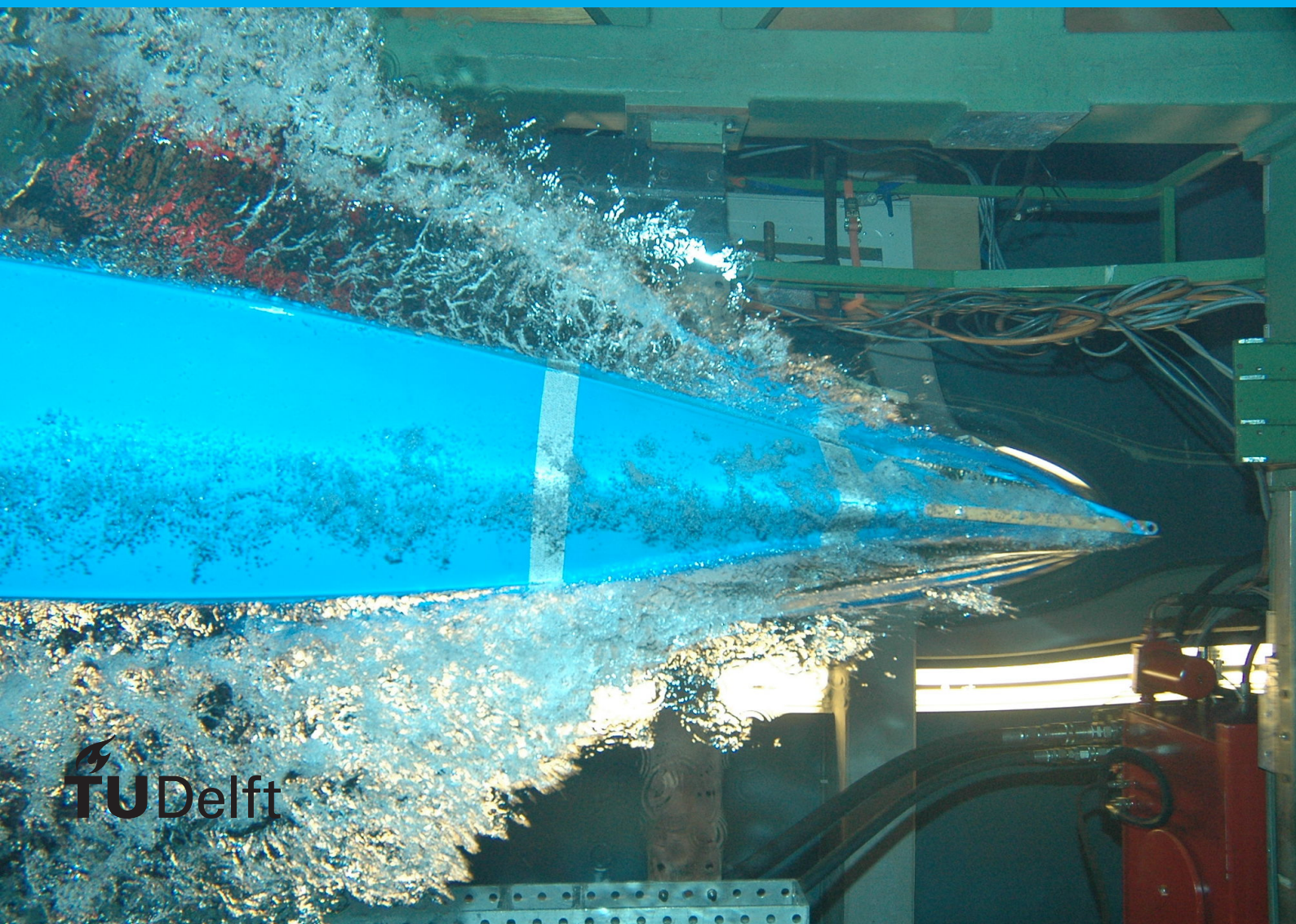
Non-parametric Estimation of Generators of Elliptical Distributions

Bachelor Thesis Report

Victor Ryan



Technische Universiteit Delft
Faculteit Elektrotechniek, Wiskunde en Informatica
Delft Institute of Applied Mathematics

Verdelingsvrije model van de generatoren van elliptische verdelingen
(Engelse titel: Non-parametric estimation of generators of elliptical distributions)

Verslag ten behoeve van het
Delft Institute of Applied Mathematics
als onderdeel ter verkrijging

van de graad van

BACHELOR OF SCIENCE
in
TECHNISCHE WISKUNDE

door

VICTOR RYAN

Delft, Nederland
July 2022

BSc verslag TECHNISCHE WISKUNDE

"Verdelingsvrije model van de generatoren van elliptische verdelingen (Engelse titel: "Non-parametric estimation of generators of elliptical distributions"

Victor Ryan

Technische Universiteit Delft

Begeleider	**Commissieleden**
Dr. A.F.F. (Alexis) Derumigny	Dr. Ir. G.F. (Tina) Nane
...	...
July, 2022	Delft

Summary

In this thesis, we present simulation studies of a non-parametric estimator, proposed by Liebscher (2005) [20]. This estimator uses a well-known non-parametric estimator called kernel density estimator. Non-parametric estimation is used when the parametric distribution of a given dataset is unknown. This technique is then applied with an assumption that the distribution in question has a density, so that its density can be estimated. The density that we are interested in, belongs to a class of elliptical densities which has a similar contour shape as the Gaussian distribution. One of an example of such density is the density of multivariate normal distribution. Liebscher's estimator uses elliptical density to circumvent the 'curse of dimensionality'. The 'curse of dimensionality' often appears in non-parametric estimation. When a high dimensional dataset is applied to a non-parametric estimator, the convergence rate of the estimator becomes slow. This is what we refer to as the 'curse of dimensionality'. Liebscher's estimator circumvents this 'curse' by assuming that the multi-dimensional dataset is sampled from an elliptical distribution. The estimator then transforms the dataset into one-dimensional dataset, so that we can use the univariate kernel density estimation, instead of the multivariate ones. We use Liebscher's estimator to estimate the generator of elliptical distribution. The generator is a positive real-valued function. Liebscher's estimator depends on two parameters: the bandwidth parameter and the tuning parameter around the boundary. In this thesis, we investigate how these two parameter influence the performance of the estimator. We start with the case when the simulated dataset is sampled from the standard multivariate normal distribution. Then, we apply the estimator on a different elliptical distribution with different generator. As it turns out, finding the parameter such that the estimator gives a minimal error is a difficult task. This is because the area where the error is small, depends on the generator. We also observe that as we increase the dimension of the simulated dataset, the computational time of the estimator increases as well. Lastly, the estimator is applied to Wisconsin breast cancer dataset. The estimator is used to study the accuracy of a Bayes' classifier proposed by [1]. From the study, it appears that the role of the tuning parameter is smaller in comparison the bandwidth parameter, in changing the accuracy of the classifier.

Contents

1 Introduction

Suppose we know that a dataset is sampled from a distribution P that belongs to a parametric family $\{P_\theta : \theta \in \Theta\}$, where $\Theta \subseteq \mathbb{R}^k$. Then estimating P using the given dataset is equivalent to estimating the k-dimensional parameter θ. This is what we often refer to as "parametric estimation". On the other hand, if we do not know in which parametric family P belongs to, then the parameter space is a functional space which has infinite dimension. Such estimation is referred to as "non-parametric estimation" [37, Chapter 1].

Assuming that P has a density, we can use non-parametric estimation to estimate it given a dataset. A well-known non-parametric estimation technique is called kernel density estimation. This technique can be applied to one- or higher-dimensional dataset. However, kernel density estimation suffers from the 'curse of dimensionality', i.e. the performance of the estimator deteriorates for large dimension (see [24], [15], [9]). To avoid this curse, Stute and Werner (1991) [36] suggested to use densities that belongs to the class of elliptical densities. Such densities have the same elliptical contour shape as the Gaussian distribution.

A more formal definition of elliptical densities is that the density admits a representation

$$f(\mathbf{x}) := c_d \det(\Sigma)^{-1/2} g[(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})], \ \ \mathbf{x} \in \mathbb{R}^d, \tag{1.1}$$

where $d \geq 2$, c_d is a positive constant, Σ is a non-singular matrix, $\boldsymbol{\mu} \in \mathbb{R}^d$ is the centre of the given dataset and g is a Lebesgue measurable positive real-valued function. Stute and Werner's estimator uses $y := (\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})$ to estimate g by computing its kernel density estimator. Since $y \in \mathbb{R}$, we can use kernel density estimation for one-dimensional dataset to avoid the 'curse of dimensionality', provided that the density of the dataset is elliptical.

Unfortunately, their estimator faces some difficulties. It is stated in their simulation study (see [36, Section 3]) that the variance of the estimator increases to infinity as $\mathbf{x} \to \boldsymbol{\mu}$. To improve the convergence problem, Liebscher (2005) [20] proposed to transform y using a differentiable function. Then use it to compute the kernel density estimator of g. So, in addition to the smoothing (or bandwidth) parameter from the kernel density estimation in mind, Liebscher's estimator has an additional parameter in comparison to Stute and Werner's estimator. Through simulation studies, we would like to investigate how these two parameters influence the performance of Liebscher's estimator.

The report starts with theoretical background on elliptical distributions and kernel density estimation (Chapter 2). In the last section of Chapter 2 we also describe Liebscher's estimator. In Chapter 3, we apply Liebscher's estimator to simulated data. We outline the method of the simulation in Section 3.1. In Section 3.2 we estimate g when the dataset is sampled from multivariate normal distribution for small and large dimension. After that, we investigate the performance of the estimator for different g (Section 3.3). Lastly, we apply the estimators to Wisconsin breast cancer data (Chapter 4). Here, we are going to use Liebscher's estimator to study the accuracy of a Bayes classifier.

2

Literature Review

2.1. Elliptical distributions

Let $\mathbf{X}$ be a random vector in $\mathbb{R}^d$, $\boldsymbol{\mu}$ be a vector in $\mathbb{R}^d$, and Σ be a $d \times d$ non-negative definite matrix. It is said that $\mathbf{X}$ is elliptically distributed if the characteristic function $\boldsymbol{\varphi}(\mathbf{t})$ of $\mathbf{X} - \boldsymbol{\mu}$ can be written as

$$\boldsymbol{\varphi}(\mathbf{t}) := \exp(\mathrm{i}\mathbf{t}^T\boldsymbol{\mu})\psi(\mathbf{t}^T\Sigma\mathbf{t}),$$

for some function $\psi : \mathbb{R}_+ \to \mathbb{R}_+$.

In this report however, we are only interested in distributions that belong to a class of elliptically symmetric densities. The element of such class has a density that admits the following representation

$$f_{\mathbf{X}}(\mathbf{x}) := c_d \det(\Sigma)^{-1/2} g[(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})], \tag{2.1}$$

where $g : \mathbb{R}_+ \to \mathbb{R}_+$ and c_d is a constant [16, Chapter 6, p. 107]. We refer g as a generator of $f_{\mathbf{X}}$, Σ as the nonsingular scale matrix and $\boldsymbol{\mu}$ as the mean of $\mathbf{X}$. The distribution of $\mathbf{X}$ will be denoted as $\mathbf{X} \sim El(\boldsymbol{\mu}, \Sigma, g)$ if its density is defined as in (2.1). Since (2.1) is a density function, by [8, Lemma 2], we can integrate (2.1) on $\mathbb{R}^d$ to get:

$$c_d \frac{\pi^{d/2}}{\Gamma(d/2)} \int_0^\infty t^{d/2-1} g(t)\, dt = 1. \tag{2.2}$$

We use (2.2) to find the correct c_d for a fixed g, i.e.

$$c_d = \frac{\Gamma(d/2)}{\pi^{d/2}} \left(\int_0^\infty t^{d/2-1} g(t)\, dt \right)^{-1}. \tag{2.3}$$

To generate a random variable that has a density function given by (2.1), one can use another representation of elliptical distributions. Suppose $\mathbf{X} \sim El(\boldsymbol{\mu}, \Sigma, g)$ then

$$\mathbf{X} \overset{d}{=} \boldsymbol{\mu} + \Sigma^{1/2} R \mathbf{U}^{(d)}, \tag{2.4}$$

where R is a random variable on $\mathbb{R}^+$ that is independent of $\mathbf{U}^{(d)}$ having the density of $[(\mathbf{X}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{X}-\boldsymbol{\mu})]^{1/2}$ [16, Chapter 6, p. 110] and $\mathbf{U}^{(d)}$ is a random vector in $\mathbb{R}^d$ that is uniformly distributed on the unit sphere in $\mathbb{R}^d$; see [2, Theorem 1]. Note that if we define $Z := \Sigma^{-1/2}(\mathbf{X}-\boldsymbol{\mu})$, then $Z^T Z \overset{d}{=} R^2$. The density of Z is

$$f_Z(z) = c_d g(z^T z),$$

where the generator g is related to the density of R^2 by [22, Section 1.5, p. 36-37]

$$f_{R^2}(t) := c_d \frac{\pi^{d/2}}{\Gamma(d/2)} t^{d/2-1} g(t). \tag{2.5}$$

Here are some examples of elliptical distributions:

Multivariate normal distributions If $\mathbf{X} \in \mathbb{R}^d$ and $\mathbf{X} \sim \mathcal{N}_d(\boldsymbol{\mu}, \Sigma)$, then the density of $\mathbf{X}$ is

$$f(\mathbf{x}) := \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})\right). \tag{2.6}$$

Then (2.6) yields

$$g(t) := \exp\left(-\frac{1}{2}t\right) \quad \text{and} \quad c_d := \frac{1}{(2\pi)^{d/2}}. \tag{2.7}$$

Note that when (2.7) is substituted for g in (2.5), the function f_{R^2} in (2.5) is the density function of χ^2_d.

Multivariate t-distribution If $\mathbf{X}$ has the density

$$f(\mathbf{x}) := \frac{\Gamma((\nu+d)/2)}{\Gamma(\nu/2)(\nu\pi)^{d/2}\det(\Sigma)^{1/2}} \left[1 + \frac{1}{\nu}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})\right]^{-(\nu+d)/2}, \tag{2.8}$$

then $\mathbf{X}$ has multivariate t-distribution with the degrees of freedom $\nu > 0$. When $\nu = 1$, then the distribution of $\mathbf{X}$ is the multivariate Cauchy distribution [23].

From (2.8), we see that g is

$$g(t) := \left[1 + \frac{1}{\nu}t\right]^{-(\nu+d)/2} \quad \text{and} \quad c_d := \frac{\Gamma((\nu+d)/2)}{\Gamma(\nu/2)(\nu\pi)^{d/2}}. \tag{2.9}$$

2.2. Kernel density estimation

Assume that a given dataset has only one dimension. One often use the histogram, which is a non-parametric estimation technique, to estimate the density function of the dataset. However, if a smoother estimator is desired, then kernel density estimator is a more suitable choice. In this section, we explain what kernel density estimation is.

Let $X_1, \ldots, X_n$ be i.i.d with density $p(x)$. The kernel density estimator of p is defined in the following manner:

$$\widehat{p}_n(x;h) := \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right), \tag{2.10}$$

where $h > 0$ is the smoothing parameter called the "bandwidth", and K is a kernel function which has the following properties:

1. $K(u)$ is non-negative and integrable.
2. K is normalized, i.e.
$$\int_{\mathbb{R}} K(u)\, du = 1.$$
3. K is symmetric, i.e. $K(u) = K(-u)$ for any $u \in \mathbb{R}$.

Kernel function	$K(u)$
Rectangle (Unifrom)	$K(u) = \frac{1}{2}, \lvert u\rvert \le 1$
Triangular	$K(u) = (1 - \lvert u\rvert), \lvert u\rvert \le 1$
Gaussian	$K(u) = (2\pi)^{-1/2} \exp\left(-\frac{1}{2}u\right)$
Epanechnikov [7]	$K(u) = \frac{3}{4}(1-u^2), \lvert u\rvert \le 1$

Table 2.1: Examples of kernel functions

See Table 2.1 for some examples of common kernel functions. The estimation method using (2.10) is called the kernel density estimation (KDE).

In measuring the performance of (2.10), we first look at the mean square error (MSE) of the estimator. The results that are shown here are from [13]. To see more detail on the computation and other properties of KDE, we refer the reader to [26] and [29].

Recall that MSE of $\hat{p}_n$ for p is

$$\mathrm{MSE}(p(x); \widehat{p}_n(x;h)) := \mathbb{E}\|\widehat{p}_n(x;h) - p(x)\|^2$$

which can be decomposed into

$$\mathrm{MSE}(p(x); \widehat{p}_n(x;h)) = \mathrm{Var}[\widehat{p}_n(x;h)] + \{\mathbb{E}[\widehat{p}_n(x;h)] - p(x)\}^2. \tag{2.11}$$

Since $\mathrm{Bias}[\widehat{p}_n(x;h)] := \mathbb{E}[\widehat{p}_n(x;h)] - p(x)$, we see that the MSE is the sum of the variance of the estimator and its bias squared.

Let $M_2(K) = \int_{\mathbb{R}} u^2 K(u)\, du$ and $\|\cdot\|$ is the L^2-norm. Using variable substitution and Taylor expansion, we can obtain an approximation of the bias and the variance of KDE:

$$\mathrm{Bias}[\widehat{p}_n(x;h)] = \frac{h^2}{2} p''(x) M_2(K) + o(h^2), \text{ as } h \to 0, \tag{2.12}$$

$$\mathrm{Var}[\widehat{p}_n(x;h)] = \frac{1}{nh}\|K\|_2^2 p(x) + o\left(\frac{1}{nh}\right), \text{ as } nh \to \infty. \tag{2.13}$$

From (2.12), we see that to reduce the bias we need to choose small h, since it is approximately proportional to h^2. Additionally, we can derive from (2.13), that large h implies small variance. Suppose h is fixed, then $(nh)^{-1}$ will decrease as $n \to \infty$, and so the variance can also be reduced in this manner.

Now that we have the bias and the variance of KDE, Equations (2.12) and (2.13) yield

$$\mathrm{MSE}(p(x); \widehat{p}_n(x;h)) = \frac{h^4}{4} p''(x)^2 M_2(K)^2 + \frac{1}{nh}\|K\|_2^2 p(x) + o\left(h^4 + \frac{1}{nh}\right). \tag{2.14}$$

The challenge with KDE is to choose an h such that both the bias and the variance are small. It is important to choose such h to ensure the optimal uniform convergence rate [5].

