Institute of Parallel and Distributed Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Adaptive Kernel Density Estimation on Large Datasets

Jan-Oliver Schmidt

Course of Study:	Informatik
Examiner:	Jun.-Prof. Dr. rer. nat. Dirk Pflüger
Supervisor:	M.Sc.-Inf. Fabian Franzelin
Commenced:	February 7, 2017
Completed:	August 7, 2017
CR-Classification:	G.3

Abstract

In application fields such as data mining or uncertainty quantification very large, high dimensional datasets are rather common. While there are several different approaches to kernel density estimation, they all of them share the disadvantage of being very time consuming, when the sample set is really large.
In this thesis, we want to overcome this issue by sub-sampling the dataset and choosing an adaptive kernel density estimator, which is as easy and time-saving, while still being as accurate as possible. Apart from being directly compared to another proved efficient density estimation technique, the Rosenblatt transformation is adducted to serve as an application example.

Kurzfassung

In Anwendungsbereichen wie Data Mining oder bei Quantifizierung von Unsicherheiten, kommen häufig sehr große, hochdimensionale Datensätze vor. Es existieren einige verschiedene Herangehensweisen für Kerndichteschätzung, allerdings laufen alle auf das selbe Problem zu: bei zu großen Datensätzen ist die Schätzung sehr zeitaufwändig. In dieser Abschlussarbeit soll dieses Problem mit Hilfe eines sub-Sampling Verfahren und einem geeignetem Kerndichteschätzer, welcher möglichst zeitsparend und dennoch akkurat ist, überwunden werden. Neben der Evaluierung des vorgestellten Verfahren anhand eines anderem etablierten Dichteschätzer, wird auch noch die Rosenblatt Transformation als ein Anwendungsbeispiel herangezogen.

Contents

1 Introduction 11

2 Theoretical Background 13
2.1 Density Estimation . . . 13
2.2 Log-likelihood . . . 13
2.3 Cross Validation . . . 14

3 Kernel Density Estimation 17
3.1 Kernel Functions . . . 18
3.2 Fixed Bandwidth KDE . . . 19
3.3 Motivation for adaptive KDE . . . 20
3.4 *Balloon* Estimator . . . 21
3.5 *Sample Point* Estimator . . . 21
3.6 Bandwidth selection . . . 22
3.7 Fixed Bandwidth Selection . . . 22
3.8 Adaptive Bandwidth Selection . . . 23
3.9 Rosenblatt Transformation . . . 24

4 Probability density estimation on large data sets with kernel-based methods 27
4.1 Used Adaptive KDE . . . 27
4.2 Sub-sampling . . . 28

5 Experiments 31
5.1 Kernel choice . . . 31
5.2 Comparison with Benjamin Peherstorfers' *Density Estimation with Adaptive Sparse Grids* . . . 33
5.3 Rosenblatt Transformation . . . 38

6 Summary and Prospects 41

Bibliography 43

List of Figures

3.1 Mixture of six kernel functions, located at some sample points. 17
3.2 Gaussian kernel. 19
3.3 Epanechnikov kernel. 19
3.4 Influence of the bandwidth on a fixed KDE, using a Gaussian kernel function. 22

5.1 2-5-beta distribution, approximated by a fixed KDE using cross validation. 32
5.2 2-5-beta distribution, approximated by an adaptive KDE using cross validation. 32
5.3 Spheres - dataset, KDEs with Gaussian kernel. 33
5.4 Spheres - dataset, KDEs with Epanechnikov kernel. 33
5.5 DR6 dataset. Log-likelihood of different KDEs against size of sub-sampled training set. 35
5.6 Checker dataset. Log-likelihood of different KDEs against size of sub-sampled training set. 36
5.7 Svmguide1 dataset. Log-likelihood of different KDEs against size of sub-sampled training set. 37
5.8 Liver-disorder. Log-likelihood of different KDEs against size of sub-sampled training set. 38
5.9 Two Moons dataset, before the Rosenblatt transformation. 39
5.10 Two Moons dataset, after the Rosenblatt transformation. 39

List of Tables

5.1 Log-likelihood values for the adaptive KDE, using Gaussian and Epanechnikov kernel functions, on different datasets. 32
5.2 Log-likelihood values for the different estimates on distinct datasets, used by Peherstorfer in [PPB14]. The log-likelihood values for SGDE and numbers of grid points (#GP) are also taken over from Peherstorfer. . . . 34

1 Introduction

Apart from estimating probability densities, density estimators also find usage in other application fields, such as classification (e.g. [Mil11]) or clustering (e.g. [PPB12]). This work, however, limits itself to probability density estimates only.

The goal is to utilize adaptive kernel based methods for estimation densities, using possibly very large and high dimensional datasets. While kernel density estimators are very popular, they do not scale very well with the size of the dataset, since the time-complexity of evaluating the estimate at any given point lies in $\mathcal{O}(n)$, when $n \in \mathbb{N}$ is the size of the dataset. On the other hand, kernel density estimators are not inherently dependant on the dimensionality of the dataset.

Another state-of-the-art density estimation method, *Sparse Grids Density Estimation* (SGDE) (see e.g. [PPB14]), is only negligible dependant on the datasets' size and therefore more appealing than a standard kernel density estimator, when dealing with large datasets. However, while a large dataset does not matter too much for SGDE, the dimensionality of the dataset heavily affects the runtime.

