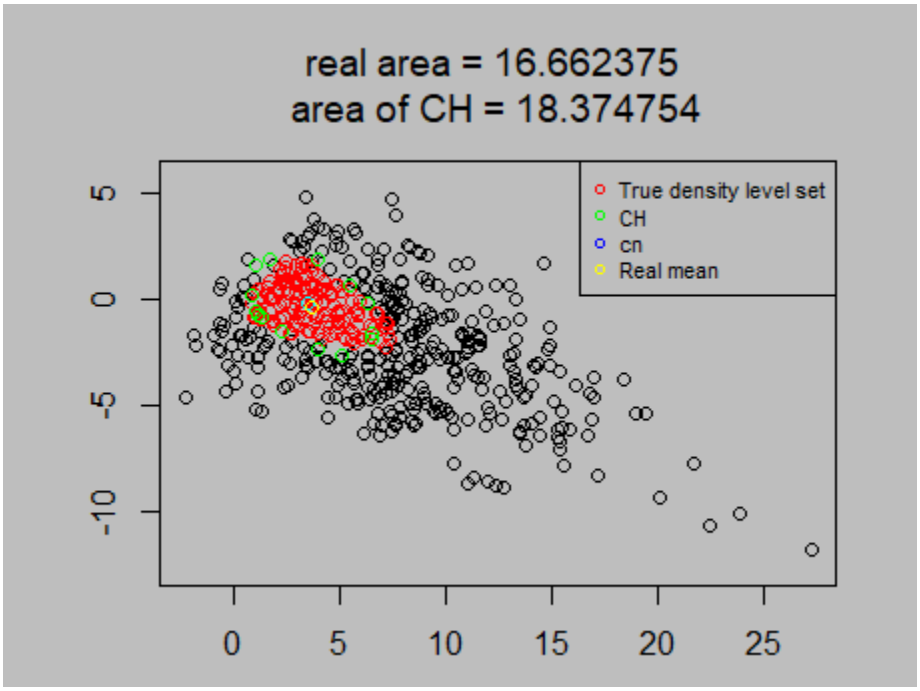

Figure 4.8: Non-uniform scaling based on standard deviation gamma = 0.08, n=500 and p=0.3

For the case of $n = 5000$ the procedure did not meet the stoppage criterion. Instead, it stopped after reaching the maximal number of iterations which is fixed to 100.

4.4 Non-uniform scaling based on kernel denisty estimation

Consider the projections of the points of $\mathcal{Y}$ on the vectors $(V_i - c_0)$ for each vertex $V_i \in \mathcal{V}$. We compute a kernel density estimator for each vertex V_i using these projections as observations in the equation (1.4). Then we can compute the value of the density at each vertex V_i, using the Gaussian function as a kernel. Let $\hat{f}_{V_i}$ be the value of the density estimation on the vertex V_i and $\bar{\hat{f}}$ be the mean of these values. We define the parameters $P_i,\ i = 1, .., I$ as follows:

$$P_i = \begin{cases} g & \text{if } \hat{f}_{V_i} < \bar{\hat{f}},\ with\ g > 1 \\ s & \text{if } \hat{f}_{V_i} > \bar{\hat{f}},\ with\ s < 1 \end{cases} \tag{4.5}$$

The computation of these estimations will add a time complexity of $\mathcal{O}(m \times I)$. The following results were obtained with $g = 1.05$, $s = 0.95$ and a bandwidth $h = 0.3$.

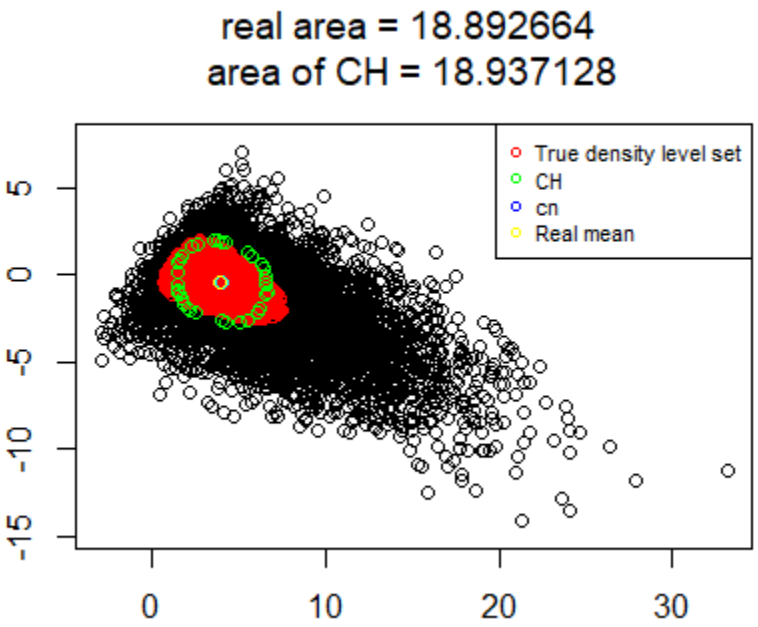

Figure 4.9: Density estimation approach - first version
g=1.09, s=0.95 and h=0.3
n=5000, p=0.3