Now let us investigate the mean integrated squared error (MISE) of $\widehat{p}_n(x;h)$. According to [13], this is because MISE has the advantage of measuring the estimation accuracy globally rather than local measures. The MISE of KDE is

$$\begin{aligned}\mathrm{MISE}[\widehat{p}_n(x;h)] &= \int_{\mathbb{R}} \mathrm{MSE}[\widehat{p}_n(x;h)]\, dx \\ &= \frac{h^4}{4}\|p''\|_2^2 M_2(K)^2 + \frac{1}{nh}\|K\|_2^2 + o\left(h^4 + \frac{1}{nh}\right)\end{aligned} \tag{2.15}$$

as $h \to 0$ and $nh \to \infty$.

We can then define AMISE, by leaving out the higher order term of (2.15):

$$\text{AMISE}[\widehat{p}_n(x;h)] = \frac{h^4}{4}\|p''\|_2^2 M_2(K)^2 + \frac{1}{nh}\|K\|_2^2. \tag{2.16}$$

Using (2.16), we find an h such that AMISE is minimal. This is done by solving for h in equation

$$\frac{d}{dh}\text{AMISE}[\widehat{p}_n(x;h)] = 0.$$

It yields

$$h^* := \left(\frac{\|K\|_2^2}{\|p''\|_2^2 M_2(K)^2 n}\right)^{1/5} \sim n^{-1/5}. \tag{2.17}$$

We can plug in h^* in (2.16) to get

$$\text{AMISE}[\widehat{p}_n(x;h^*)(x)] = \frac{5}{4}\left(\|K\|_2^2\right)^{4/5}\left\{\|p''\|_2^2 M_2(K)^2\right\}^{1/5} n^{-4/5} \sim n^{-4/5}. \tag{2.18}$$

It means that, with an optimal bandwidth, the estimator converge at the rate $n^{-4/5}$.

Unfortunately, the optimal bandwidth in (2.17) is not applicable in practice since it still depends on the unknown function p''. There are however several bandwidth selection methods that are frequently used. One of the method was introduced by Silverman in [34], which often is referred to Silverman's rule of thumb

$$h_{\text{Silv}} := 1.06\widehat{\sigma}n^{-1/5},$$

where $\widehat{\sigma}$ is the estimated standard deviation of the data. A more robust bandwidth selection method uses the interquartile range

$$IQR := \text{upper quartile} - \text{lower quartile}.$$

We refer the bandwidth selection method that uses this quantity as h_{rot}, where 'rot' stands for 'rule of thumb':

$$h_{\text{rot}} := 1.06\min\left\{\widehat{\sigma}, \frac{IQR}{1.34}\right\} n^{-1/5}; \tag{2.19}$$

see [13, Section 3.3.1].

There are more sophisticated bandwidth selection other than the rule of thumb that Silverman proposed. For instance, the ordinary least squares cross-validation computes

$$\min_{h>0}\text{CV}(h) := \int \widehat{p}_n^2(x;h)\, dx - \frac{2}{n}\sum_{i=1}^{n}\widehat{p}_{n,-i}(X_i;h), \tag{2.20}$$

where $\widehat{p}_{n,-i}(X_i,h)$ is the estimated density without the ith sample. However [11] and [12] state that the convergence rate is slow. Despite of this, Stone (1984) [35] proves that (2.20) is indeed optimal asymptotically.

The cross-validation method has been altered and modified to improve its performance. If the reader wishes to see other variation of cross-validation method and discussion/review of them, then see [18] and [30]. Lastly, it is mentioned in [17] that cross-validation and Silverman's rule of thumb are not recommended for general practice. Instead, they investigate other data-based bandwidth selection such as [25] and [32].

2.3. Estimating the generators of elliptical distributions: Liebscher's estimator

As we have mentioned in Chapter 1, we can use non-parametric estimation when we do not know which parametric family the dataset is sampled from. Unfortunately its accuracy deteriorates for large dimension of $\mathbf{X}$. Stute and Werner suggested an estimator that overcomes 'the curse of dimensionality', by assuming that the density of the dataset is defined as in (2.1).

Stute and Werner's [36] estimator faces two difficulties. Firstly, the variance of the estimated $f_{\mathbf{X}}$ keeps increasing around $\boldsymbol{\mu}$. Then, there is a bias when estimating g around $t = 0$. The latter was fixed by modifying the KDE; Liebscher [20] improved it by using a differentiable function ψ, such that its derivative ψ' is bounded and $\lim_{x\to 0^+} x^{-d/2+1}\psi'(x)$ is a positive constant.

Let us describe the estimator that Liebscher proposed. The estimator uses kernel density estimator to estimate g, with estimators for $\boldsymbol{\mu}$ and Σ plugged-in. An example of estimator for $\boldsymbol{\mu}$ and Σ is the sample mean and the sample variance respectively. Let $\widehat{\boldsymbol{\mu}}_n$ and $\widehat{\Sigma}_n$ be such estimators of $\boldsymbol{\mu}$ and Σ. The proposed estimator is computed in the following way:

1. Let $X_1, \dots, X_n$ be i.i.d samples of random vectors in $\mathbb{R}^d$ from $f_{\mathbf{X}}$ and define a function $\psi : \mathbb{R}_+ \to \mathbb{R}_+$ by
$$\psi(x) = -a + (a^{d/2} + x^{d/2})^{2/d}, \tag{2.21}$$
with $a > 0$. Let $Y_{ni} = \psi((X_i - \widehat{\mu}_n)^T \widehat{\Sigma}_n^{-1}(X_i - \widehat{\mu}_n))$, $i = 1, \dots, n$. Using this newly acquired data, we estimate the density of Y_n by the KDE below:
$$\widehat{p}_n(y; h) = \frac{1}{h}\sum_{i=1}^{n}\left[K\left(\frac{y - Y_{in}}{h}\right) + K\left(\frac{y + Y_{in}}{h}\right)\right]. \tag{2.22}$$
Note that if $d = 2$ or $a = 0$, then $\psi(x) = x$, which is the estimator that was proposed by Stute and Werner (1991); see [36]. Further note that the kernel density estimation in (2.22) differs than the one in (2.10). This is the "mirror image" technique that fixes the second difficulty that Stute and Werner's estimator has. The "mirror image" technique is explained further in [31].

2. Using the estimator in (2.22), we obtain an estimator of the generator g as follows:
$$\widehat{g}_n(z) = \left(\frac{\pi^{d/2}}{\Gamma(d/2)}\right)^{-1} z^{-d/2+1}\psi'(z)\widehat{p}_n(\psi(z); h) \tag{2.23}$$
where $z \in \mathbb{R}_+$.

3. To estimate the density function of elliptical distribution, we simply plug-in $\widehat{\Sigma}_n$, $\widehat{\boldsymbol{\mu}}_n$ and (2.23) in (2.1):
$$\widehat{f}_n(\mathbf{x}) = \det(\widehat{\Sigma}_n)^{-1/2}\widehat{g}_n[(\mathbf{x} - \widehat{\boldsymbol{\mu}}_n)^T \widehat{\Sigma}_n^{-1}(\mathbf{x} - \widehat{\boldsymbol{\mu}}_n)]. \tag{2.24}$$

There are several assumptions for $\widehat{\Sigma}_n$, $\widehat{\boldsymbol{\mu}}_n$, K, ψ and h need to satisy so that the estimator $\widehat{f}_n$ converges uniformly as described in [20, Theorem 3.1]. Liebscher then was able to prove that the optimized convergence rate of his estimator is the same as the optimal one from one-dimensional kernel density estimation. But he also noted that the convergence rate is slow around $\boldsymbol{\mu}$. Other source such as [1], mentioned that Liebscher's estimator rules out many distributions of practical interest, since it requires at least four moments for the random variables of interest.

3

Simulation Studies

In this chapter, we investigate the estimator (see Section 2.3) of the density generator of (2.1) through simulations. We start with outlining the methods and steps that are used to produce the results shown in Section 3.2 and Section 3.3. We first restrict ourself by looking at a particular case when $f_{\mathbf{X}}$ is the density of the multivariate normal distribution. We are also interested in how long it takes to compute the estimated generator, as we increase the sample size and the dimension of the simulated dataset. After that, the performance of the estimator is investigated when g is not defined as in (2.7). The programming language that is used in this study is R, version 4.1.3 [27].

3.1. Method

In this simulation study, we set the mean $\boldsymbol{\mu} = \mathbf{0}$ and the scale matrix Σ as an identity $d \times d$ matrix I_d. First, let us list the input that we use in the simulation:

- The number of simulations n_{sim}. Here $n_{\text{sim}} := 100$.
- The number rows/sample of the data n that is used to estimate g. In this study, we look at $n := 25, 100, 500, 1000$.
- The bandwidth parameter h and the tuning parameter a around 0. Because $a = 0$ is equivalent to Stute and Werner's estimator, we decide to choose h and a in two ways so we can compare the two estimators:
  1. $a = 0$ and compute h_{rot} (see (2.19)), which is one of the Silverman's rule of thumb. We will refer this method as 'a0 & S', where 'S' stands for Silverman.
  2. the parameter a and h are chosen such that the error is minimal. We label this method as 'best a & h'.
- The dimension of the data d.
- The grid G which is used to compute $\widehat{g}$ shifts to $+\infty$ as the dimension increases. So we let

$$G := [0.1 + (d-2), 20 + (d-2)],$$

  which means that the grid shifts by $d-2$. We take 200 points from G such that the distance between each points is 0.1, i.e. we let $\Delta x := 0.1$. These points are denoted by $x_i, i = 1, \ldots, 200$ and the grid size of G is 200.

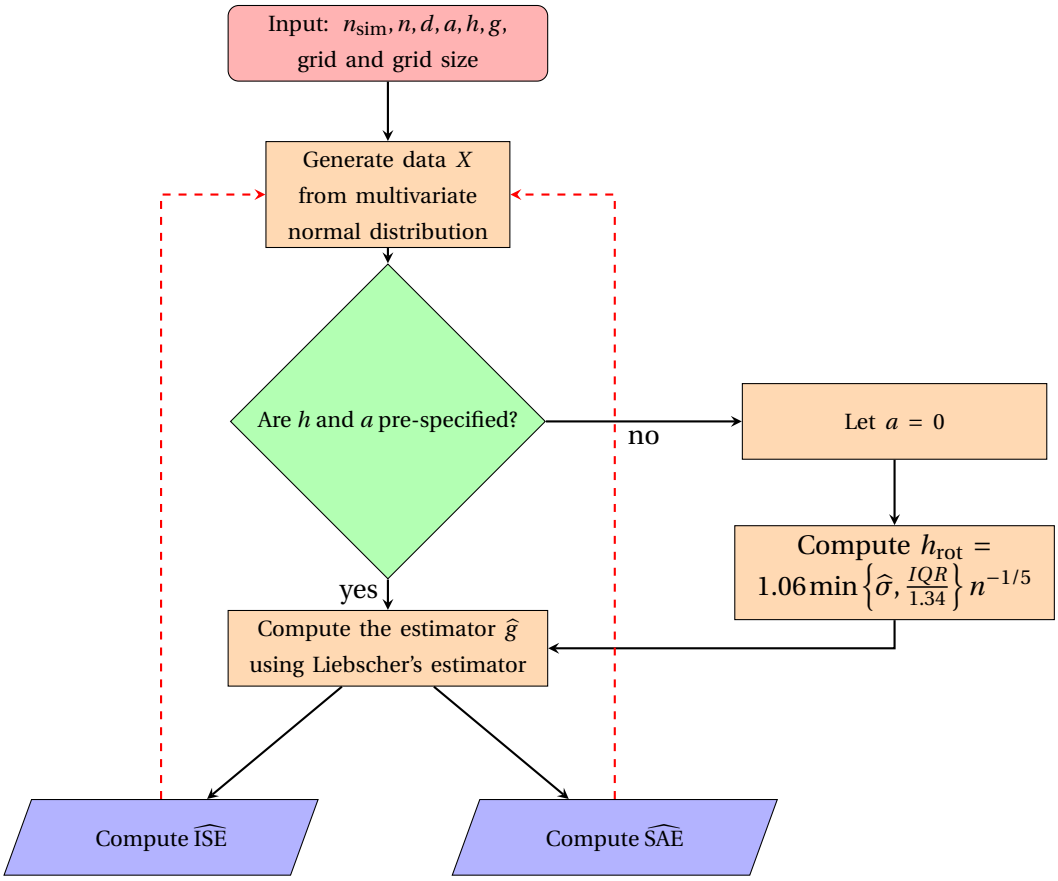


Figure 3.1: The flowchart of the simulation. The red dashed arrow indicates the loop.

Each simulation will generate data from a distribution with density (2.1), which then used to estimate g by applying the Liebscher's estimator (see Section 2.3). To generate the dataset, we use an R package `EllipticCopulas` [3] and its implementation is based on [4]. We choose the Epanechnikov kernel for the kernel density estimation (see Table 2.1).

In order to measure the performance of Liebscher's estimator, we decide to use the integrated squared error (ISE) and the supremum absolute value error (SAE):

$$\text{ISE} := \|g - \widehat{g}\|_2^2 = \int_0^{+\infty} \left|g(x) - \widehat{g}(x)\right|^2 dx \quad \text{and} \quad \text{SAE} := \|g - \widehat{g}\|_\infty = \sup_{x>0} |g(x) - \widehat{g}(x)|,$$

i.e. the L^2- and L^∞-norm error respectively. To estimate these two quantities, we use the following estimators:

$$\widehat{\text{ISE}} := \sum_{i=1}^{200} \left|g(x_i) - \widehat{g}(x_i)\right|^2 \cdot \Delta x \quad \text{and} \quad \widehat{\text{SAE}} := \max_{i=1,\ldots,200} \left\{\left|g(x_i) - \widehat{g}(x_i)\right|\right\}.$$

See Figure 3.1 for the illustration of the simulation.

Once the simulation is ended, we then obtain n_{sim} estimated errors. Let $\widehat{\text{ISE}}_j$ and $\widehat{\text{SAE}}_j$ be the estimated error in j-th simulation, $j = 1, \ldots, n_{\text{sim}}$. These estimated errors are used to estimate $\text{MISE} := \mathbb{E}\left[\left\|g - \widehat{g}\right\|_2^2\right]$ and $\text{MSAE} := \mathbb{E}\left[\left\|g - \widehat{g}\right\|_\infty\right]$ by taking the average over n_{sim}:

$$\widehat{\text{MISE}} = \frac{1}{n_{\text{sim}}} \sum_{j=1}^{n_{\text{sim}}} \widehat{\text{ISE}}_j, \quad \widehat{\text{MSAE}} = \frac{1}{n_{\text{sim}}} \sum_{j=1}^{n_{\text{sim}}} \widehat{\text{SAE}}_j.$$

3.2. Simulation results: the multivariate normal case

Let g and c_d be defined in (2.7), then the generated data is sampled from the standard multivariate normal distribution $\mathcal{N}_d(\mathbf{0}, I_d)$. We first generate one simulated d-dimensional dataset and apply Liebscher's estimator to get the estimated g (denoted by $\widehat{g}$) with a fixed h, a and n. The results of the estimation can be seen in Figure 3.2. Note that x-axis is different for each d due to how we define the grid G. It appears that the value of $\widehat{g}$ keeps getting smaller as d increases. Also, the estimator performs reasonably well when $n = 500$ and $a = h = 1$.

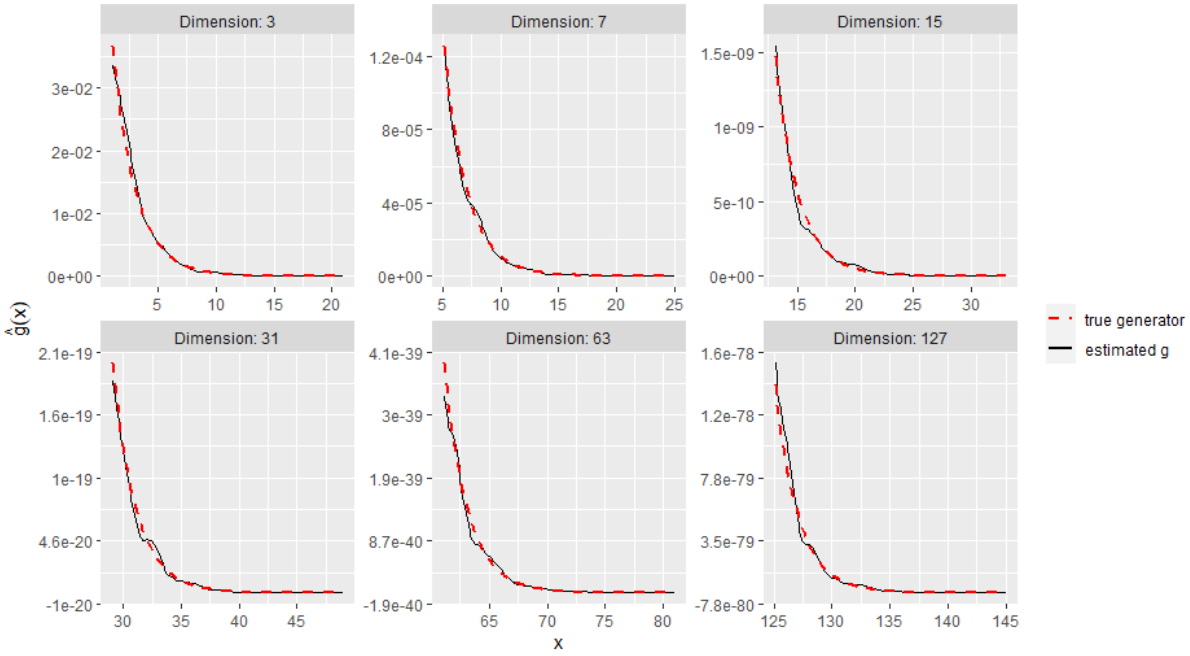


Figure 3.2: The estimated $g(x) = \exp(-x/2)$ from one simulated d-dimensional dataset. Here we choose $d = 3, 7, 15, 31, 63, 127$. Further, $a = 1, h = 1$ and $n = 500$. The true generator g is given by the red dashed curve. The estimator $\widehat{g}$ is the solid black curve.

Based from this figure, a number of question arise. Firstly, for different d and n, how does a and h influence the performance of Liebscher's estimator? Secondly, suppose there is an area where the errors are small, then how large is the interval of a and h that we can choose to get the 'best' estimate? Then, we also want to know how long the computational time is as we increase the sample size and the dimension of the simulated dataset. Lastly, we are curious about the relationship between the error and the sample size. We compare the estimated error when $a = 0$ and $h = h_{\text{rot}}$, with the errors using the 'best' a and h.