Hence, for a kernel density estimator to be competitive with SGDE, when dealing with large datasets some kind of sub-sampling could do the trick. In this work we want to explore, whether the efficiency-accuracy trade-off is worth it or not.

This thesis is structured as follows: After clarifying some basics in chapter 2, the concept of kernel density estimation is introduced in chapter 3, as well as various different KDE approaches, ways to optimize the estimates and the description of how the Rosenblatt transformation can be performed, using an adaptive kernel density estimator of a certain type. In chapter 4 we propose how adaptive kernel density estimation can be efficiently used on large datasets, by sub-sampling the data. The experiments, that were realized in the context of this thesis, are discussed in chapter 5, together with the results. Lastly this work is summarized in chapter 6.

2 Theoretical Background

This chapter deals with concepts and methods, which are utilized in this thesis. First the core problem, density estimation, is introduced, followed by two popular methods, that play a role in optimization of density estimators.

2.1 Density Estimation

Let $(\Xi, \mathcal{F}, P)$ be a complete probability space, with sample space Ξ, σ-algebra $\mathcal{F}$ and probability measure $P : \mathcal{F} \to [0, 1]$ (for further details, see [CKS08]). Furthermore, let's consider a random vector $\boldsymbol{\xi}$, its corresponding cumulative distribution function (CDF) F and probability density function (PDF) f and set of samples $\mathcal{S} = \{\xi^{(1)}, \ldots, \xi^{(n)}\}$, drawn from ξ according to f.
Density estimation is the construction of a density function $\hat{f}$, while utilizing the sample set $\mathcal{S}$, such that $\hat{f} \approx f$.

2.2 Log-likelihood

When being tasked to compare different density estimates of the same underlying density function or, by extension, to optimize a density estimator which depends on some parameter $\mathcal{P}$, some metric is required to gauge the quality of the estimated density function $\hat{f}$. The *Kullback-Leibler divergence* D_{KL} describes how similar two density functions f and $\hat{f}$ are, whereby $D_{\mathrm{KL}}(f, \hat{f})$ being zero, indicates that f and $\hat{f}$ are very similar and higher values raise suspicion, that the distributions are rather different.

$$D_{\mathrm{KL}}(f, \hat{f}) := \int_{\mathbb{R}} f(\xi) \log\left(\frac{f(\xi)}{\hat{f}(\xi)}\right) d\xi. \tag{2.1}$$

The *Kullback-Leibler divergence* would be a great metric for the purpose of density estimation, though it can often not be directly computed, for the lack of the real density function f. But i is possible to derive a usable metric from the *Kullback-Leibler divergence*.

Let's assume two density estimates of f, $\hat{f}^{(1)}$ and $\hat{f}^{(2)}$ and $D_{\mathrm{KL}}(f, \hat{f}^{(1)}) < D_{\mathrm{KL}}(f, \hat{f}^{(2)})$:

$$\begin{aligned} D_{\mathrm{KL}}(f, \hat{f}^{(1)}) &< D_{\mathrm{KL}}(f, \hat{f}^{(2)}) \\ \int_{\mathbb{R}} f(\xi) \log\Big(\frac{f(\xi)}{\hat{f}^{(1)}(\xi)}\Big) d\xi &< \int_{\mathbb{R}} f(\xi) \log\Big(\frac{f(\xi)}{\hat{f}^{(2)}(\xi)}\Big) d\xi \\ \int_{\mathbb{R}} f(\xi) \log f(\xi) d\xi - \int_{\mathbb{R}} f(\xi) \log \hat{f}^{(1)}(\xi) d\xi &< \int_{\mathbb{R}} f(\xi) \log f(\xi) d\xi - \int_{\mathbb{R}} f(\xi) \log \hat{f}^{(2)}(\xi) d\xi \\ \int_{\mathbb{R}} f(\xi) \log \hat{f}^{(1)}(\xi) d\xi &> \int_{\mathbb{R}} f(\xi) \log \hat{f}^{(2)}(\xi) d\xi \end{aligned} \tag{2.2}$$

Let $\mathcal{T} := \{\xi^{(1)}, \ldots, \xi^{(m)}\}$ be a set of samples that is drawn according to the true density f. Then the above integrals can be approximated as

$$\int_{\mathbb{R}} f(\xi) \log \hat{f}(\xi) d\xi \approx \frac{1}{m} \sum_{i=1}^{m} \log\Big(\hat{f}(\xi^{(i)})\Big), \tag{2.3}$$

this is also known as the approximation of the *log-likelihood* $\mathcal{L}_{\mathcal{T}}(\hat{f})$. Combining the knowledge from 2.2 and 2.3, we can expect $\mathcal{L}_{\mathcal{T}}(\hat{f}^{(1)})$ to be greater than $\mathcal{L}_{\mathcal{T}}(\hat{f}^{(2)})$. Thus making $\mathcal{L}_{\mathcal{T}}(\hat{f})$ a suitable metric for density estimation.

Since we want to compare the results form chapter 5 with the ones from [PPB14], who used the *log-likelihood*, we shall use that as our evaluation method as well.

2.3 Cross Validation

Let f be a probability density function, F the corresponding CDF and $\mathcal{S} := \{\xi^{(1)}, \ldots, \xi^{(n)}\}$ a set of samples, which are drawn according to f. Furthermore, let $\hat{f}$ be an estimate of f, which is obtained from using $\mathcal{S}$ as training-sample set.