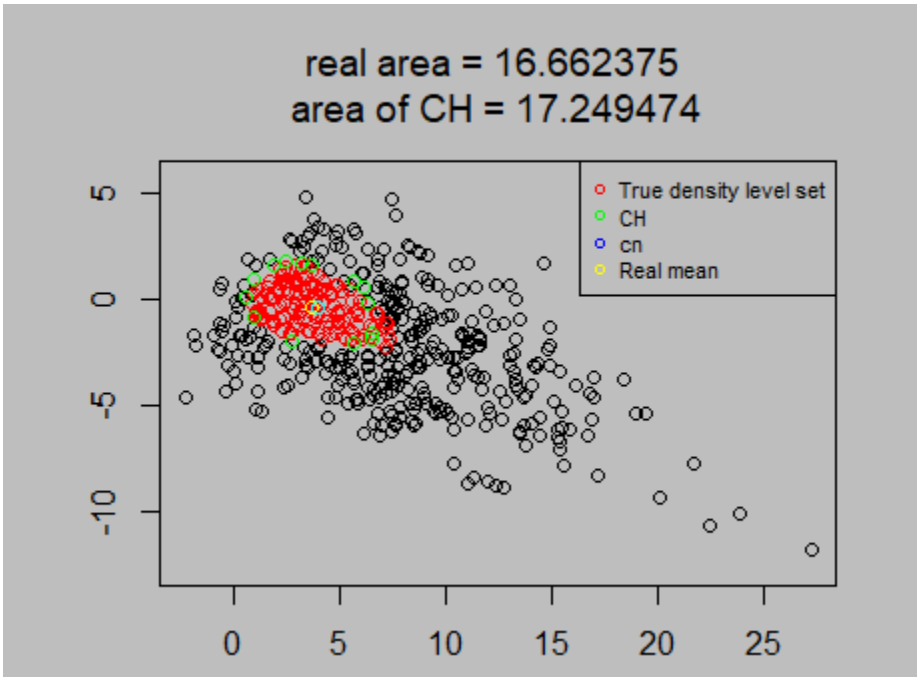

Figure 4.10: Density estimation approach - first version
g=1.08, s=0.8 and h=0.3
n=500, p=0.3

We then tried limiting the points used in the density estimation for each vertex to only the data points inside a cylinder of a certain ray around the vector $(V_i - c_0)$. We used the distance between the vector $(V_i - c_0)$ to the mean c_n of $\mathcal{Y}$ as the ray length, and we obtained the following results with the parameters $g = 1.05$, $s = 0.95$ and $h = 0.3$. Note that the procedure did not converge after 100 iterations.

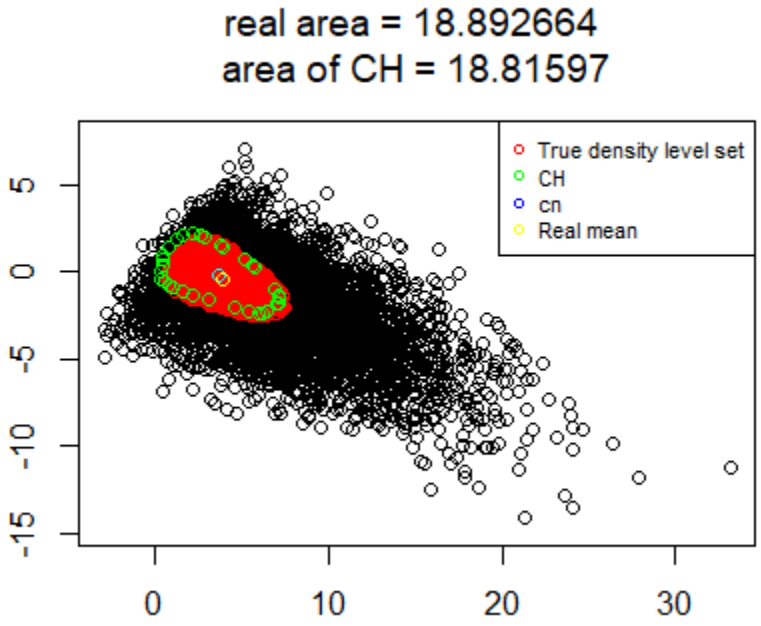

Figure 4.11: Density estimation approach - second version
g=1.05, s=0.95 and h=0.3
n=5000, p=0.3