3.2.1. Contour plots of the errors as a function of (a, h)

To answer the first question, we measure the performance of the estimator with various combination of a and h and visualize it using contour plots. We decide to look at two different measure of performance: MISE and MSAE, i.e. the L^2- and L^∞-norm error. The parameters a and h range from 0.1 to 1000. However, since the range is too large, we only take 20 points out of this interval such that the distance between each point increases logarithmically.

We have seen in Figure 3.2 that the range of $\widehat{g}$ is different for each dimension. This means that, depending on d, the range of $\widehat{\text{MISE}}$ and $\widehat{\text{MSAE}}$ are different. This explains why the contour lines in Figure 3.3 have different values for each $d = 3, 7, 15, 31$. So it is appropriate to define the following set:

$$A_{n,d}^{(\text{Err})}(k_d) := \left\{(a, h) : \widehat{\text{Err}}(a, h, n, d) \leq k_d\right\},$$

where Err is either MISE or MSAE and k_d is a constant. Note that instead of using the notation k, we use k_d due to the difference in range of values for each d. Further note that $\widehat{\text{Err}}(a,h,n,d)$ depends on the choice of a and h, as well as n and d. The set $A_{n,d}^{(\text{Err})}(k_d)$ can be interpreted as the area of a and h such that the estimated error in that area is bounded above by k_d.

Now, let us focus on Figure 3.3 (see Figure 3.4 for the legend of these plots). Here, we plot the contours of the function $(a,h) \mapsto \text{Err}$ for different dimensions and n. There are four sub-figures: the two sub-figures 3.3a and 3.3b are the contour plots of L^2- and L^∞-norm error when $n = 25$. The other two sub-figures (3.3c and 3.3d) are structured similarly as 3.3a and 3.3b, but here $n = 1000$. In each sub-figure, we also plot results for $d = 3, 7, 15, 31$.

The coloured area can be interpreted as the set $A_{n,d}^{(\text{Err})}(k_d)$. The area with darker green colour is the area where the error is the smallest. For instance, $A_{25,3}^{(\text{MISE})}(k_3)$ has darker green colour when $k_3 \approx -4.2$. However, this would be the case when $k_7 \approx -9$ with the same n and Err. On the contrary, if the colour becomes lighter, then the error is larger in comparison to the area where the colour is green. So, we are able to see the different contour curves, when we choose the appropriate k_d for each d.

Let us fix the aforementioned k_3 and k_7. We observe that the area of $A_{n,3}^{(\text{Err})}(k_3)$ is smaller than $A_{n,7}^{(\text{Err})}(k_7)$. Further, let k_{15} and k_{31} be chosen such that $A_{n,15}^{(\text{Err})}(k_{15})$ and $A_{n,31}^{(\text{Err})}(k_{31})$ are coloured dark green. Then the h-values that belongs in their respective sets are more constrained than the a-values. These areas are stretched downwards starting around $d = 7$ or $d = 15$. As for the location of where the area has darker green (small error), it is quite similar for both the contour plots of $(a,h) \mapsto \widehat{\text{MISE}}$ and $(a,h) \mapsto \widehat{\text{MSAE}}$.

Another observation that we should point out, is how the contour curves are very similar across two distinct measures of performance. Visually, the contour curves in Figures 3.3c and 3.3d are similar. However, the values of these contour curves are not the same. If we look at Figure 3.3c, the most outer contour curve for $d = 15$ is approximately -17.6. In Figure 3.3d, with the same d, the most outer contour curve has value -8.9. The same observation can be made if we focus on L^2-norm error and compare between $n = 25$ and $n = 1000$. Visually 3.3a and 3.3c are similar, but the values on the contour lines are different. Additionally, in both $(a,h) \mapsto \widehat{\text{MISE}}$ and $(a,h) \mapsto \widehat{\text{MSAE}}$, we see that the most outer contour curve becomes more flatter as d increases.

From these contour plots, we conclude that the two performance measures give a similar behaviour. In both measures, we notice that the area with small error becomes more reduced as d increases. However, the vertical extent of such area increases for large d. It appears that we have to zoom into a part of the figure to have a clear visualization of where the error is small since the upper part of the contour plots gets flatter. The only noticeable difference between the two measures is the range of the values on the contour curves. We also have looked at the case when $n = 100$ and $n = 500$. Because the plots are visually similar than the ones in Figure 3.3, we decide to put them in Appendix A, Figure A.1.

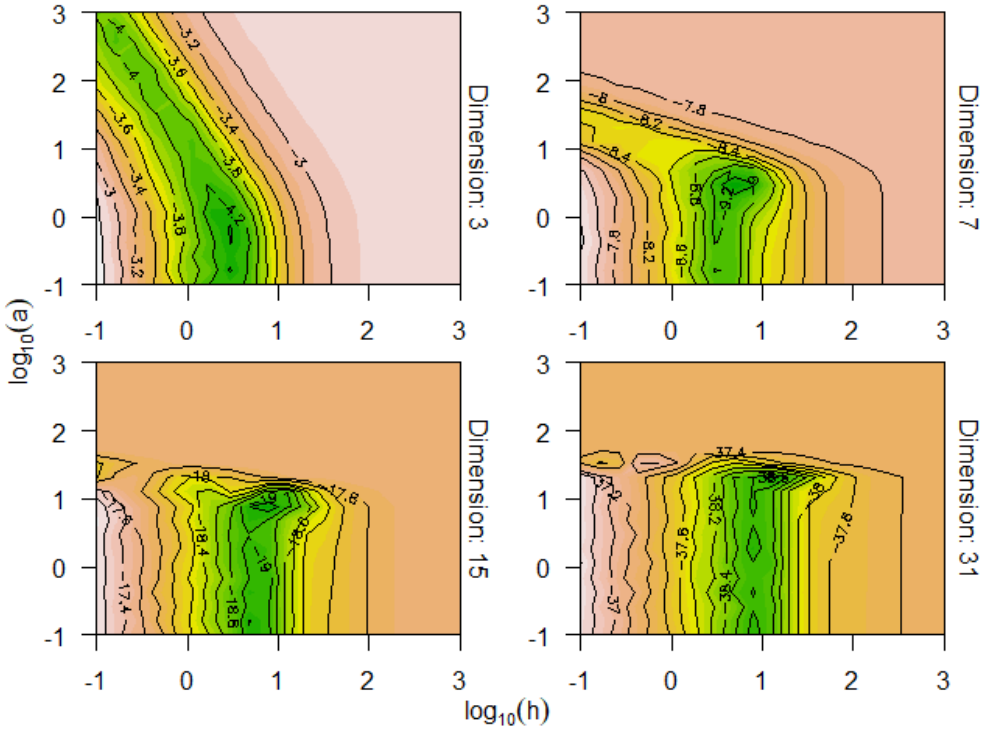
3.2.2. The relationship between the 'best' parameters and the sample size

In this part, we are going to find the parameters a and h for which the errors are small for each dimension. We call such a and h as the 'best' a and h. On top of that, we can also find out the relationship between n and the parameters of the estimator.

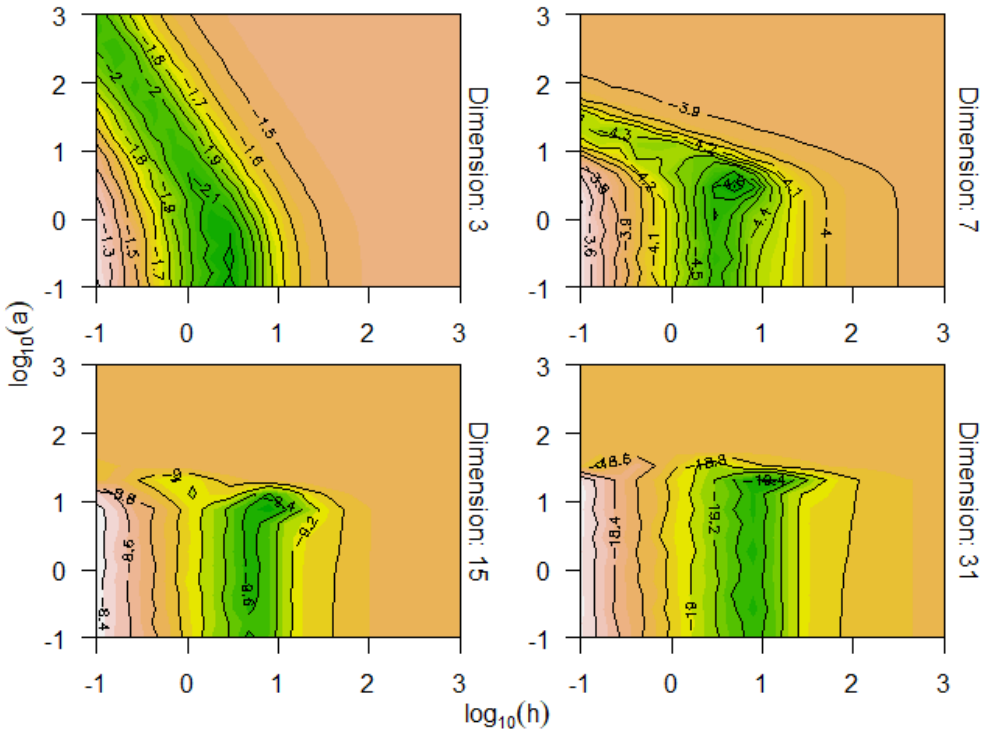
However, the 'best' parameters are obtained from the estimated errors. So, instead of looking at specific coordinates where the error is small, we also look at an acceptable interval of a and h where the error is 10% of the minimum error for instance. To define the interval more properly, we look at the set

$$\mathscr{A}_{n,d} := \left\{ a > 0 : \widehat{\text{Err}}(a,h,n,d) \leq q \cdot \min_{a>0, h>0} \widehat{\text{Err}}(a,h,n,d) \right\}.$$

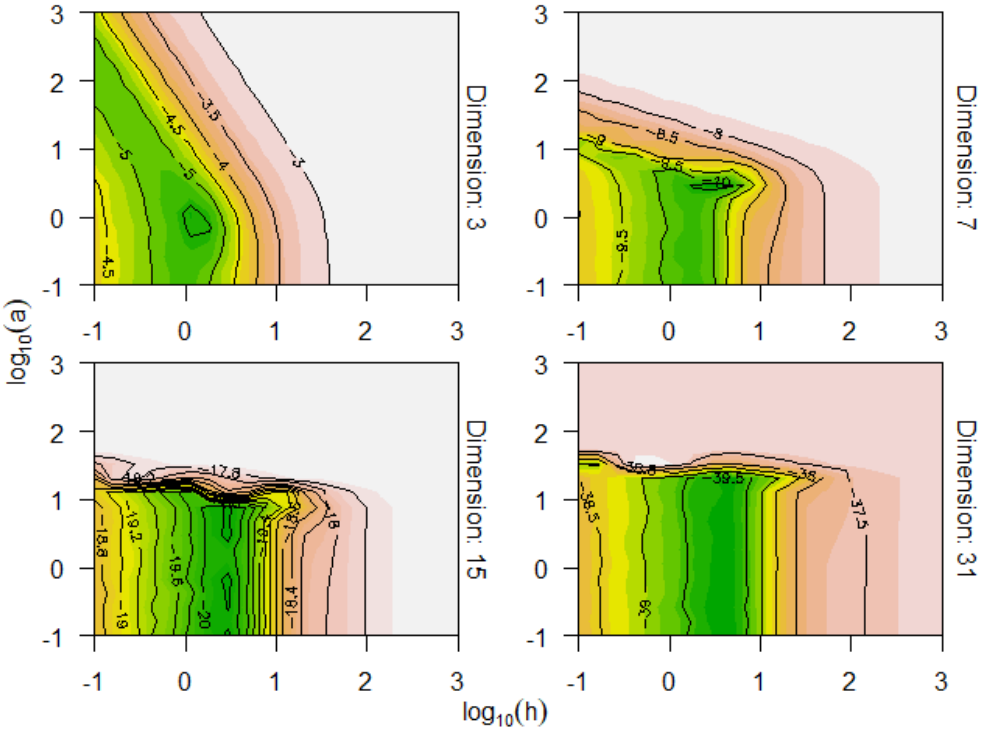
We define the set $\mathscr{H}_{n,d}$ similarly as $\mathscr{A}_{n,d}$, corresponding to the bandwidth parameter $h > 0$. Note that when $q = 1$, then these $\mathscr{A}_{n,d}$ and $\mathscr{H}_{n,d}$ contain the 'best' a and h respectively.



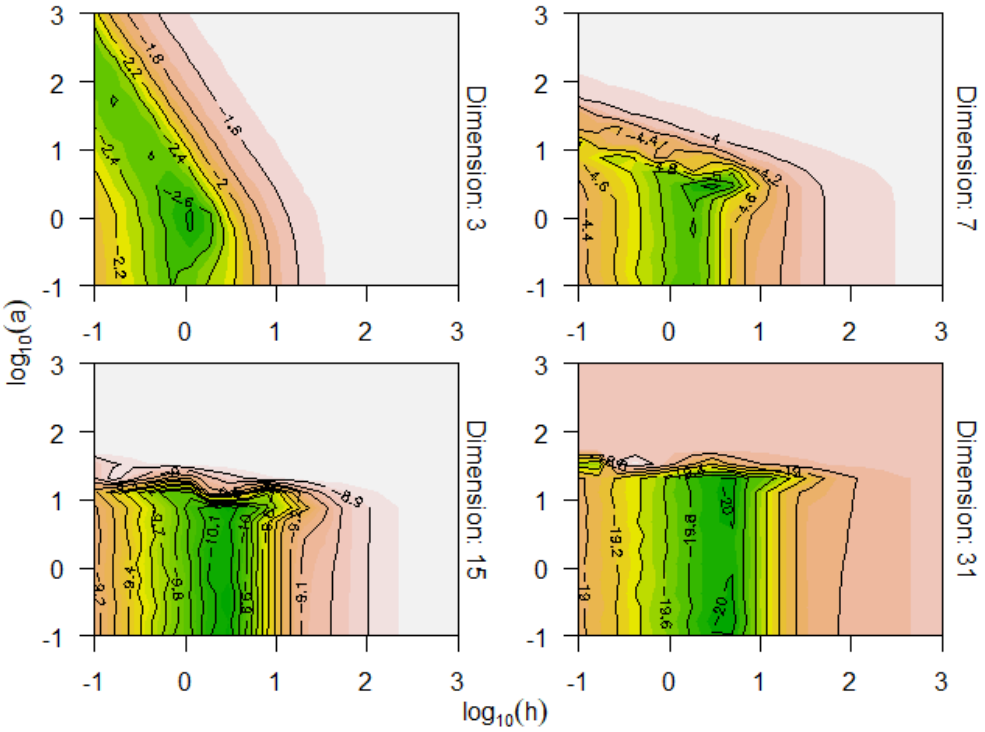
(a) $L^2 : n = 25$



(b) $L^\infty : n = 25$



(c) L^2 : $n = 1000$



(d) L^∞ : $n = 1000$

Figure 3.3: Contour plots of $\widehat{\text{MISE}}$ and $\widehat{\text{MSAE}}$ as a function of (a, h). The dimensions that are shown in each sub-figure are (from left to right) 3, 7, 15 and 31. In (a) and (b), the sample size is 25. As for (c) and (d), we choose $n = 1000$. Darker green colour means that $\widehat{\text{MISE}}$ (or $\widehat{\text{MSAE}}$) is smaller than the colour that are lighter.

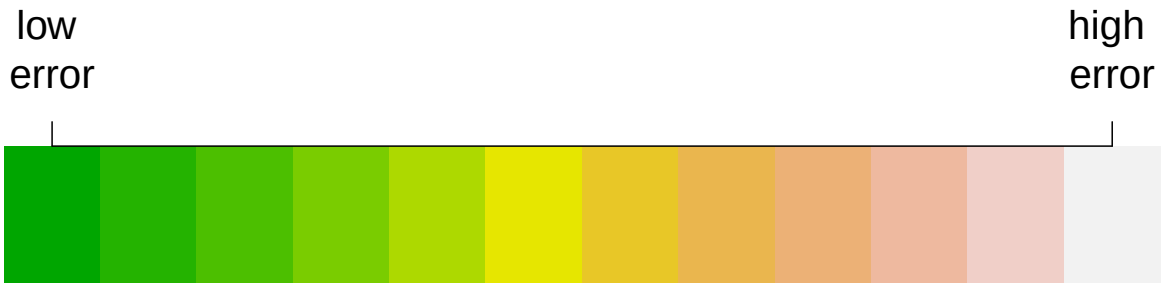


Figure 3.4: The legend bar for Figure 3.3.

Looking at the dashed line in Figure 3.5a, it seem that a behaves unpredictably due to large uncertainty. For instance, when $d = 3$ the dashed line increases slightly as n becomes large. But when $d = 127$ and $d = 31$, the best a seems to fluctuates. As for the best h, also visualized by the dashed line in Figure 3.5b, it decreases as $n \to \infty$. The same behaviour is observed even if we take a different measure of error. It might be difficult to see how the 'best' a and h change with n in Figure 3.5. So we refer the reader to Figure A.2a and A.2b in Appendix A. These figures are the visualization of the set $\mathscr{A}_{n,d}$ and $\mathscr{H}_{n,d}$ when $q = 1$.

Another thing that we can point out from Figure 3.5 is that choosing h matters much more than a, and this, for any choice of dimension. The set $\mathscr{A}_{n,d}$ becomes wider as $d \to \infty$ and it starts around $d = 7$ and $d = 15$. This means that we can choose any a from 0.1 until around 10 for $d = 7, 15$ to get a good estimate of g. The choice of a can even reach to $a = 100$ when $d = 127$. On the contrary, the set $\mathscr{H}_{n,d}$ is narrow for each d.

On the contrary, the set $\mathscr{H}_{n,d}$ is narrow for each d. We conclude that for larger dimension, we have more flexibility in choosing a than h to have a good estimate of g. The reason for this flexibility might be because g moves away from the boundary as $d \to \infty$ and the parameter a improves the behaviour of the estimator around the boundary. Therefore, the parameter a has less influence for sufficiently large d. The estimator's performance seems to be more sensitive to changes of the bandwidth parameter h. This tells us the importance of choosing h for both low- and high-dimensional dataset.