If $\hat{f}$ depends, apart from the training-samples, on some parameters $\mathcal{P}$, it is not necessarily clear how to optimize $\mathcal{P}$ such that $\hat{f}$ approximates f best. A very commonly used method, that yields good results, is the cross validation (see e.g. [EM12]). There are different variations of this method, e.g. maximum-likelihood cross validation.

$$CV(\mathcal{P}) = \frac{1}{n} \sum_{i=1}^{n} \log(\hat{f}_{\mathcal{P},-i}(\xi^{(i)})), \tag{2.4}$$

with $\hat{f}_{P,-i}$ being a so called *leave-one-out* estimator. $\hat{f}_{P,-i}$ estimates f, utilizing the reduced training sample set $\mathcal{S} \setminus \{\xi_i\}$ and is dependent on $\mathcal{P}$. Choosing $\mathcal{P}$ so that it maximizes $CV(\mathcal{P})$, optimizes $\hat{f}$.

The time complexity of computing 2.4 lies in $\mathcal{O}(n)$, doing this for each training sample, it is already $\mathcal{O}(n^2)$. Since constructing $\hat{f}_{\mathcal{P},-i}$ might be in $\mathcal{O}(n)$ or higher, which would result in at least $\mathcal{O}(n^3)$. Therefore it is not feasible to use the *leave-one-out* for larger training-sample sets. In this case it is much more appropriate to use the *k*-fold maximum-likelihood cross validation (see [HPK11, p. 113]), which requires the training sample set to be segmented into *k* different subsets $\mathcal{S}_i$, with $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$ and evaluate $CV(\mathcal{P})$ analogously to the maximum-likelihood cross validation, but this time there are *k* iterations during each one of the training subsets is used as the testing set to evaluate the estimate, which is obtained by using all the training subsets except the one used for the testing.

3 Kernel Density Estimation

Kernel density estimation (KDE) is somewhat reminiscent of histograms, for the density estimate at a certain data point is mainly influenced by the sample point in its' near proximity. Though in contrast to histograms, these sample points do not necessarily share the same influence over the estimate as the others. Essentially, a kernel density estimator is just a summation of multiple shifted kernel functions (see 3.1).

KDE offer mainly two very notable advantages over histogram based methods. First, KDEs generally do not get palpably influenced by the dimensionality of the sample set and second, they do produce a smooth probability density function. It is also noteworthy that, using an appropriate kernel will lead to an easy calculation of derivatives or the commutative distribution function.

In the following sections, we will describe some commonly used kernel function, that are utilized in the field of kernel density estimation. Furthermore we will present some of the various KDE types and how they can be optimized. Lastly the established Rosenblatt

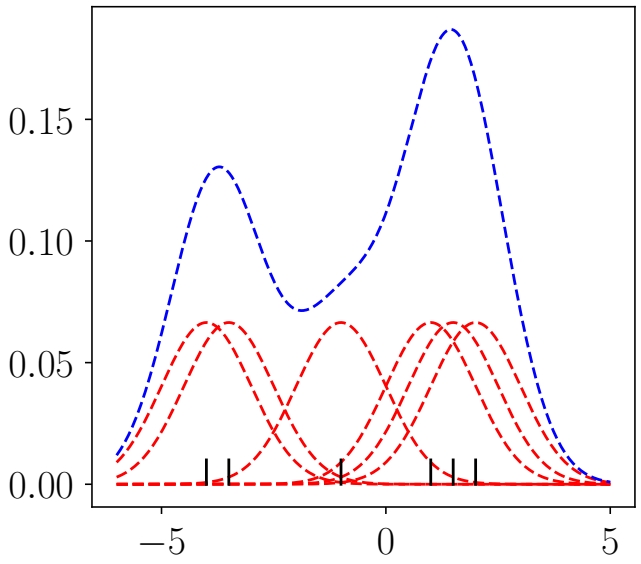

Figure 3.1: Mixture of six kernel functions, located at some sample points.

transformation is introduced, together with a method to conduct this transformation, using an adaptive kernel density estimator.

3.1 Kernel Functions

A smooth kernel function κ, that is to be used for a kernel density estimation, generally has to satisfy the following requirements:

$$\begin{aligned} &\forall \xi : \kappa(\xi) \geq 0 && \int_{\mathbb{R}} \kappa(\xi)\, d\xi = 1 \\ &\int_{\mathbb{R}} \xi\, \kappa(\xi)\, d\xi = 0 && \int_{\mathbb{R}} \xi^2\, \kappa(\xi)\, d\xi < \infty \end{aligned} \quad . \tag{3.1}$$

It is generally excepted, that the choice of the kernel is not of utmost importance (see e.g. [PPB14, p. 1]) and an appropriate selection of the bandwidth(s) is much more critical. This claim is ascertained in chapters 3.6 and 5.1.

3.1.1 Gaussian Kernel

The Gaussian kernel is the most popular among the kernel functions and finds usage in all sorts of application fields and is commonly used for kernel density estimation (see e.g. [Sai94] or [Mil11]).

The Gaussian kernel function is defined as

$$\kappa(\xi) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\xi^2}, \tag{3.2}$$

Derivatives and integrals of this function are easy to compute, which implies the same for derivatives and the CDF of the estimate $\hat{f}$.