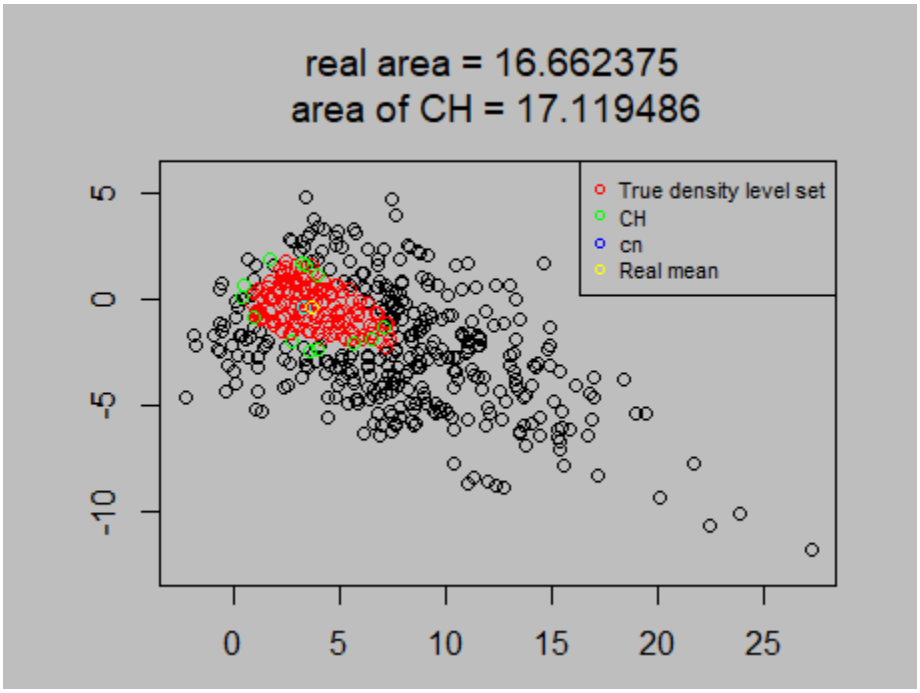

Figure 4.12: Density estimation approach - second version
g=1.08, s=0.89 and h=0.3
n=500, p=0.3

The estimation's quality has clearly improved using this method. It might be possible to get even better results if we continue to tune the parameters g and s, but obviously, we will have to find a way to automatically or systematically select these values in order to have a practical and publishable procedure.

Conclusion

In this report we proposed a new procedure for fast estimation of convex density contours for quasi-concave probability distributions. The results obtained by the kernel density estimation and by the standard deviation approaches prove that the method we proposed can produce consistent estimations if we choose appriopriate values for the scaling parameters. This opens the way for future improvements of the procedure by focusing on developping an efficient strategy for the selection of these parameters. It may be possible to find an optimization problem to determine optimal values for these parameters. Such an optimization problem must not disadvantage the scalability of the procedure. An other part of the procedure that is open to improvements is the convergence criterion, since for most cases with $n = 5000$ the procedure did not converge.

Bibliography

[1] Karl Mosler and Pavo Mozharovskyi. (2020). *Choosing among notions of multivariate depth statistics.*

[2] Rousseeuw, P.J and Van Driessen, K. (1999) *A fast algorithm for the minimum covariance determinant estimator.* Technometrics, 41,212-2013

[3] Karl Mosler. (2013). *Depth statistics.* 5

[4] B.W Silverman. (1986). *Density estimation for statistics and data analysis.* 9-13

[5] J. A. Hartigan. (1987). *Estimation of a convex density contour in two dimensions.*

[6] T. Kirschtein, S. Liebscher, G.C Porzio and G. Ragozini. (2016) *Minimum volume peeling: A robust nonparametric estimator of the multivariate mode.* 459-462

[7] C. Bradford Barber. (1996). *the Quickhull algorithm for convex hulls.*

[8] Madeleine Cule, Richard Samworth and Michael Stewart. (2010). *Maximum likelihood estimation of a multi-dimensional log-concave density.*