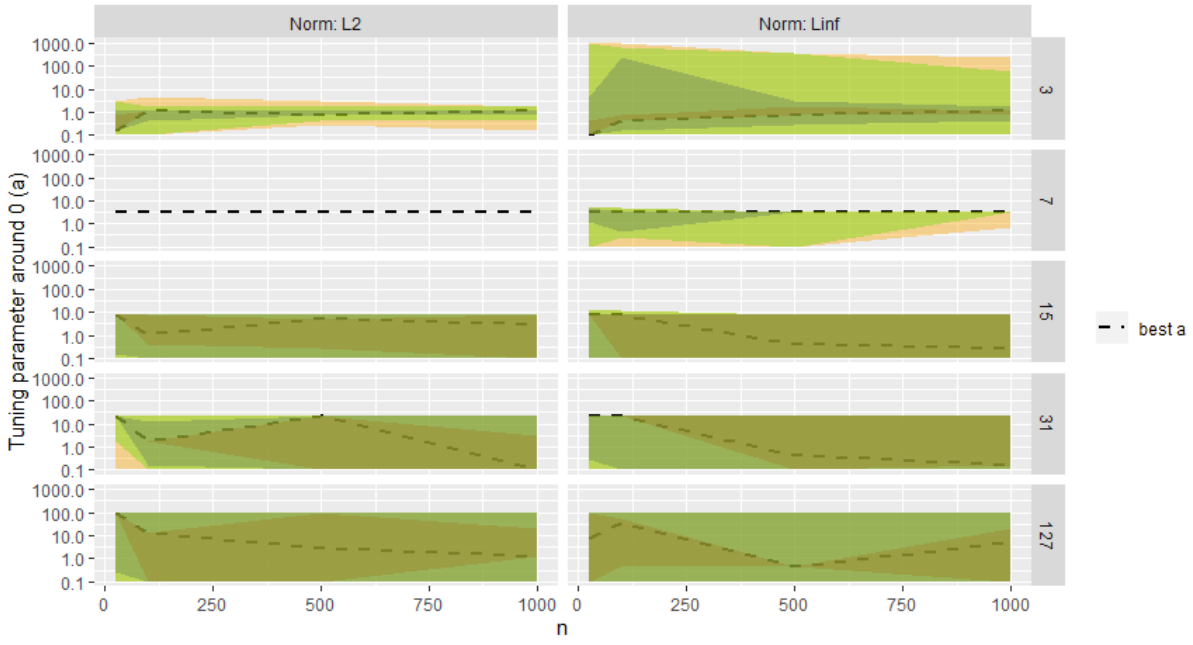
3.2.3. Computational time

In each simulation, we also manage to measure the computation time when estimating g. To summarize how long that takes, we choose the 'best' a and h obtained from the estimated MISE. We may also consider to choose the 'best' a and h from the estimated MSAE. However, we see in Figure 3.5 that the 'best' parameters, obtained from $\widehat{\text{MISE}}$, still lie in their respective set $\mathscr{A}_{n,d}$ and $\mathscr{H}_{n,d}$ (set Err in both sets to be MSAE).

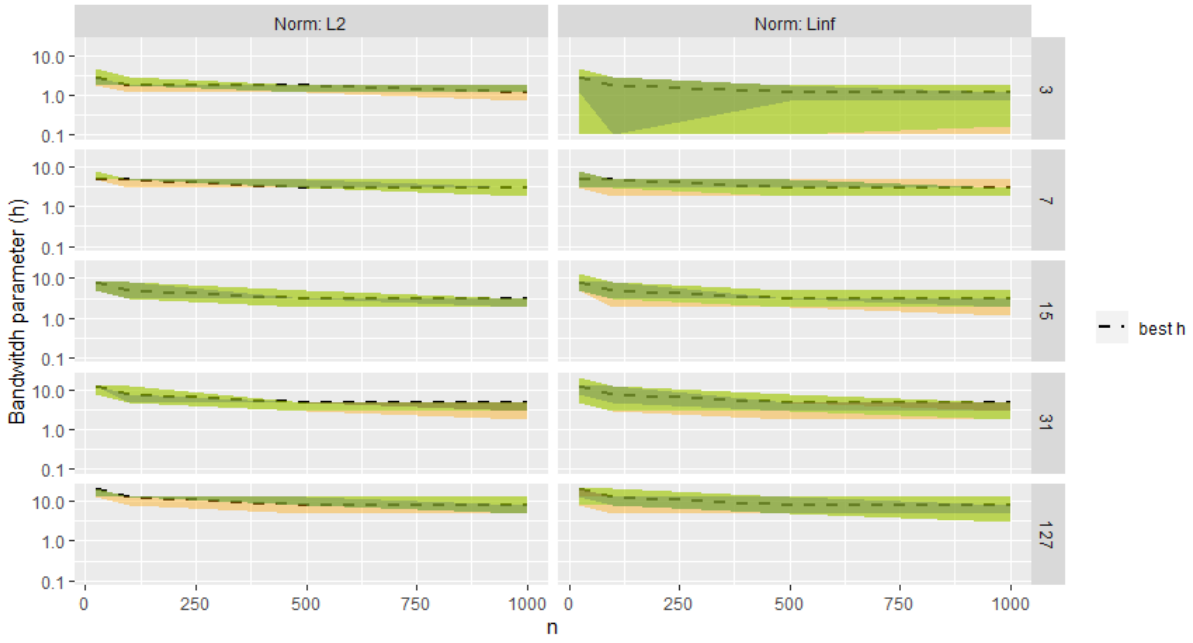
Using these parameters we compute $\widehat{g}_n$ using the simulated d-dimensional dataset. The simulation is also done n_{sim} times. We then obtained n_{sim} elapsed times of each simulation, whereafter we compute the average of these values.

The plot of the average computational time against n for each choice of d can be seen in Figure 3.6. We observe that the average computational time increase with n linearly for each choice of d. Further, for $d = 127$, the average computational time seems to be larger than any points for $d < 127$.

Let us relate the previous results that we have with this finding. We have learnt that when the dimension of the given dataset is large, we have more flexibility in choosing a. This means that the choice of h is the main concern, in order to obtain a good estimate of the generator. Combining the result in this subsection, we conclude that there is a cost in computational time when we increase the dimension of the dataset. That is, the higher the dimension of the dataset we have, the more time



(a)



(b)

Figure 3.5: The plots of a and h as a function of n for different dimension and norm error. The dashed line indicates the location a^* and h^* such that $\min_{a,h} \widehat{\mathrm{Err}}(a,h,n,d) = \widehat{\mathrm{Err}}(a^*,h^*,n,d)$. The coloured areas are the results of $\mathscr{A}_{n,d}$ (a) and $\mathscr{H}_{n,d}$ (b). We choose $q = 1.1, 1.4, 1.7, 1.9$

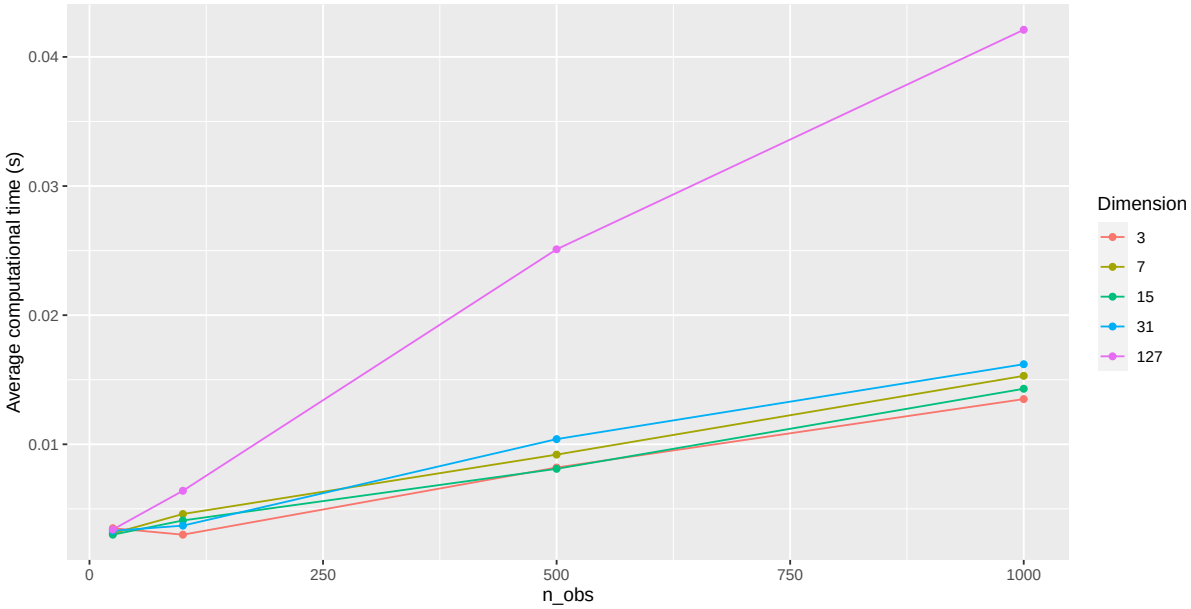


Figure 3.6: The line plots of the average computational time of $\widehat{g}_n$ against the sample size n. The different colours represents the dimension of the simulated dataset.

Liebscher's estimator takes to compute $\widehat{g}_n$. Fortunately, there is more flexibility in choosing a for a good estimate of g.

3.2.4. The relationship between the error and the sample size

In this last subsection, let us focus on $\widehat{\text{MISE}}$. Recall from Section 3.1, that we decide to choose a and h in two ways: 'a0 & S' ($a = 0$ and $h = h_{\text{rot}}$ as defined in 2.19) and 'best a & h' (a and h such that the error is minimal). With these two different methods, we are interested in how the estimated MISE changes as the sample size n increases.

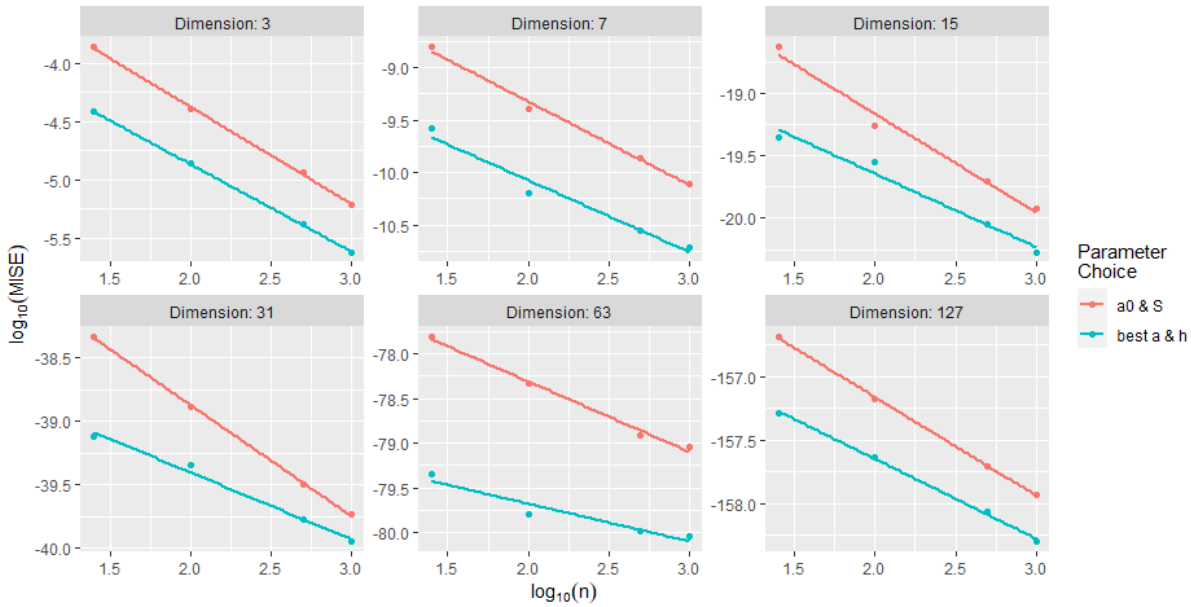


Figure 3.7: The linear relationship between $\log_{10}\widehat{\text{MISE}}$ and $\log_{10}(n)$ for each dimension, where the estimated L^2 norm error is computed using the method 'a0 & S' and 'best a & h'. The fitted line is plotted using ordinary linear regression.

First, we plot $\log_{10}(n)$ against $\log_{10}(\widehat{\text{MISE}})$ to see how the errors change when we have large sample size. In Figure 3.7, we see that the relationship between the two can be summarized using linear model. We refer the reader to Appendix B for the estimated values of the slope and the intercept of these lines. An interesting observation of Figure 3.7, is that the line of the group 'a_0 and S' seem to decrease faster than the line of the group 'best a and h'. Then one can expect that for large enough n, $a = 0$ and $h = h_{\text{rot}}$, the $\log_{10}$ of $\widehat{\text{MISE}}$ will be as large as the error when we choose the best a and h. This means that we may use Stute and Werner's estimator and h_{rot} to get as good as Liebscher's estimator, when we have a sufficiently large dataset ($n > 1000$).

3.3. Simulation results: other generators

So far, we only focus on the case of the standard multivariate normal distribution. Since Liebscher's estimator is a non-parametric estimation, we would like to see how the estimator performs for other generators. We would like to see whether the contour plots of the errors is similar to the ones in Section 3.2.1. The generators that we investigate in this section has one or multiple bumps (a similar study has been done in [4]). We also choose some generator that has steeper curve around $x = 0$ for instance.

3.3.1. Defining the generators and first impressions

There are six generators that we consider: $g_1(x) = 2\sqrt{x}(1+x^3)^{-1}, g_2(x) = 4e^{-x}(1+e^{-x})^{-2}, g_3(x) = e^{-x} + 4e^{-x}\sin^2(x), g_4(x) = e^{-x}\sin^2(2x), g_5(x) = e^{-x/2} + e^{-(x-2)^2/2}$ and $g_6(x) = x^2 e^{-x^2/3}$. The visualization of each of these generators can be seen in Figure 3.8. We estimate each of the generator by computing the Stute and Werner's estimator (i.e. $a = 0$) with different sample size n. As for the bandwidth parameter h, we compute the corresponding h_{rot}.

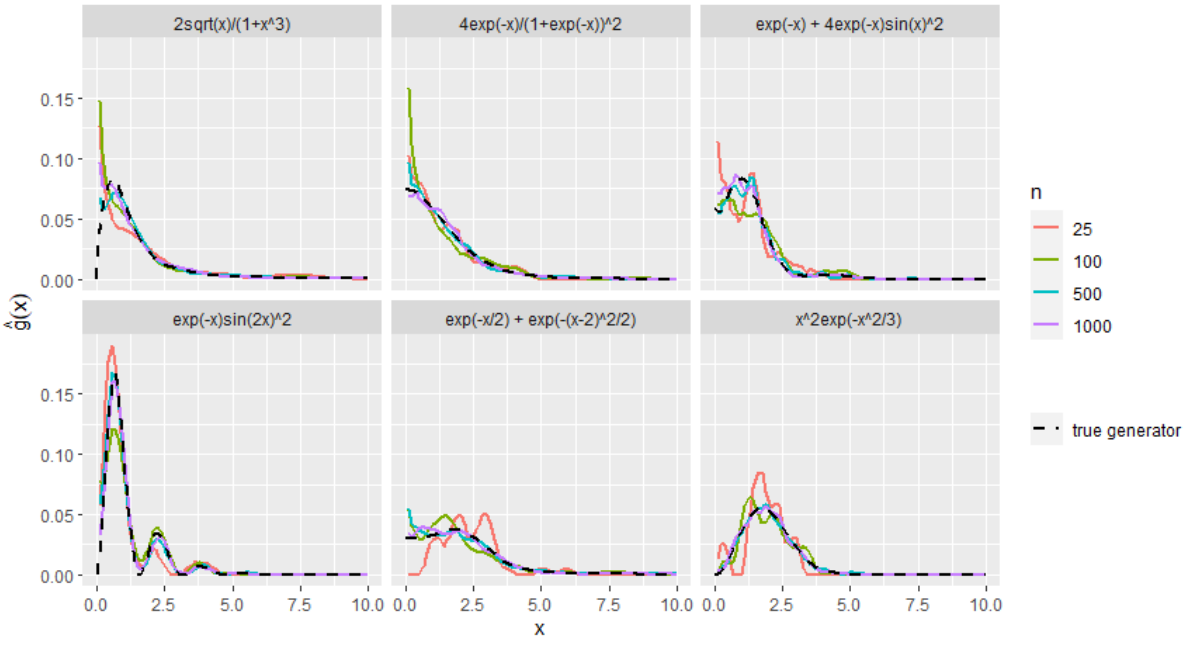


Figure 3.8: The estimated different generators where $d = 3$, $a = 0$ and h is computed by using (2.19). The true generator g is the black dashed curve.

Let $\widehat{g}_{i,n}, i = 1,\ldots,6$ be the estimated generator i obtained by using n sample size. In Figure 3.8 we see that $\widehat{g}_{1,n}$ does not perform well around $x = 0$ for any choice n of interest. For $\widehat{g}_{i,n}$ where $i \neq 1$ and $n = 1000$, each of the estimated generator lies very close to their corresponding true generator. Further, we observe that the estimator struggles to estimate the generator when there is a bump near

$x = 0$, e.g. $\widehat{g}_{3,n}, \widehat{g}_{5,n}$ and $\widehat{g}_{6,n}$ for $n \neq 1000$. Interestingly, when the generator has multiple bumps, the estimator $\widehat{g}_{4,n}$ performs better around $x = 0$ than $\widehat{g}_{1,n}$ around the same point.

Overall, the estimator can adequately estimate different generator other than (2.7). As usual, one can improve the estimator's performance by having larger sample size. But in this case, we might want to use the tuning parameter a as well, especially when n is small ($n < 1000$).

In the next part of this section, similar to what we did in Section 3.2.1, we vary a and h to investigate how Liebscher's estimator perform when the dataset is not sampled from the Gaussian distirbution.

3.3.2. Contour plots for a case $d = 3$

Let us focus on MISE. We shall look at how the mean integrated squared error changes when a and h varies. Also, we do another 100 simulations, in which we set $a = 0$ and compute h_{rot} of the simulated dataset, sampled from elliptical distribution with generator $g_i, i = 1, \dots 6$. By doing this, we would like to know whether h_{rot} is a good bandwidth selection method.

We fix $n = 25$, $d = 3$ and we visualize the results by contour plots. Let

$$A^{(\text{MISE})}_{n,d,g}(c) := \left\{(a,h) : \widehat{\text{MISE}}(a,h,n,d,g) \leq c\right\},$$

which can be interpreted as the coloured area in Figure 3.9. One can observe that the shape of the area $A^{(\text{MISE})}_{n,d,g}(-3.5)$ differ for each generator. We also see that the average h_{rot}, illustrated by the black dashed line, intersect with the area $A^{(\text{MISE})}_{n,d,g}(-3.5)$ for g_1, g_3 and g_5, but not for the other three generators. It means that there are cases where we can use $h = h_{\text{rot}}$ if one wish to have a good estimate of the generator. But we still have to choose an appropriate a for different generators. This means that h_{rot} is not a reliable bandwidth selection method.