3.1.2 Epanechnikov Kernel

The Epanechnikov kernel is, like the Gaussian, well known and does also find utilisation in kernel density estimation (see [EM12])

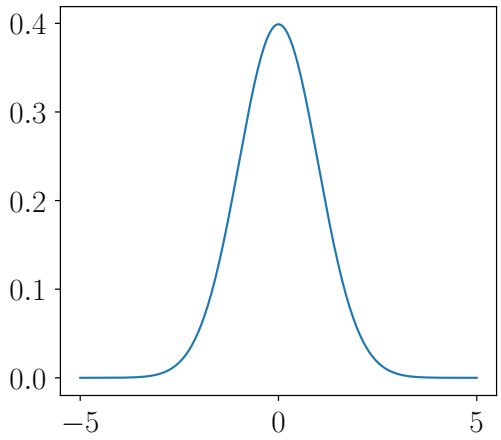

Figure 3.2: Gaussian kernel.

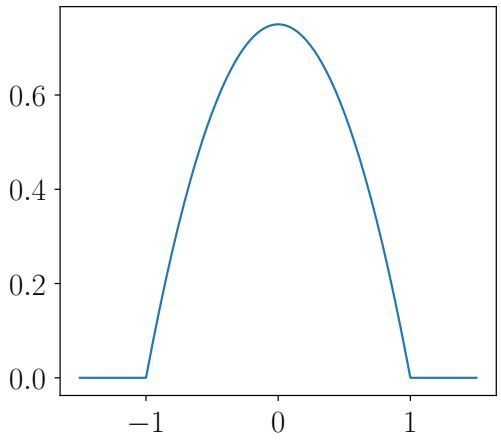

Figure 3.3: Epanechnikov kernel.

The Epanechnikov kernel function is defined as:

$$\kappa(\xi) = \begin{cases} \frac{3}{4}(1-\xi^2) & \text{if } |\xi| \leq 1 \\ 0 & \text{else} \end{cases}, \tag{3.3}$$

When using the Rosenblatt transformation with (adaptive) kernel density estimation on a dataset, where the samples were drawn from a bounded sample space, the Epanechnikov kernel with its limited support has the edge over the Gaussian kernel, since it can be constructed that no values outside of these bounds are processed by the kernel density estimator.

3.2 Fixed Bandwidth KDE

Kernel density estimates possess a smoothing parameter h, which is also called the bandwidth of the kernels. Fixed bandwidth KDE, or just fixed KDE, describes a subset of kernel density estimators, where the kernels share the same bandwidth(s). The simple one dimensional fixed kernel density estimator with general weights is defined as:

$$\hat{f}(\xi) := \frac{1}{h} \sum_{i=1}^{n} c_i \, \kappa\left(\frac{\xi^{(i)} - \xi}{h}\right), \tag{3.4}$$

with $0 \leq c_i \leq 1$, $\sum_{i=1}^{n} c_i = 1$ and $h \in \mathbb{R}$.

While this method surely is viable, it seems that most people prefer to use uniformly weighted kernels, i.e. $c_i = \frac{1}{n}$, due to it's simplicity.

$$\hat{f}(\xi) := \frac{1}{nh} \sum_{i=1}^{n} \kappa\left(\frac{\xi^{(i)} - \xi}{h}\right), \tag{3.5}$$

When being confronted with an multidimensional sample set, one has to choose just one bandwidth for all dimensions or one for each dimension. Obviously it is more time consuming to optimize the bandwidths in the former case, but the KDE does adapt more to the data.

When using a multiplicative kernel κ',

$$\kappa'(\xi_1, \xi_2, \ldots, \xi_d) = \kappa(\xi_1) \ldots \kappa(\xi_d), \tag{3.6}$$

the multidimensional KDE can be computed as:

$$\hat{f}(\boldsymbol{\xi}) = \frac{1}{nh_1 \ldots h_d} \sum_{i=1}^{d} \kappa\left(\frac{\xi_1^{(i)} - \xi_1}{h_1}\right) \ldots \kappa\left(\frac{\xi_d^{(i)} - \xi_d}{h_d}\right), \tag{3.7}$$

with $h_1 = h_2 = \cdots = h_d = h'$ when using only one bandwidth h'.

3.3 Motivation for adaptive KDE

Even tough fixed kernel methods are absolutely able to produce good results, they can still fail to capture essential information of the true probability function, as shown in [Sai94, p. 5ff]. The problem is, that some area of the sample space need a significantly larger or smaller bandwidth then others, to accurately estimate the density in said area.

Not limiting the choice of the bandwidth(s) to one, or one per dimension respectively, should solve this problem. Adaptive kernel density estimators choose the kernel bandwidths based on either the corresponding sample point or the testing point, at which the density is to be estimated.

Important to note is that adaptive methods are said to be, in many cases, inferior, to fixed KDEs when dealing with one dimensional data. For higher dimensions adaptive KDEs ought to generally yield good results (see e.g. [TS92, p. 3] and [Sai94, p. 7]).

3.4 *Balloon* Estimator

The keyword *balloon* means, that the choice of the bandwidth is influenced by the position of the testing point, rather than just the training sample points.

$$\hat{f}(\boldsymbol{\xi}) := \frac{1}{nh(\boldsymbol{\xi})^d} \sum_{i=1}^{n} \kappa\left(\frac{\boldsymbol{\xi}^{(i)} - \boldsymbol{\xi}}{h(\boldsymbol{\xi})}\right) \tag{3.8}$$

It is worth to be noted that another well-known density estimator, the nearest neighbour estimator proposed in [LQ+65], is asymptotically of the *balloon* estimator type (KDE like 3.8 with $h(\boldsymbol{\xi})$ being the distance from $\boldsymbol{\xi}$ to it's *k*-th nearest neighbour in sample space.