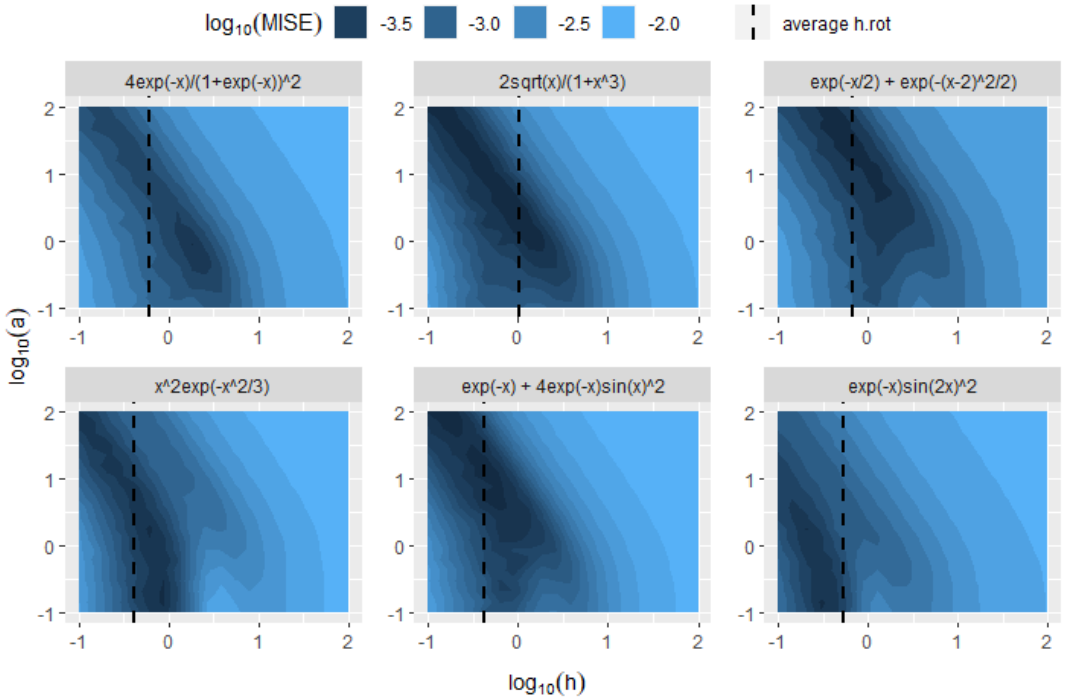


Figure 3.9: The values of $\widehat{\text{MISE}}$ as a function of $(\log_{10} a, \log_{10} h)$. The horizontal dashed line is the average bandwidth parameter that is obtained from (2.19). Here we choose $a = 0$ and $d = 3$.

Lastly, let us look at what happen on $\widehat{\text{MISE}}$ for $a = 0$ and compare the results with $n = 1000$. In Figure 3.10, we plot $h \mapsto \log_{10} \widehat{\text{MISE}}, h \in (0,1]$ for $n = 25, 1000$. Further, we add two vertical lines that

represents the average h_{rot}, obtained by datasets with the corresponding sample size n. The colours correspond to the sample size that are used to compute the estimator.

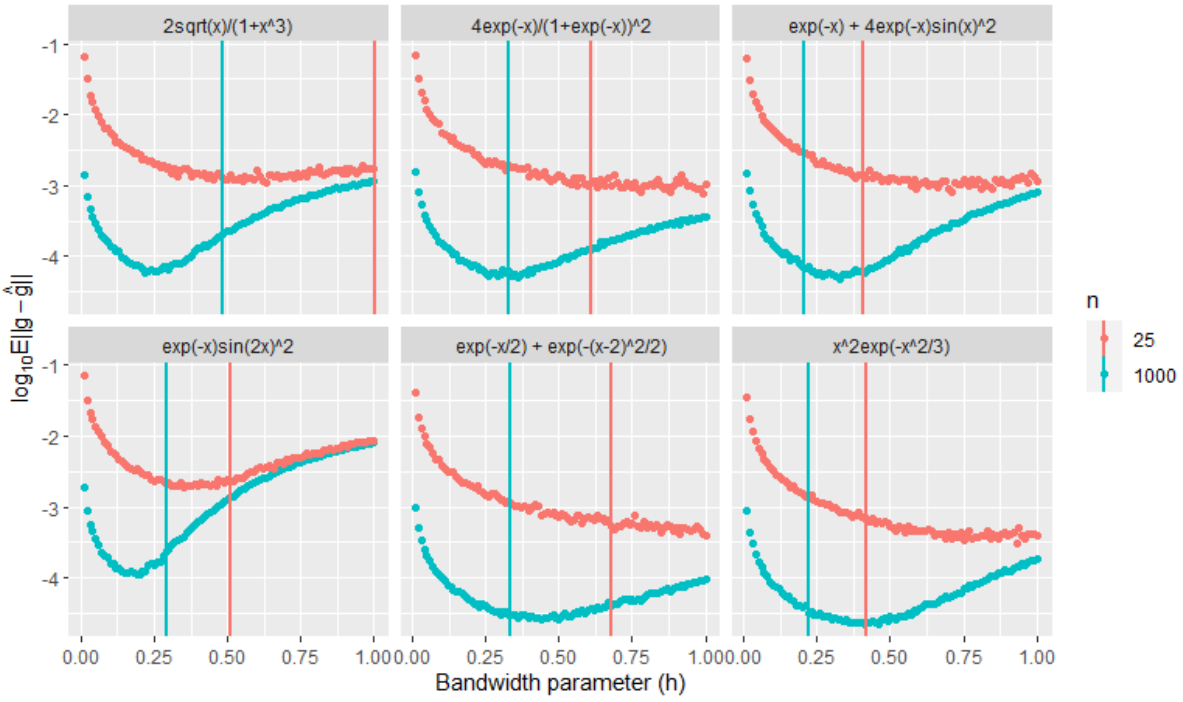


Figure 3.10: The $\log_{10}$ of $\widehat{\text{MISE}}$ as a function $h \in (0,1]$, $d = 3$ and $a = 0$ for different generators. The different coloured points represents how many sample is used to estimate $\hat{g}$. These same colours are used to represent the location of the average h_{rot}, computed by using 25 or 1000 samples.

We observe in Figure 3.10 that the required h to minimize the L^2 norm-error is smaller for large n, i.e. the average h_{rot} is smaller when $n = 1000$.. This same observation was also seen in Section 3.2.2. Nevertheless, it seems that choosing $h = h_{\text{rot}}$ may still gives us a good estimator for some generator (e.g. g_2, g_3 and g_5). But there is also case where such choice of h does not give the smallest $\widehat{\text{MISE}}$ (e.g. generator g_1).

We conclude that determining the best a and h appears to be a hard problem. This is because the area where the performance of the estimator at its best, are different for each generator. Further, choosing $h = h_{\text{rot}}$ appears to be unreliable, which is a reason to not use this bandwidth selection method. Though, we find that having a large sample size improves the performance of the estimator, when we use Silverman's rule of thumb. Nevertheless, even if one wishes to use this method, we still have to choose an appropriate a, for a small dimensional dataset.

4

Application on Real Dataset: Statistical Classification Problem

In the health care industry, there are diseases that are difficult to identify and diagnose. Fortunately, there are programs that can efficiently and accurately identify heart diseases [19]. Further, the authors in [21] present overview of programs that can detect dermatological diseases using image processing. Both sources use a technique, called "machine learning" (ML), to achieve their goal.

Machine learning, cited from [28], is

> "a field of computer science that studies algorithms and techniques for automating solutions to complex problems that are hard to program using conventional programming methods."

The machine essentially learns a set of labels from a given dataset. Then, it uses them to predict the labels of data points that are not in the given dataset.

There are several ways of how the machine learns. In [28], they mention four learning models: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning. In supervised learning, which is the focus on this chapter, the machine uses a dataset in which data points have already been classified (e.g. by an expert). The machine then learns from these classifications, to predict a classification from new unlabelled data points. One of the categories that falls into supervised learning is the classification problem.

A classification problem occurs when an observation can be categorized into a disjoint set of classes. The categorized data is used to predict in which class a data point belongs to after the machine has learnt a given dataset with the correct classes. A classifier that can be applied to make such predictions is the naive Bayes classifier. This classifier is based on the well-known Bayes theorem:

$$\mathbb{P}(B_j|A) = \frac{\mathbb{P}(A|B_j)\mathbb{P}(B_j)}{\sum_i \mathbb{P}(A|B_i)\mathbb{P}(B_i)} \tag{4.1}$$

where $\{B_1, B_2, \ldots\}$ is a partition of the sample space such that $\mathbb{P}(B_j) > 0$ for any j. To read more about other classifier, see [10].

The classifier that we are using is a Bayes' classifier that is based on the estimated posterior probabilities. This method was proposed by [1] as a possible application of their non-parametric estimation technique. For the classifier, they seem to replace $\mathbb{P}(A|B_j)$ with an estimated conditional density function of the given dataset. In this thesis, we estimate it using Liebscher's estimator. Since this estimator depends on the bandwidth parameter h and the tuning parameter a, we will investigate how these two parameters influence the accuracy of the proposed classifier.

To investigate the accuracy, the Wisconsin breast cancer (diagnostic) dataset [6] is used for learning the characteristics of two types of tumour: benign and malignant tumour. The dataset consists of 569 breast tumour patients, in which 212 of them are diagnosed with malignant tumour and 357 patients are diagnosed with benign tumour. Further, there are 10 characteristics of each cell nuclei observation of the patients. We will not use 10 of them and we assume that the measured characteristics of interest are sampled from elliptical distribution.

This chapter is organized as followed: in Section 4.1 we explain how the classifier is defined. In Section 4.2, we explain what we mean by the 'accuracy'. Lastly, in Section 4.3, we visualize how h and a influence the accuracy of the classifier.

4.1. Method

Let the train set be the dataset that is used to construct the classifier. In this section, we outline how the train the set is used in the machine, so that it classifies new data points into either benign or malignant class.

Let n be the sample size of the train set, $\#\mathrm{B}_{\text{train}}$ and $\#\mathrm{M}_{\text{train}}$ be the number of benign and malignant cases in the train set respectively. Let $\mathbf{y} \in \mathbb{R}^d$ be a new data point that we want to classify. Then we construct the classifier using the ones that were used in [1], which is a Bayes' classifier based on the estimated posterior probabilities:

$$\widehat{\mathbb{P}}(\mathrm{M}|\mathbf{y}) = \frac{\widehat{f}_n(\mathbf{y}|\mathrm{M})\widehat{\mathbb{P}}(\mathrm{M})}{\widehat{f}_n(\mathbf{y}|\mathrm{M})\widehat{\mathbb{P}}(\mathrm{M}) + \widehat{f}_n(\mathbf{y}|\mathrm{B})\widehat{\mathbb{P}}(\mathrm{B})} \text{ and } \widehat{\mathbb{P}}(\mathrm{B}|\mathbf{y}) = \frac{\widehat{f}_n(\mathbf{y}|\mathrm{B})\widehat{\mathbb{P}}(\mathrm{B})}{\widehat{f}_n(\mathbf{y}|\mathrm{M})\widehat{\mathbb{P}}(\mathrm{M}) + \widehat{f}_n(\mathbf{y}|\mathrm{B})\widehat{\mathbb{P}}(\mathrm{B})}. \tag{4.2}$$

Here, $\widehat{\mathbb{P}}(\mathrm{B}) = \#\mathrm{B}_{\text{train}}/n$ and $\widehat{\mathbb{P}}(\mathrm{M}) = \#\mathrm{M}_{\text{train}}/n$. The estimator $\widehat{f}_n$ is the Liebscher's estimator as defined in (2.24). After (4.2) is computed, we do the following procedure: if $\widehat{\mathbb{P}}(\mathrm{M}|\mathbf{y}) < \widehat{\mathbb{P}}(\mathrm{B}|\mathbf{y})$ then the data point of $\mathbf{y}$ belongs to benign class, otherwise $\mathbf{y}$ belongs to malignant class.

From the simulation studies, we have seen that the choice of parameter h is more constrained than parameter a, especially for high-dimensional dataset. Therefore, for the Liebscher's estimator, we look at different combination of h for $\widehat{f}_n(\mathbf{y}|\mathrm{B})$ and $\widehat{f}_n(\mathbf{y}|\mathrm{M})$. We denote h_{Benign} and $h_{\text{Malignant}}$ as the bandwidth parameter for $\widehat{f}_n(\mathbf{y}|\mathrm{B})$ and $\widehat{f}_n(\mathbf{y}|\mathrm{M})$ respectively. The influence of a on the accuracy of the classifier will also be investigated, but we decide to choose a for both $\widehat{f}_n(\mathbf{y}|\mathrm{B})$ and $\widehat{f}_n(\mathbf{y}|\mathrm{M})$.

4.2. Sensitivity, specificity and accuracy

In the previous section, we have explained a method on classifying new data points into either benign or malignant class. Here, we explain how the accuracy of the classifier is computed.

In order to avoid over-fitting, we decide to split the Wisconsin dataset into two sets: 60% of its data points are randomly chosen to construct the classifier (i.e. the train set) and the rest is used to measure the accuracy. The data points in the test set will be classified and we will compare the prediction with the true labels in that set.

To compare the classification that is made by the algorithm with the true classification from the given dataset, one can use a confusion matrix [33]. Let us start with explaining what a "confusion matrix" is. It is a square matrix such that the number of columns and rows is equal to the number of classes in the given dataset. In the Wisconsin dataset, there are two classes: benign (B) and malignant (M) tumour. Therefore, we have a 2×2 confusion matrix.

In the confusion matrix, we use the following terminologies:

True Benign (TB) is the number of data points for which the algorithm correctly predict the benign class. The number of correct malignant class prediction is similarly referred to as **True Malignant (TM)**

False Benign (FB) is the amount of misclassification of the benign class, i.e., the number of incorrect prediction as having benign tumour. Similarly, we call the number of incorrect prediction as having malignant tumour, as **False Malignant (FM)**

The confusion matrix for Wisconsin dataset can be seen in Table 4.1. Let $\#B_{test}$ and $\#M_{test}$ be the

	Benign (From test set)	Malignant (From test set)
Predicted Benign	TB	FB
Predicted Malignant	FM	TM

Table 4.1: A confusion matrix of the Wisconsin dataset

number of patients diagnosed with benign and malignant tumour in the test set respectively. Note that TB + FM = $\#B_{test}$ and TM + FB = $\#M_{test}$.

Using Table 4.1, we can now compute sensitivity, specificity and accuracy of the classifier. The sensitivity is computed by taking the ratio of TB with $\#B_{test}$. The specificity is the ratio of TM with $\#M_{test}$. Formally we define these two quantities as the following:

$$\text{sensitivity} := \frac{\text{TB}}{\#B_{test}} \quad \text{and} \quad \text{specificity} := \frac{\text{TM}}{\#M_{test}} \tag{4.3}$$

The sensitivity shows how good the classifier with predicting the benign class, while the specificity measures the performance of predicting the malignant class.

To compute the accuracy, we take the ratio of the sum of TB and TM with the sample size of the dataset:

$$\text{accuracy} := \frac{\text{TB} + \text{TM}}{\text{TB} + \text{FM} + \text{FB} + \text{TM}} = \frac{\text{TB} + \text{TM}}{\#B_{test} + \#M_{test}}. \tag{4.4}$$

The accuracy is interpreted as the overall performance of the classifier, i.e., how good is the classifier with predicting both benign and malignant class correctly.

Recall that we split the dataset randomly. This means that the measured accuracy on the test set is random. So we decide to estimate the final accuracy by repeating the procedure 10 times. Then compute the average of the accuracies among the 10 repetitions. The goal of this averaging is to improve and stabilize the estimation by reducing the variance.

4.3. Classification results

The Wisconsin dataset consists of ten real-valued columns. In this report we are interested in five of them: texture, area, smoothness, compactness and fractal dimension of the worst nuclei observations for each patient (5-dimensional dataset). We also consider a dataset that contains only the worst texture and worst smoothness (2-dimensional dataset). Lastly, we look at another dataset where we add the feature 'worst area' to the aforementioned 2-dimensional dataset. This 3-dimensional dataset was also used in [1].

For each dataset, we split the dataset into benign class and malignant class after we divide it into the train and test set. We split them based on the diagnosis to compute $\widehat{f}_n(\mathbf{y}|\text{B})$ and $\widehat{f}_n(\mathbf{y}|\text{M})$. We also estimate the mean and the variance of each class in the train set using the sample mean and sample variance. Then, their respective generator are estimated and compare them with function g

as defined in (2.7), i.e. the generator of the multivariate normal distribution (see Figure 4.1). This is because we want to see if the dataset we are using can be approximated by some elliptical density. We observe that the estimated generator lies very close to the generator as defined in (2.7). Though for $a = 1$ and $h = 1$, the estimated generator of the malignant group using 5-dimensional train set at $x = 0$ does not perform well. This is probably due to the inappropriate choice of a for the malignant group.

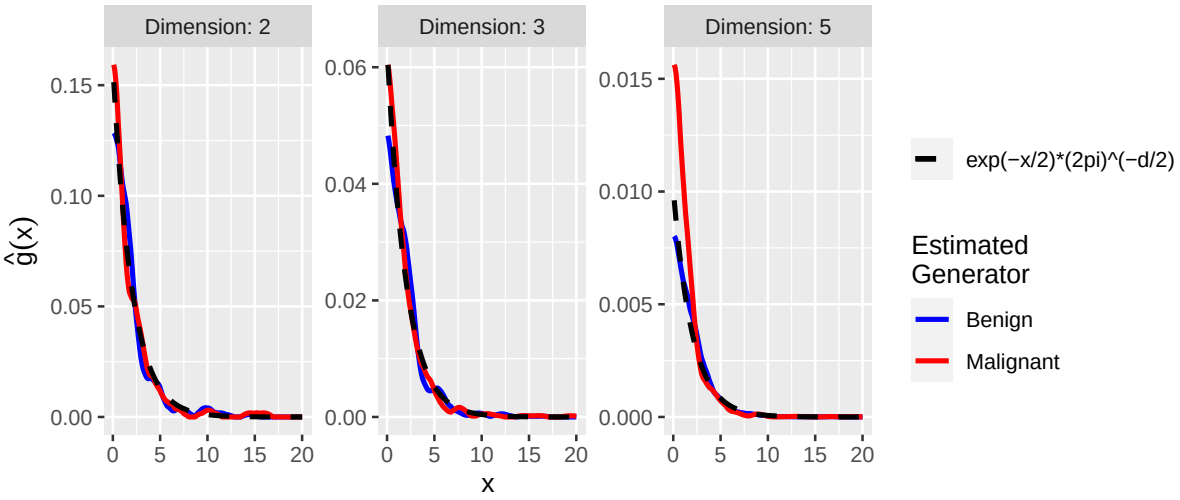


Figure 4.1: The estimated generators of both benign and malignant group from their corresponding train set. The dashed black line is the generator of (2.7). We choose $a = 1$ and $h = 1$.