Since this estimator is known for producing good results in the application field of classification (see [Mil11, p. 3]), *balloon* estimator are bound to find usage in this field, see e.g. *libafg* (see [Mil11]).

A possible disadvantage of this approach is that the estimate does not necessarily result in a probability density, even when the used kernel function is. While this may not necessarily be a downside to tasks like *classification* or *clustering*, if a PDF is to be estimated, this approach might not be the first choice, as the result has to be appropriately adjusted to describe a probability density.

3.5 *Sample Point* Estimator

With this approach, the bandwidths are solely dependent on the sample points and not where the estimate is evaluated, as seen in 3.4. The established adaptive KDEs proposed by Prucell et al (see [BMP77]) and Abramson (see [Abr82]) are of this type.

$$\hat{f}(\boldsymbol{\xi}) := \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h(\boldsymbol{\xi}^{(i)})^d} \kappa\left(\frac{\boldsymbol{\xi}^{(i)} - \boldsymbol{\xi}}{h(\boldsymbol{\xi}^{(i)})}\right) \tag{3.9}$$

In contrast to the *balloon* estimator, *sample point* estimator produces a probability density, as long as the used kernel describes one too. This attribute obviously makes the *sample point* estimator much more attractive for probability density estimation.

But this method also is not flawless. Terrell and Scott mention the occurrence of *nonlocality* ([TS92, p. 4]). This signifies that samples, far away from the testing point, can have a palpable effect on the estimate at said point. This is especially pronounced, when there are outlier among the testing samples.

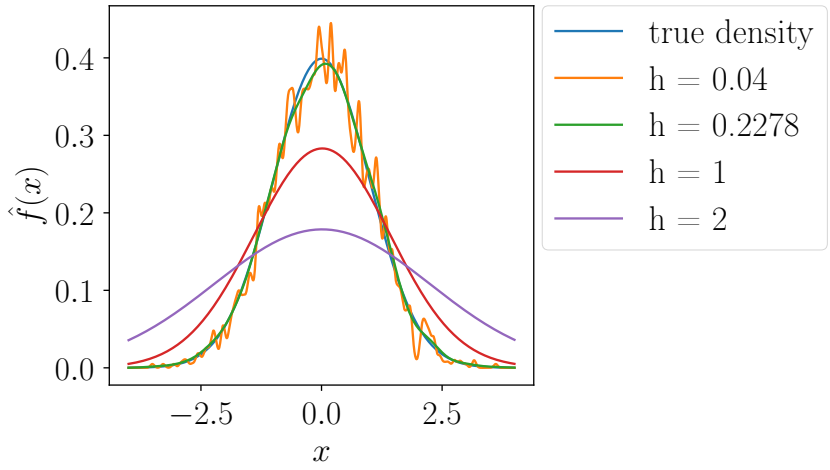

Figure 3.4: Influence of the bandwidth on a fixed KDE, using a Gaussian kernel function.

3.6 Bandwidth selection

As mentioned in the last section (3.1), the selection of the bandwidth is crucial. As shown in figure 3.4, it becomes evident that the bandwidth choice is by no means irrelevant. While the green estimate almost matches the true underlying density function, the violet one over smooths the data significantly, while the orange one afflicted by palpable noise.

As for how the bandwidth(s) is/are chosen, there are multiple approaches reaching from rule-of-thumb values to elaborate procedures, such as cross validation optimization (see chapter 2.3).

3.7 Fixed Bandwidth Selection

For selecting bandwidths for a fixed kernel density estimator, there are several rule-of-thumb methods to choose from. For example, there is Silverman's Rule:

$$h = \left(\frac{4\hat{\sigma}^5}{3n}\right)^{\frac{1}{5}}, \tag{3.10}$$

with $\hat{\sigma}$ being the standard deviation of the training-samples. Silverman's Rule is designed to reduce the mean integrated square error of a kernel density estimate of an univariate Gauss distribution. Therefore using 3.10 as bandwidth can likely result in poor estimates, but as shown in chapter 5, it is not useless to be used on datasets from other distributions.

A more sophisticated approach would be to utilize cross validation, as described previously, in conjunction with an optimizer, for example Nelder-Mead (see [NM65]), to optimize the bandwidth(s) of a fixed kernel density estimator.

3.8 Adaptive Bandwidth Selection

There are two popular approaches to the bandwidth selection of adaptive kernel density estimators.

First there is the method described by Abramson (see [Abr82]), where initially a pre-estimate $\hat{f}_{\text{pre}}$ is computed, e.g. with a fixed KDE using a rule-of-thumb bandwidth. With these values and some prefactor γ, the bandwidths of the adaptive KDE defined as such:

$$h(\boldsymbol{\xi}^{(i)}) = \gamma(\boldsymbol{\xi}^{(i)})\ \hat{f}_{\text{pre}}(\boldsymbol{\xi}^{(i)})^{-\frac{1}{2}} \tag{3.11}$$

That leaves the need to optimize the bandwidths with γ as optimization factor.