4.3.1. The influence of h_{Benign} and $h_{\text{Malignant}}$ on the accuracy ($a = 0$)

Let us now show how the accuracy of (4.2) is influenced, when we let $a = 0$ and we vary the parameter h_{Benign} and $h_{\text{Malignant}}$. We start with the contour plot of the average accuracy using the 2-dimensional dataset (Figure 4.2). In this figure, we manage to capture an area where the average accuracy is the smallest (75% - 80% accuracy). We also observe that choosing a very small h for both class, results in a worst accuracy. Choosing a very large h for both groups seems to be not be a good idea as well as the accuracy is around 25% - 30%.

For comparison purpose, we do have investigated how the contour plots look like when we take the 3- and 5-dimensional dataset and $a = 0$ (see Figure 4.3 and Figure 4.4 respectively). It appears that when we choose the appropriate column, h and a, the classifier can have a high accuracy as close as to 100%; the highest average accuracy is approximately 96% in the case of our 3-dimensional dataset. When we add the other two features of interest to the 3-dimensional dataset, the highest accuracy becomes lower (around 85% - 90%).

4.3.2. The influence of a and the bandwidth parameter on the accuracy

Let us incorporate the tuning parameter a as well when estimating $\widehat{f}_n$. Here, we let $a_{\text{Benign}} = a_{\text{Malignant}}$ as opposed to the bandwidth parameter where h_{Benign} and $h_{\text{Malignant}}$ may not be equal when we vary them. So instead of using the notation a_{Benign} or a_{Benign}, we denote the tuning parameter per usual (i.e. the parameter a). To cover the large a as well, we choose $a = 2, 5, 10, 30, 50, 100$. Also, only the 3- and 5-dimensional dataset are investigated, since the parameter a will not influence the estimated generator (see Section 2.3).

We start with the 3-dimensional dataset (see Figure 4.5). Visually, the area where the average accuracy is the lowest, is smaller than when $a = 0$. Surprisingly, the highest average accuracy is still around 95% - 100% when we increase a. What is clear from this figure, is how the dark blue area start to appears around $a = 30$. It seems as if this area pushes the high accuracy area to the left hand side.

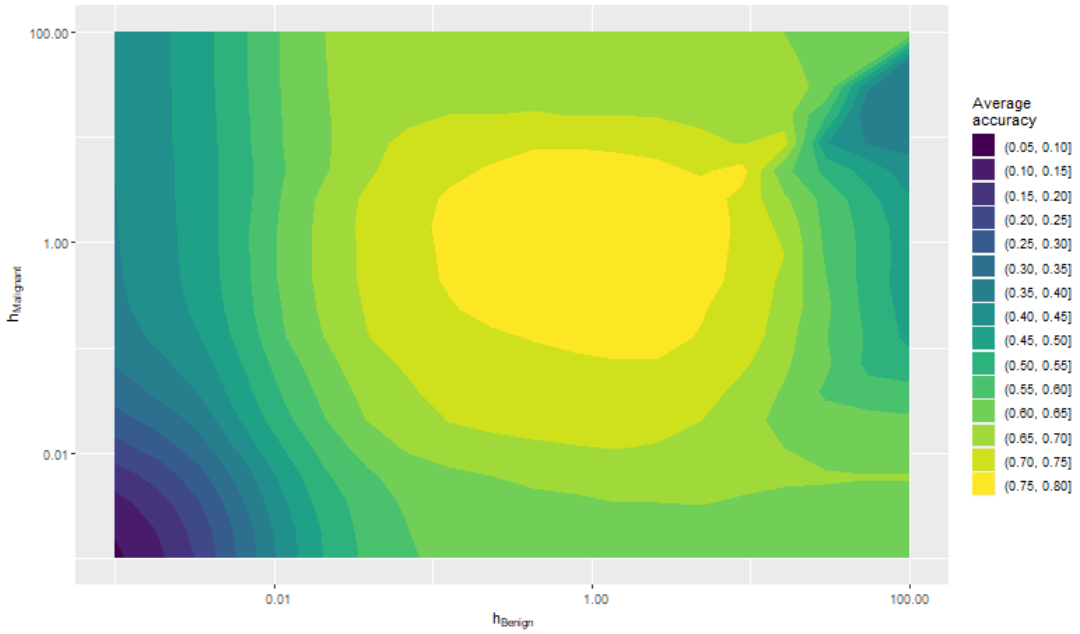


Figure 4.2: The contour plot of the average accuracy of the classifier using the dataset with column worst texture and worst smoothness (2-dimensional dataset).

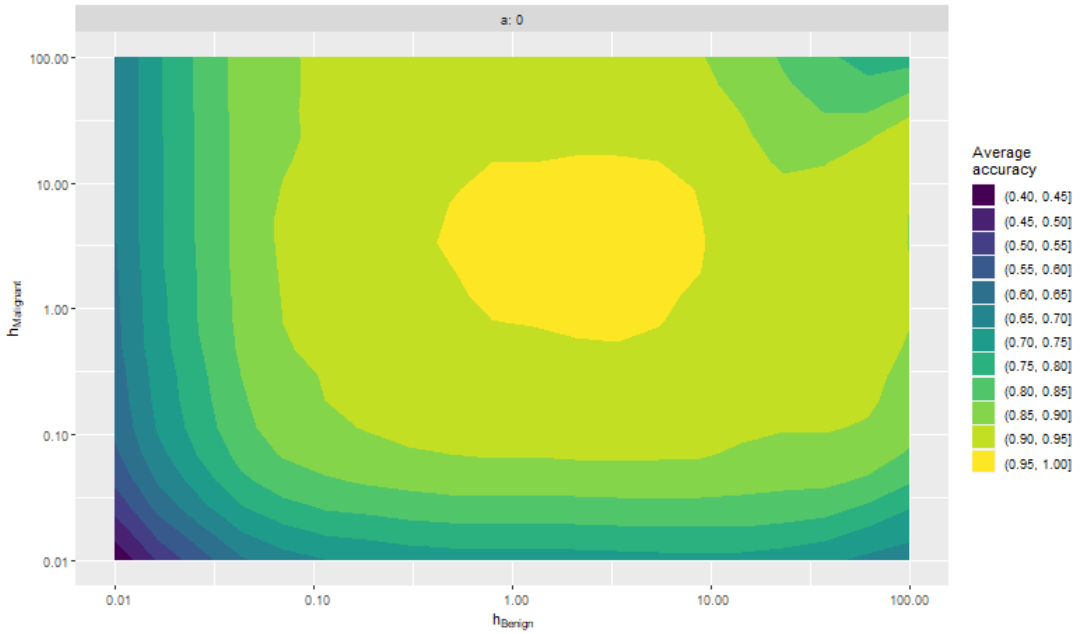


Figure 4.3: The contour plot of the average accuracy of the classifier using the dataset with column texture, smoothness and area of the worst nuclei (3-dimensional dataset).

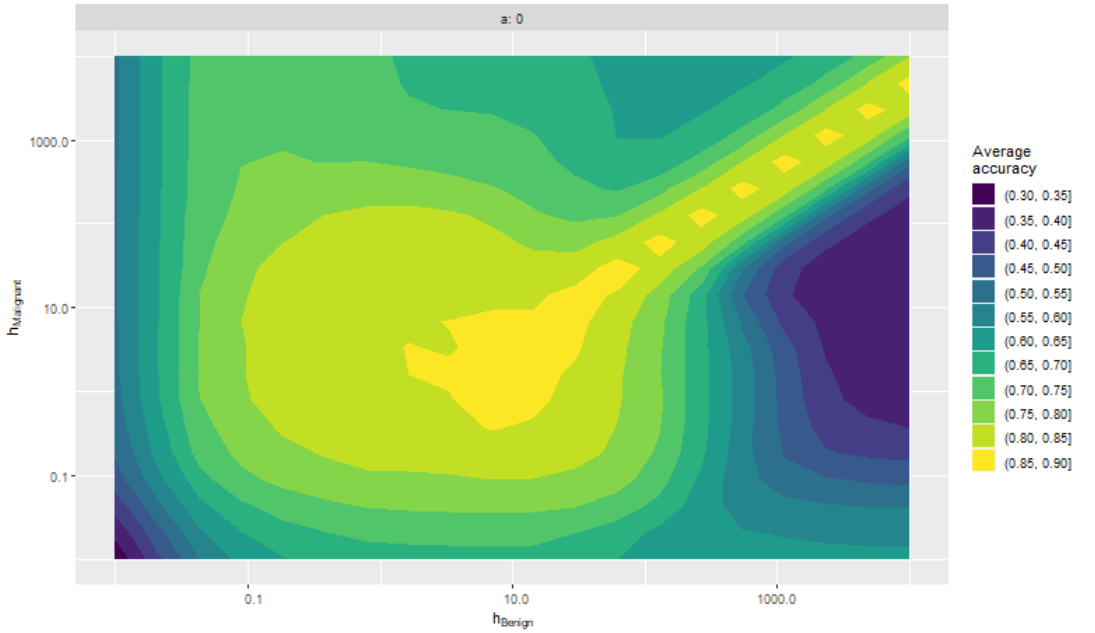


Figure 4.4: The contour plot of the average accuracy of the classifier using the dataset with the additional column compactness and fractal dimension of the worst nuclei (5-dimensional dataset).

Which means that a smaller h is required for the high accuracy, if a is increased.

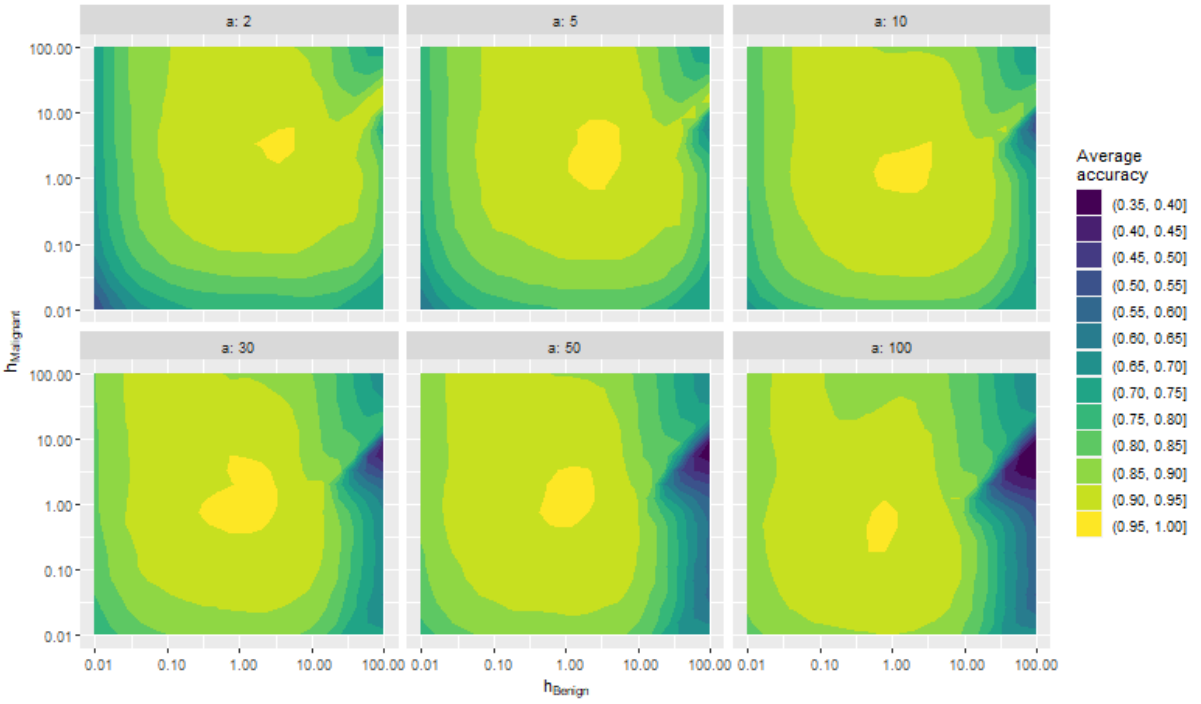


Figure 4.5: The average accuracy of the classifier using the dataset with column worst texture, worst smoothness and worst area (3-dimensional dataset). We choose $a = 2, 5, 10, 30, 50, 100$.

This same behaviour can be seen when we consider the 5-dimensional dataset (see Figure 4.6). But here, the area where the average accuracy is the highest change from 85% - 90% to 80% - 85%. Though from this figure we can not tell whether the accuracy is lower or that the high accuracy area

becomes very small.

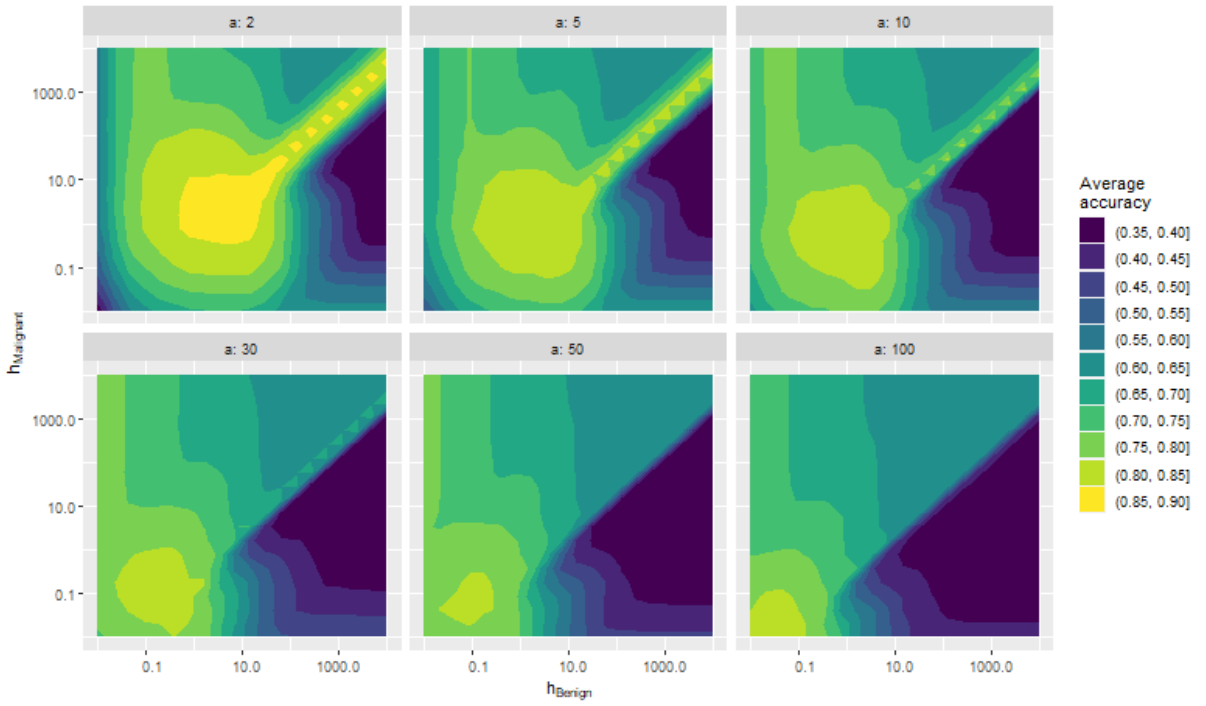


Figure 4.6: The average accuracy of the classifier using the dataset with with the additional column compactness and fractal dimension of the worst nuclei (5-dimensional dataset). The same a is chosen as in Figure 4.5.

Note here that both h_{Benign} and $h_{\text{Malignant}}$ range from 0.01 to 10000, but for the 3-dimensional dataset case they range from 0.01 to 1000. We do this because in Figure 4.4 and 4.6 ($a = 2$), there are smaller areas where the average accuracy is at the highest. And it seems that those areas lie on the line $h_{\text{Malignant}} = h_{\text{Benign}}$. We wanted to know whether that area will turn dark blue as we increase the bandwidth parameter. But that does not seem the case, instead, those smaller yellow areas start to dissappear as we increase a.

Additionally, we observe that as a is increased, the average accuracy does not change by a lot. This might be explained in the following way. Recall that we have to compute $y := (\mathbf{y} - \widehat{\boldsymbol{\mu}}_n)^T \widehat{\Sigma}_n^{-1} (\mathbf{y} - \widehat{\boldsymbol{\mu}}_n)$ to be able to apply KDE for one-dimensional dataset. This is then used to estimate the generator and it is subsequently used to estimate the density function of the dataset. Since the average accuracy does not change drastically, it might means that there are too few data points around the mean in the test set. If the influence of a is investigated, we would need more points around the centre of the data points.

There are other results where we show the sensitivity rate and specificity rate of the classifier. For these results, the whole Wisconsin dataset serves as both the train and test set. If the reader wishes to see how the these two rates change when the parameters in Liebscher's estimator vary, we refer to Appendix C.

5

Conclusion

In this thesis, we use Liebscher's estimator to estimate the generator of some elliptical distribution. We study the estimator by varying the bandwidth and the tuning parameter, and measure its performance. At first, we are interested in the particular case where the simulated data is sampled from the standard multivariate normal distribution.

From the simulations results, we conclude that the parameter a plays a smaller role in influencing the performance of the estimator as the dimension of the dataset becomes large. However, the importance of choosing h is high and independent from the dimension, i.e. choosing h is less flexible than choosing a. This makes an estimation of the generator become easier when we have a high-dimensional data, because we only need to make sure that the bandwidth parameter is correctly chosen.

Interestingly, we have seen how the logarithmic error linearly decreases with the sample size for different choice of dimension. It appears that, for a sufficiently large sample size, we can use Stute and Werner's estimator to get a performance as good as Liebscher's estimator. Unfortunately, there is a trade-off with computational time for high-dimensional data. We have seen that the computational time increases linearly with the dimension and the sample size.