Secondly there is the k-th nearest neighbour (kNN) approach by Prucell et al (see [BMP77]). They propose to choose the bandwidths as follows:

$$h(\boldsymbol{\xi}^{(i)}) = \alpha_k\ d_{i,k}, \tag{3.12}$$

where $d_{i,k}$ equals the distance from sample point i to its' k-th nearest neighbour in the train-sample set $\mathcal{S} = \{\boldsymbol{\xi}^{(1)}, \ldots, \boldsymbol{\xi}^{(n)}\}$ and some pre-factor α_k. This leaves two optimization criterion, *k* and α_k.

Terrell and Scott suggested omitting the pre-factor α_k altogether (see [TS92, p. 3]), which leaves only one optimization criterion that is even discrete.

$$h(\boldsymbol{\xi}^{(i)}) =\ d_{i,k} \tag{3.13}$$

That makes the optimization process possibly much easier, as there are only a limited amount of possible values for *k*.

When dealing with multivariate data, there is usually only one bandwidth for all dimensions, for the neighbours are searched in the entire sample space and not only amidst one dimension.

3.9 Rosenblatt Transformation

In [Ros52], Rosenblatt presents a way to transform an arbitrary probability distribution into an uniformly distributed and independent one.

Not only is this an interesting application example, it is also often utilized in the context of uncertainty quantification, where large datasets are usually used to normalize input spaces. Thus it would be informative whether the proposed method of this thesis yields good results for this kind of task. Thereby the Rosenblatt transformation, besides being an example, also doubles up as another evaluation fo the content of this thesis.

3.9.1 Theory

Let $\boldsymbol{\xi}$ be a random vector, F its CDF and f the corresponding PDF and Ξ the sample space.

Then the Rosenblatt transformation is defined as

$$\begin{aligned} \eta_1 &= F_1(\xi_1) \\ \eta_2 &= F_{2|1}(\xi_2|\xi_1) \\ &\vdots \\ \eta_D &= F_{D|D-1,\dots,1}(\xi_D|\xi_{D-1},\dots,\xi_1) \ , \end{aligned} \tag{3.14}$$

with

$$\begin{aligned} F_{k|k-1,\dots,1}(\xi_k|\xi_{k-1},\dots,\xi_1) &= \int_{-\infty}^{\xi_k} f_{k|k-1,\dots,1}(\hat{\xi}_k|\xi_{k-1},\dots,\xi_1)d\hat{\xi}_k \\ &= \int_{-\infty}^{\xi_k} \frac{f_{k,k-1,\dots,1}(\hat{\xi}_k,\xi_{k-1},\dots,\xi_1)}{f_{k-1,\dots,1}(\xi_{k-1},\dots,\xi_1)}d\hat{\xi}_k \ , \end{aligned} \tag{3.15}$$

and all η_i are mutually independent random variables, that are uniformly distributed.

Let's assume a training-sample set $\mathcal{S} = \{\xi^{(1)},\dots,\xi^{(n)}\}$ drawn from Ξ according to F, which is used to create a *sample point* kernel density estimator with uniform weights $\hat{f}$ (see equation 3.13). Up to our knowledge, the calculations for the Rosenblatt transformation, while using such an adaptive kernel density estimator like $\hat{f}$, has not been done before. In the following, we will describe how to conduct the Rosenblatt transformation using $\hat{f}$, while being inspired by [SDNM10], where the calculation was done for fixed kernel density estimator with Gaussian kernel functions.

As seen in the equation 3.15, there are, two operations that are required for the transformation: the marginalization and the conditionalization. Furthermore, Let Ξ_i be the i-th dimension of the sample space.

$\hat{f}$ can be marginalized as such:

$$\begin{aligned}
\hat{f}(\xi_k,\ldots,\xi_1) &= \int_{\Xi_D}\cdots\int_{\Xi_{k+1}} \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_i^k}\prod_{d=1}^{k}\kappa\left(\frac{\xi_d-\xi_d^{(i)}}{h_i}\right)\frac{1}{h_i^{D-k}}\prod_{d=k+1}^{D}\kappa\left(\frac{\hat{\xi}_d-\xi_d^{(i)}}{h_i}\right)d\hat{\xi}_k\ldots d\hat{\xi}_D \\
&= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_i^k}\prod_{d=1}^{k}\kappa\left(\frac{\xi_d-\xi_d^{(i)}}{h_i}\right)\prod_{d=k+1}^{D}\underbrace{\frac{1}{h_i}\int_{\Xi_d}\kappa\left(\frac{\hat{\xi}_d-\xi_d^{(i)}}{h_i}\right)d\hat{\xi}_d}_{=1} \\
&= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_i^k}\prod_{d=1}^{k}\kappa\left(\frac{\xi_d-\xi_d^{(i)}}{h_i}\right)
\end{aligned} \tag{3.16}$$

With the marginalization, as described in the above equation 3.16, we can now conditionalize $\hat{f}$ as follows:

$$\begin{aligned}
\hat{f}(\xi_k|\hat{\xi}_{k-1},\ldots\hat{\xi}_1) &= \frac{f^{\kappa}_{k,\ldots,1}(\xi_k,\hat{\xi}_{k-1},\ldots\hat{\xi}_1)}{f^{\kappa}_{k-1,\ldots,1}(\hat{\xi}_{k-1},\ldots\hat{\xi}_1)} \\
&= \frac{\frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_i^{k-1}}\left(\prod_{d=1}^{k-1}\kappa\Big(\frac{\hat{\xi}_d-\xi_d^{(i)}}{h_i}\Big)\right)\frac{1}{h_i}\kappa\Big(\frac{\xi_k-\xi_k^{(i)}}{h_i}\Big)}{\frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_i^{k-1}}\prod_{d=1}^{k-1}\kappa\Big(\frac{\hat{\xi}_d-\xi_d^{(i)}}{h_i}\Big)} \\
&= \sum_{i=1}^{n}\underbrace{\left(\frac{\frac{1}{h_i^{k-1}}\prod_{d=1}^{k-1}\kappa\Big(\frac{\hat{\xi}_d-\xi_d^{(i)}}{h_i}\Big)}{\sum_{i=1}^{n}\frac{1}{h_j^{k-1}}\prod_{d=1}^{k-1}\kappa\Big(\frac{\hat{\xi}_d-\xi_d^{(i)}}{h_j}\Big)}\right)}_{=:c_k}\frac{1}{h_i}\kappa\Big(\frac{\xi_k-\xi_k^{(i)}}{h_i}\Big) \\
&= \sum_{i=1}^{n}c_k\frac{1}{h_i}\kappa\Big(\frac{\xi_k-\xi_k^{(i)}}{h_i}\Big)
\end{aligned} \tag{3.17}$$

As mentioned section 3.1, it is very easy to compute the integral of a kernel density estimate, when using kernel functions like the Gaussian or Epanechnikov. Thereby we have all the tool needed to compute equation 3.15, which leaves us with calculating each eta_i from equation 3.14 one after another. With that, the Rosenblatt transformation is concluded.

In chapter 5.3, the above described method will be deployed to demonstrate that it works and also to see, how well suited adaptive kernel density estimator are for such a task.

4 Probability density estimation on large data sets with kernel-based methods

In the course of this chapter the issues with kernel density estimation and large datasets are addressed, as well as possible solutions in the form of sub-sampling. There will also be reasoning behind the choice of the KDE approach and how it interacts with the sub-sampling method.

4.1 Used Adaptive KDE

The first choice to be made is whether to use a fixed KDE or an adaptive one. Since it is our goal to estimate highly dimensional probability densities, it would be advisable to go with the adaptive approach, according to the reasoning described in section 3.3.

As for the two different approaches to adaptive kernel density estimation; only the 'sample point' estimator yields a probability density by default and has not to be post-processed to archive this. Hence we choose the 'sample point' estimator.

Unfortunately all adaptive KDEs are rather costly to optimize. Let's assume a sample set $\mathcal{S}$ with $|\mathcal{S}| = n$. For the Meißel/Prucell approach n nearest neighbour algorithms need to be executed, which results in a runtime complexity of $\mathcal{O}(n^2)$. The Abramson approach runs into the same issue, since the density at each sample point is required. Even when using a rule-of-thumb bandwidth, the evaluation of the approximated PDF at a given point lies in $\mathcal{O}(n)$, which means the whole procedure also falls into $\mathcal{O}(n^2)$.

Since both methods lie in the same complexity class it might be sensible to choose the one, that is easier to optimize. The prefactor γ of 3.11 is a real number, which makes it harder and more cumbersome to optimize than k from 3.13, for $k \in \mathbb{N}$.

Another convenience is that there is no need to use multiple bandwidths for the kNN method, for the neighbours being searched in all dimensions and thus making one bandwidth per dimension obsolete.

For the sake of optimization we deploy cross validation, using log-likelihood as metric for the quality of the density estimate, as described 2.3.

4.2 Sub-sampling

Even when dealing with slightly larger datasets (about 10000 samples) kernel-based methods already require quite some time, when not using any plug-in optimizer, to compute an estimate. When the dataset on hand is even much larger, KDEs are confronted by its limits. No matter how much the implementation is optimized, especially when using costly optimization methods such as Cross Validation, the computation cost becomes infeasible.

Without changing the fundamentals of a kernel-based estimator, the most obvious way to speed up the computation is to only use a subset of the original training samples instead of all of them. Depending on how small the subset is chosen to be, the speed-up will be massive.

Of course, a decline in quality is to be expected when using a sample set of smaller magnitude. But this loss might be reasonably small, so that a KDE with sub-sampled training data might be considered. Such techniques that balance quality of the approximation and runtime complexity of the method are employed by other methods such as the sparse grid density estimation (SGDE) (see e.g. [PPB14]) as well. For example, in the case of SGDE, sub-sampling is done by the sparse grid generation. Since the original sample set is expected to be vast, there are not many different ways to find a suitable sub-sampling of the data, without an infeasible amount of calculation cost.

The most blatant and trivial way to choose the subset is a random sub-sampling approach. It is a very inexpensive method and maybe absolutely viable even. To make it a slightly more robust and consistent, a repeated random sub-sampling should be used instead.

Before drawing the first subset, a testing-sample set has to be determined. This also occurs by drawing random samples from the sample set $\mathcal{S}$. Afterwards we draw multiple sample subsets $\mathcal{S}_{\text{sub}^{(i)}} \subset \mathcal{S}$. Each $\mathcal{S}_{\text{sub}^{(i)}}$ gets evaluated, using the same testing set. In the end the subset with the greatest log-likelihood is chosen.