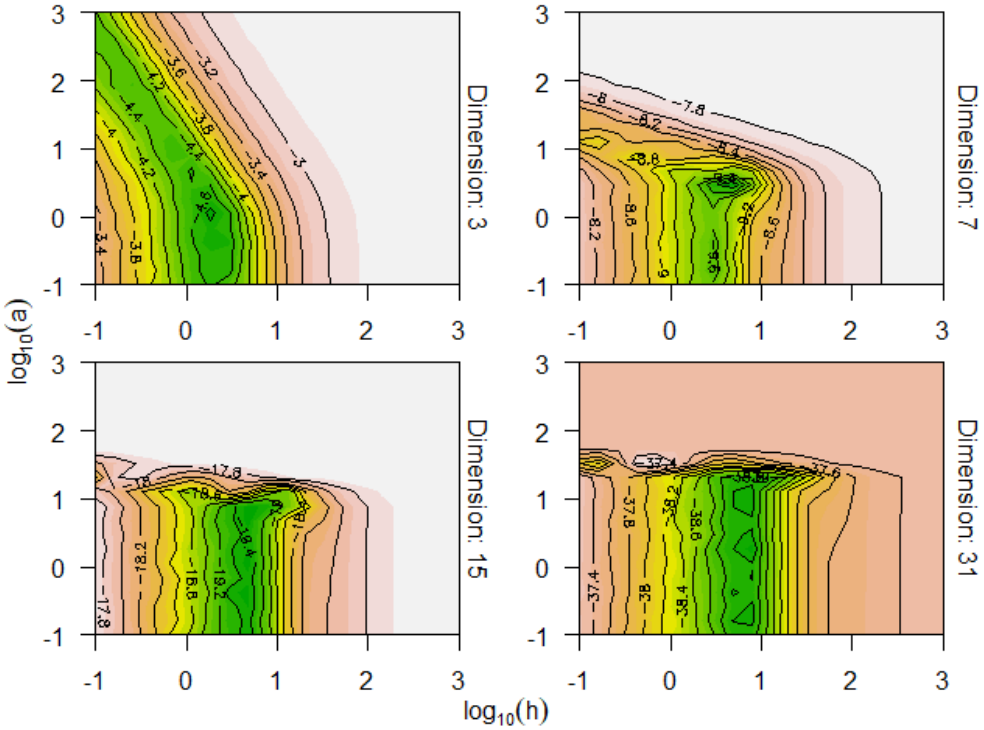
Next, we simulate datasets that are sampled from elliptical distributions with other generators. We first observe that Stute and Werner's estimator has indeed difficulty in estimating the points around the boundary, even when the sample size is large. Though, the estimator performs quite well away from the boundary, even when the generator has multiple bumps. In this part of simulation studies, we further learn that the robust version of Silverman's rule of thumb method is not an ideal bandwidth selection. It depends on the generator and the sample size, which is why we find this method unreliable. So, we do not recommend the use of this bandwidth selection method in general practice, which is also what [17] has mentioned.

Further, we conclude that finding the optimal a and h is a hard problem. This is because, for small-dimensional data, the location of the area where the error is the smallest differs for each generator. Though we did not investigate the high-dimensional case for different generator, we conjecture that the previous statement is also true for larger dimension.

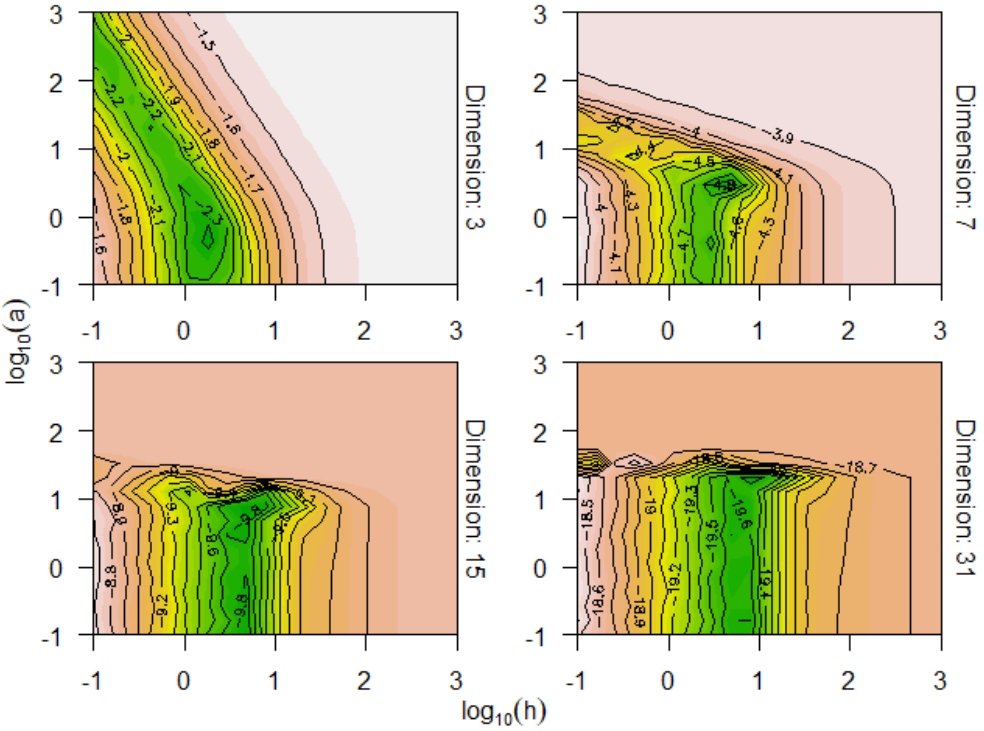
Lastly, we study an application of these non-parametric estimation techniques to a real life classification problem. We use a Bayes' classifier based on the estimated posterior probabilities. In this study, we find that the parameter a has smaller role in influencing the accuracy of the classifier. However, it seems to influence the choice of the bandwidth parameter. For a larger a, we might want to choose smaller h for a better accuracy.

A

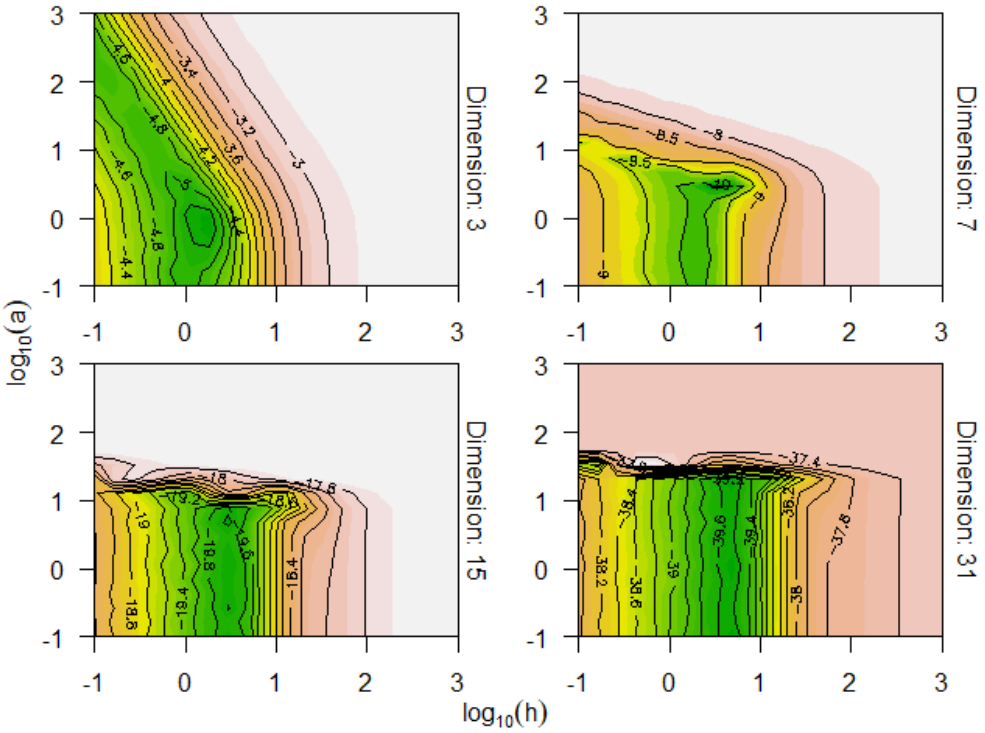
Extra Plots from The Simulation Studies



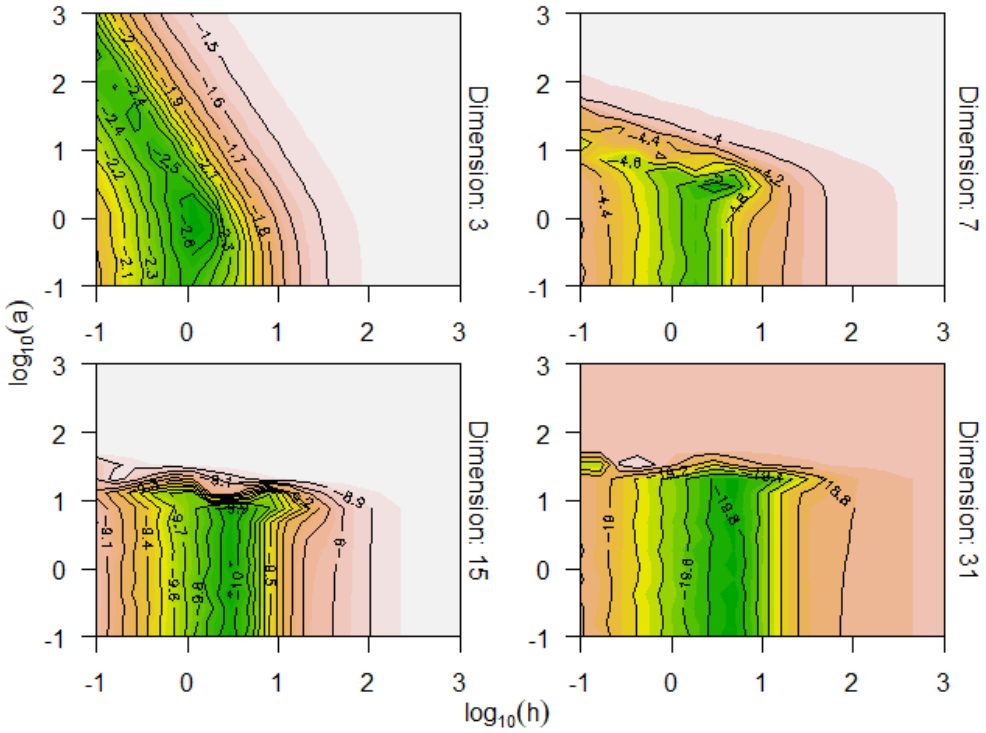
(a) $L^2 : n = 100$



(b) L^{∞} : $n = 100$

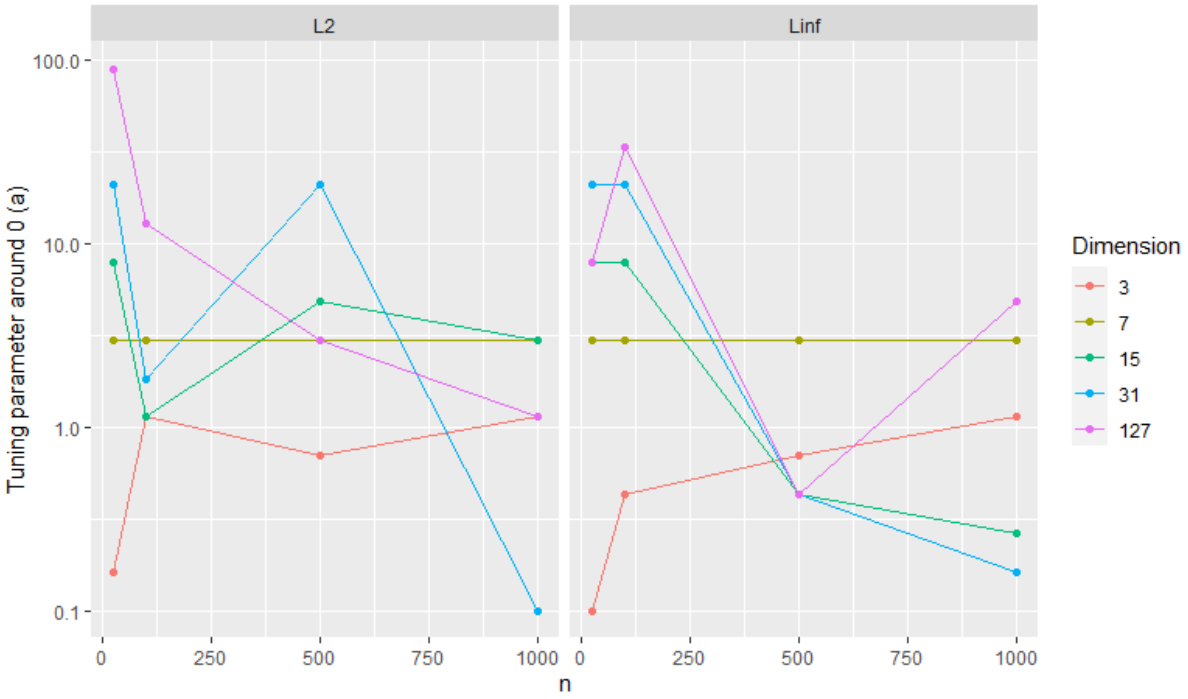


(c) L^2 : $n = 500$

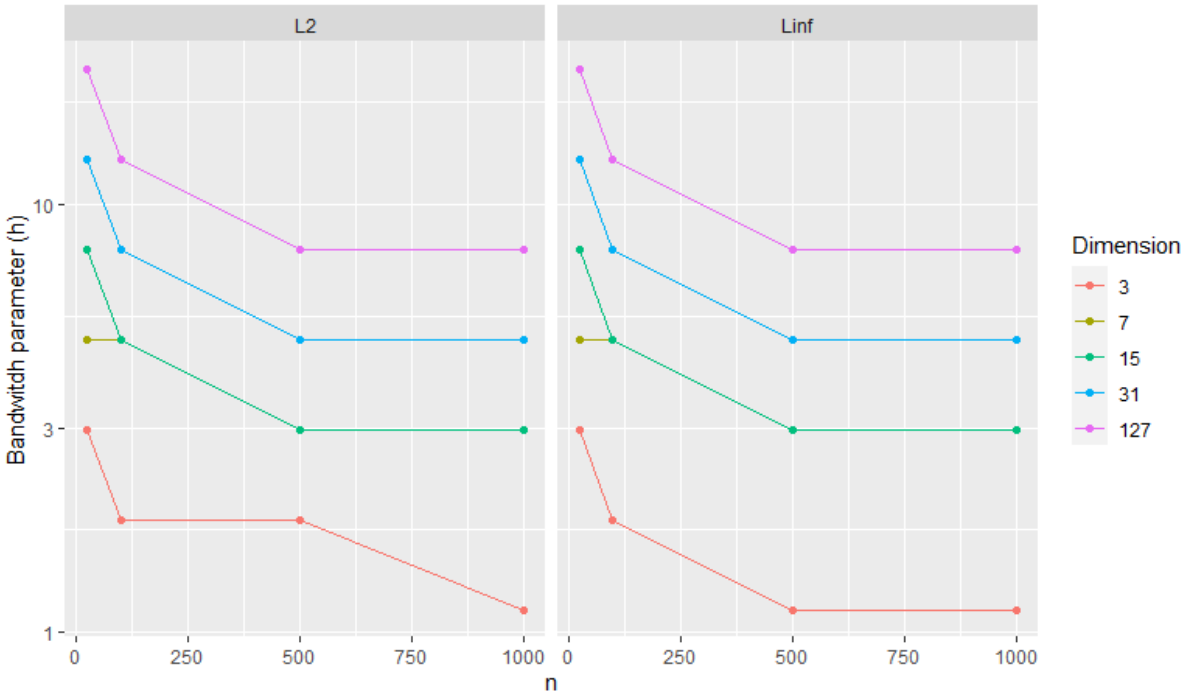


(d) L^{∞} : $n = 500$

Figure A.1: Contour plots of $\widehat{\text{MISE}}$ and $\widehat{\text{MSAE}}$ as a function of (a, h). The dimensions that are shown in each sub-figure are (from left to right) 3, 7, 15 and 31. In (a) and (b), the sample size is 100. As for (c) and (d), we choose $n = 500$. Darker green colour means that $\widehat{\text{MISE}}$ (or $\widehat{\text{MSAE}}$) is smaller than the colour that are lighter.



(a) The values of 'best' a as a function of n.



(b) The values of 'best' h as a function of n.

Figure A.2: The visualization of how the parameters a and h changes when $n \to \infty$. The points represents the 'best' a (in (a)) or h (in (b)) obtained from $\widehat{\text{MISE}}$ (left) and $\widehat{\text{MSAE}}$ (right).

B

Comparing The Rate of Decrease of $\log_{10}(\widehat{\text{MISE}})$

In Section 3.2.4, we would like to know the relationship between the error and the sample size. The error is obtained by two different estimators: one with $a = 0$ and $h = h_{\text{rot}}$ (i.e. 'a0 & S') and the other one is the 'best' a and h (best a & h). In Figure 3.7, we see that the data points fit quite well with the fitted line. However, we cannot see from the figures fast the error decreases for each estimator. In this appendix, we will answer this question by applying ordinary linear regression.

For each dimension $d = 3, 7, 15, 31, 63$ and 127, we use the following linear model.

$$\log_{10}(\widehat{\text{MISE}}) = \alpha_d + \beta_d \cdot \log_{10}(n) + \gamma_d \text{Parameter Choice} + \delta_d \log_{10}(n) \cdot \text{Parameter Choice} + \varepsilon, \quad \text{(B.1)}$$

where ε is the residual. The variable 'Parameter Choice' is a dummy variable with two factors defined in the following manner:

$$\text{Parameter Choice} = \begin{cases} 1, \text{ if the } i\text{-th observation belong to 'best } a \text{ \& } h\text{'}, \\ 0, \text{ otherwise.} \end{cases} \quad \text{(B.2)}$$

This means that we set the factor 'a0 & S' as the baseline factor. Further, the interaction term $\log_{10}(n) \cdot$ Parameter Choice is included in the model, because the slope of the line seems to be different for different parameter choice (see Figure 3.7).

Using Equation (B.1), (B.2) and least-squares method, we get two linear models for each d. Those are

$$\text{a0 \& S} : \log_{10}(\widehat{\text{MISE}}) = \widehat{\alpha}_d + \widehat{\beta}_d \cdot \log_{10}(n),$$
$$\text{best a \& h} : \log_{10}(\widehat{\text{MISE}}) = (\widehat{\alpha}_d + \widehat{\gamma}_d) + (\widehat{\beta}_d + \widehat{\delta}_d) \cdot \log_{10}(n),$$

where $\widehat{\alpha}_d, \widehat{\beta}_d, \widehat{\gamma}_d$ and $\widehat{\delta}_d$ are the estimated coefficients in (B.1). For 'a0 & S', the intercept and the slope are $\widehat{\alpha}_d$ and $\widehat{\beta}_d$ respectively. As for the factor 'best a & h', the terms $\widehat{\alpha}_d + \widehat{\gamma}_d$ and $\widehat{\beta}_d + \widehat{\delta}_d$ are the intercept and the slope respectively.

The estimated coefficients are summarised in Table B.2 - B.7, which is done by using the stargazer package [14]. The values in parenthesis underneath the estimated coefficients are the standard error of the estimation. Using the estimated coefficients in these tables, we can then compute the slopes and the intercept of each line in Figure 3.7 (see Table B.1).

From Table B.1, we see that the slope with parameter choice 'a0 & S' is steeper than 'best a & h'. The rate in which the error decreases seems to be faster in 'a0 & S' in comparison to 'best a & h' (e.g. $d = 31, 63$). In Table B.2 - Table B.7, we see that the estimated coefficient for the interaction term is statistically significant for $d = 3, 31, 63$ and 127.

Dimension	Parameter Choice	Intercept	Slope
3	a0 & S	−2.715	−0.830
	best a & h	−3.371	−0.748
7	a0 & S	−7.737	−0.794
	best a & h	−8.709	−0.683
15	a0 & S	−17.595	−0.786
	best a & h	−18.475	−0.585
31	a0 & S	−37.123	−0.875
	best a & h	−38.348	−0.528
63	a0 & S	−76.732	−0.788
	best a & h	−78.837	−0.421
127	a0 & S	−155.615	−0.774
	best a & h	−156.395	−0.627

Table B.1: The slope and the intercept of each line in Figure 3.7, obtained by using ordinary linear regression.

	Dependent variable:
	$\log_{10}(\widehat{MISE})$
$\widehat{\beta}_d$	-0.830^{***}
	(0.012)
$\widehat{\gamma}_d$	-0.656^{***}
	(0.041)
$\widehat{\delta}_d$	0.082^{***}
	(0.017)
$\widehat{\alpha}_d$	-2.715^{***}
	(0.029)
Observations	8
R^2	1.000
Adjusted R^2	0.999
Residual Std. Error	0.015 (df = 4)
F Statistic	$3{,}400.052^{***}$ (df = 3; 4)
Note:	*p<0.1; **p<0.05; ***p<0.01

Table B.2: The estimated coefficients for $d = 3$

	Dependent variable:
	$\log_{10}(\widehat{MISE})$
$\widehat{\beta}_d$	-0.794^{***}
	(0.069)
$\widehat{\gamma}_d$	-0.972^{**}
	(0.232)
$\widehat{\delta}_d$	0.111
	(0.098)
$\widehat{\alpha}_d$	-7.737^{***}
	(0.164)
Observations	8
R^2	0.989
Adjusted R^2	0.981
Residual Std. Error	0.087 (df = 4)
F Statistic	121.747^{***} (df = 3; 4)
Note:	*p<0.1; **p<0.05; ***p<0.01

Table B.3: The estimated coefficients for $d = 7$

	Dependent variable:
	$\log_{10}(\widehat{MISE})$
$\widehat{\beta}_d$	-0.786^{***}
	(0.067)
$\widehat{\gamma}_d$	-0.880^{**}
	(0.222)
$\widehat{\delta}_d$	0.201
	(0.094)
$\widehat{\alpha}_d$	-17.595^{***}
	(0.157)
Observations	8
R^2	0.985
Adjusted R^2	0.974
Residual Std. Error	0.083 (df = 4)
F Statistic	89.259^{***} (df = 3; 4)
Note:	*p<0.1; **p<0.05; ***p<0.01

Table B.4: The estimated coefficients for $d = 15$

	Dependent variable:
	$\log_{10}(\widehat{MISE})$
$\widehat{\beta}_d$	-0.875^{***}
	(0.028)
$\widehat{\gamma}_d$	-1.225^{***}
	(0.092)
$\widehat{\delta}_d$	0.347^{***}
	(0.039)
$\widehat{\alpha}_d$	-37.123^{***}
	(0.065)
Observations	8
R^2	0.998
Adjusted R^2	0.996
Residual Std. Error	0.034 (df = 4)
F Statistic	560.913^{***} (df = 3; 4)
Note:	*p<0.1; **p<0.05; ***p<0.01

Table B.5: The estimated coefficients for $d = 31$

	Dependent variable:
	$\log_{10}(\widehat{MISE})$
$\widehat{\beta}_d$	-0.788^{***}
	(0.069)
$\widehat{\gamma}_d$	-2.105^{***}
	(0.229)
$\widehat{\delta}_d$	0.366^{**}
	(0.097)
$\widehat{\alpha}_d$	-76.733^{***}
	(0.162)
Observations	8
R^2	0.993
Adjusted R^2	0.989
Residual Std. Error	0.086 (df = 4)
F Statistic	202.950^{***} (df = 3; 4)
Note:	*p<0.1; **p<0.05; ***p<0.01

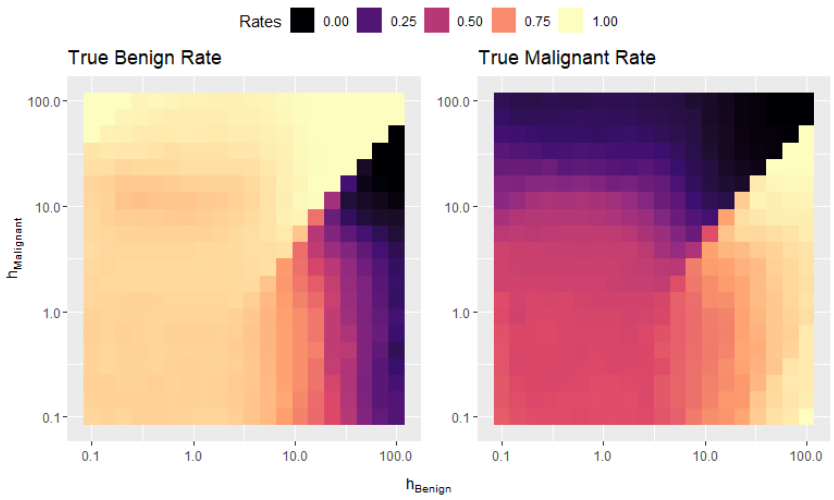
Table B.6: The estimated coefficients for $d = 63$

	Dependent variable:
	$\log_{10}(\widehat{MISE})$
$\widehat{\beta}_d$	-0.774^{***}
	(0.017)
$\widehat{\gamma}_d$	-0.780^{***}
	(0.056)
$\widehat{\delta}_d$	0.147^{***}
	(0.024)
$\widehat{\alpha}_d$	-155.615^{***}
	(0.040)
Observations	8
R^2	0.999
Adjusted R^2	0.998
Residual Std. Error	0.021 (df = 4)
F Statistic	$1{,}478.149^{***}$ (df = 3; 4)
Note:	*p<0.1; **p<0.05; ***p<0.01

Table B.7: The estimated coefficients for $d = 127$

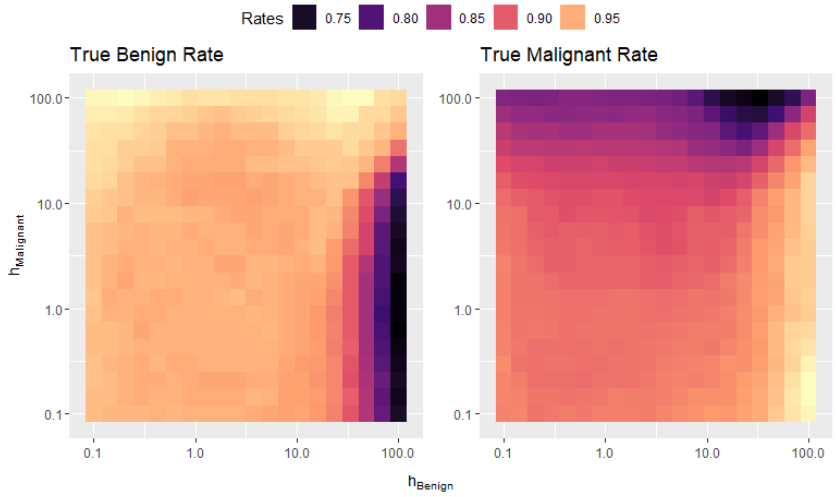
C

Extra Classification Results from The Wisconsin Dataset



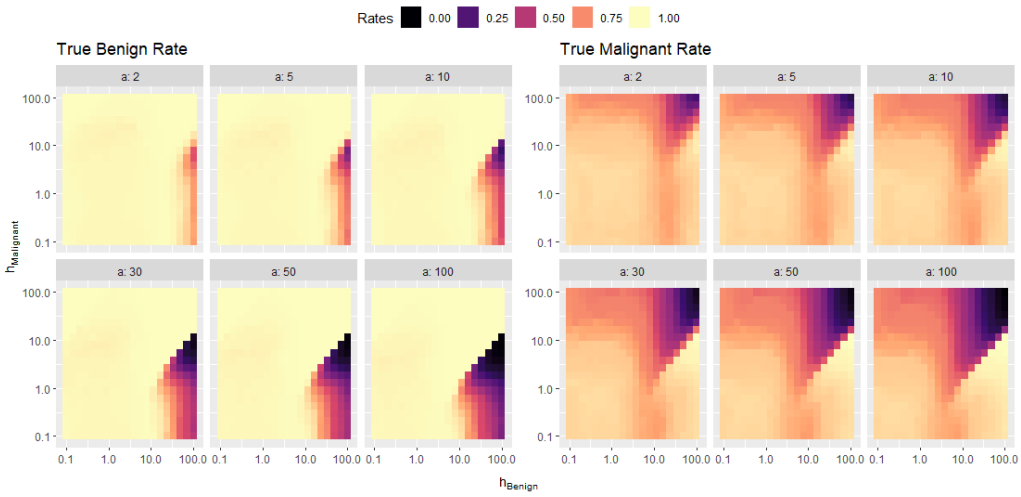
(a) The sensitivity rate (left) and the specificity (right) of the 2-dimensional dataset.

(b) The sensitivity rate (left) and the specificity (right) of the 3-dimensional dataset.

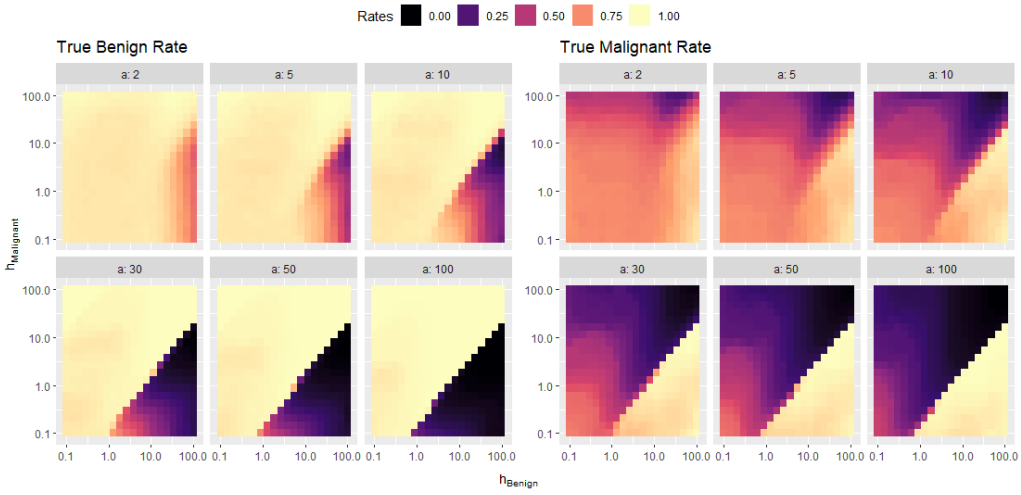


(c) The sensitivity rate (left) and the specificity (right) of the 5-dimensional dataset.

Figure C.1: The sensitivity rate (left) and the specificity (right) of the dataset using the characteristics of the worst nuclei of interest.



(a) The sensitivity rate (left) and the specificity (right) of the 3-dimensional dataset, with additional parameter a.



(b) The sensitivity rate (left) and the specificity (right) of the 5-dimensional dataset, with additional parameter a.

Figure C.2: The plots showing how the rates change when we vary both the bandwidth and the tuning parameter.

Bibliography

[1] Battey, H. and Linton, O. (2014). Nonparametric estimation of multivariate elliptic densities via finite mixture sieves. *Journal of Multivariate Analysis*, 123:43–67.

[2] Cambanis, S., Huang, S., and Simons, G. (1981). On the theory of elliptically contoured distributions. *Journal of Multivariate Analysis*, 11(3):368–385.

[3] Derumigny, A. and Fermanian, J.-D. (2022a). *ElliptCopulas: Inference of Elliptical Distributions and Copulas*. R package version 0.1.2. Available at `https://github.com/AlexisDerumigny/ElliptCopulas`.

[4] Derumigny, A. and Fermanian, J.-D. (2022b). Identifiability and estimation of meta-elliptical copula generators. *Journal of Multivariate Analysis*, 190:104962.

[5] Derumigny, A. and Schmidt-Hieber, J. (2020). On lower bounds for the bias-variance trade-off. *arXiv preprint arXiv:2006.00278*. Available at `https://alexisderumigny.wordpress.com/`.

[6] Dua, D. and Graff, C. (2017). *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences.

[7] Epanechnikov, V. A. (1969). Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, 14(1):153–158.

[8] Gómez, E., Gómez-Villegas, M. A., and Marín, J. M. (2003). A survey on continuous elliptical vector distributions. *Revista matemática complutense*, 16(1):345–361.

[9] Gray, A. G. and Moore, A. W. (2003). Nonparametric density estimation: Toward computational tractability. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, pages 203–211. SIAM.

[10] Guleria, P. and Sood, M. (2022). Artificial intelligence and machine learning for the healthcare sector: performing predictions and metrics evaluation of ml classifiers on a diabetic diseases data set. In *Cognitive and Soft Computing Techniques for the Analysis of Healthcare Data*, pages 1–28. Elsevier.

[11] Hall, P. and Marron, J. S. (1987). Extent to which least-squares cross-validation minimises integrated square error in nonparametric density estimation. *Probability Theory and Related Fields*, 74(4):567–581.

[12] Härdle, W., Hall, P., and Marron, J. S. (1988). How far are automatically chosen regression smoothing parameters from their optimum? *Journal of the American Statistical Association*, 83(401):86–95.

[13] Härdle, W., Müller, M., Sperlich, S., and Werwatz, A. (2004). *Nonparametric and semiparametric models*, volume 1. Springer.

[14] Hlavac, M. (2022). *stargazer: Well-Formatted Regression and Summary Statistics Tables*. Social Policy Institute, Bratislava, Slovakia. R package version 5.2.3. Available at `https://CRAN.R-project.org/package=stargazer`.

[15] Hwang, J.-N., Lay, S.-R., and Lippman, A. (1994). Nonparametric multivariate density estimation: a comparative study. *IEEE Transactions on Signal Processing*, 42(10):2795–2810.

[16] Johnson, M. E. (1987). *Multivariate statistical simulation: A guide to selecting and generating continuous multivariate distributions*, volume 192. John Wiley & Sons.

[17] Jones, M. C., Marron, J. S., and Sheather, S. J. (1996). A brief survey of bandwidth selection for density estimation. *Journal of the American statistical association*, 91(433):401–407.

[18] Köhler, M., Schindler, A., and Sperlich, S. (2014). A review and comparison of bandwidth selection methods for kernel regression. *International Statistical Review*, 82(2):243–274.

[19] Li, J. P., Haq, A. U., Din, S. U., Khan, J., Khan, A., and Saboor, A. (2020). Heart disease identification method using machine learning classification in e-healthcare. *IEEE Access*, 8:107562–107582.

[20] Liebscher, E. (2005). A semiparametric density estimator based on elliptical distributions. *Journal of multivariate analysis*, 92(1):205–225.

[21] Mishra, N. K. and Celebi, M. E. (2016). An overview of melanoma detection in dermoscopy images using image processing and machine learning. *arXiv preprint arXiv:1601.07843*.

[22] Muirhead, R. J. (2009). *Aspects of multivariate statistical theory*. John Wiley & Sons.

[23] Nadarajah, S. and Kotz, S. (2005). Mathematical properties of the multivariate t distribution. *Acta Applicandae Mathematica*, 89(1):53–84.

[24] Nagler, T. and Czado, C. (2016). Evading the curse of dimensionality in nonparametric density estimation with simplified vine copulas. *Journal of Multivariate Analysis*, 151:69–89.

[25] Park, B. U. and Marron, J. S. (1990). Comparison of data-driven bandwidth selectors. *Journal of the American Statistical Association*, 85(409):66–72.

[26] Parzen, E. (1962). On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076.

[27] R Core Team (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

[28] Rebala, G., Ravi, A., and Churiwala, S. (2019). *An introduction to machine learning*. Springer.

[29] Rosenblatt, M. (1956). Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832 – 837. `https://doi.org/10.1214/aoms/1177728190`.

[30] Schindler, A. (2012). Bandwidth selection in nonparametric kernel estimation.

[31] Schuster, E. F. (1985). Incorporating support constraints into nonparametric estimators of densities. *Communications in Statistics-Theory and methods*, 14(5):1123–1136.

[32] Sheather, S. J. and Jones, M. C. (1991). A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society: Series B (Methodological)*, 53(3):683–690.

[33] Sidey-Gibbons, J. A. and Sidey-Gibbons, C. J. (2019). Machine learning in medicine: a practical introduction. *BMC medical research methodology*, 19(1):1–18.

[34] Silverman, B. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis. https://books.google.nl/books?id=e-xsrjsL7WkC.

[35] Stone, C. J. (1984). An asymptotically optimal window selection rule for kernel density estimates. *The Annals of Statistics*, pages 1285–1297.

[36] Stute, W. and Werner, U. (1991). Nonparametric estimation of elliptically contoured densities. In *Nonparametric Functional Estimation and Related Topics*, pages 173–190. Springer.

[37] Tsybakov, A. B. (2009). Nonparametric estimators. In *Introduction to Nonparametric Estimation*, pages 1–76. Springer.