Another conceivable method would be to partition the sample set into many very small subsets and combine the ones, that show the most promise. However, this was not tested as to compete with SGDE, the used sub-sampled training set has to bee very small ($|\mathcal{S}_{\text{sub}}| \ll n$) and the partitions could not be small enough to warrant a combination of them, for the optimization of the bandwidths requires about 50 training samples.

In the next chapter we will see whether something as simple as random sub-sampling can actually make KDEs compete with SGDE methods when used on large datasets.

5 Experiments

In this chapter, the performed experiments are described evaluated and discussed. Almost all datasets that used in this chapter originate from *Benjamin Peherstorfer et al.*, who used these datasets in his publication [PPB14] to evaluate their proposed SGDE method and compare it to established density estimators. These datasets were chosen for two reasons.First, so that there are some realistic and plausible log-likelihood values to directly compare the results of the following experiments to them. And also to specifically compare with the SGDE method, for them being less dependant on the dataset size, in stark contrast to regular KDE approaches, which makes kernel density estimates less attractive when dealing with very large sample sets. Lastly we perform the Rosenblatt transformation to validate the calculations in chapter 3.9 and also as another validation for our implemented kernel density estimator.

The adaptive kernel density estimator, that is used in this chapter, is the one described previously in 4.1. Apart from this one, there are two more kernel density estimators that were used as comparison to the adaptive one. Both of these KDE operate with fixed bandwidths and one bandwidth per dimension (3.7). One of them uses Silverman's rule-of-thumb (3.10), the other cross validation in combination with a Nelder-Mead simplex optimizer.

5.1 Kernel choice

In chapter 3 it was mentioned, that the choice of the kernel is supposed to be rather unimportant, especially in comparison to the bandwidth(s). To ascertain this claim, we did some experiments. Figure 5.1 and 5.2 show a density estimate of a beta distribution ($\alpha = 2$, $\beta = 5$) each, one time using a fixed kernel density estimator, the other time an adaptive one. In both figures, the estimators used the Gaussian kernel, as well as the Epanechnikov kernel. It is apparent, that in each figure both curves are very similar and therefore strengthens the claim that the kernel choice is not imperative.

Albeit this one example is not proof enough. This is why we calculated the log-likelihood of some other estimates, using both kernel functions to compare them (see table 5.1). Thereby we focused on the datasets, where the KDEs proved to provided a plausible

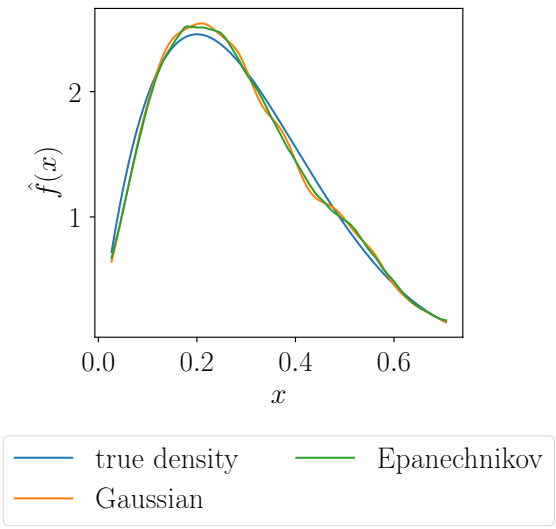

Figure 5.1: 2-5-beta distribution, approximated by a fixed KDE using cross validation.

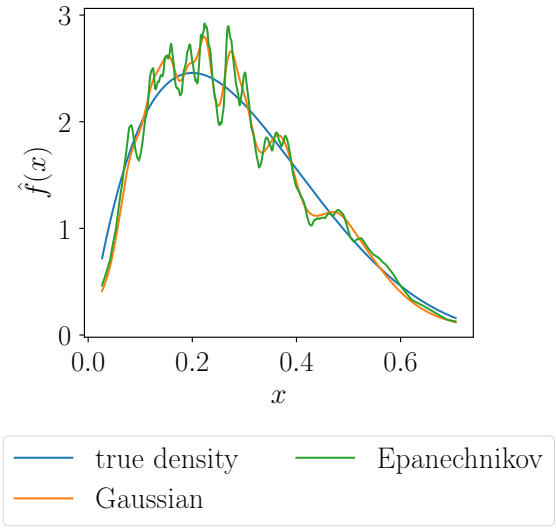

Figure 5.2: 2-5-beta distribution, approximated by an adaptive KDE using cross validation.

Dataset	DR6	liver	spheres	svmguide	checker
Gaussian	16.8	7.8	4.5	5.4	0.64
Epanechnikiv	19.3	9.2	4.8	6.1	0.69

Table 5.1: Log-likelihood values for the adaptive KDE, using Gaussian and Epanechnikov kernel functions, on different datasets.

estimate, as seen in the chapter 5.2. While the differences are not very large, it still is apparent that the Epanechnikov kernel performs better that the Gaussian overall. Furthermore are the differences not small enough to claim that the kernel choice is irrelevant, though in comparison to a change of the bandwidth we can say that the bandwidth choice is indeed more important and delicate.

Furthermore it is interesting, that the adaptive KDE seems to profit more from the Epanechnikov kernel compared to the fixed KDE, at least with the datasets from Peherstorfer. That is noticeable e.g. with the dataset *spheres*, as shown in figure 5.3 and 5.3.

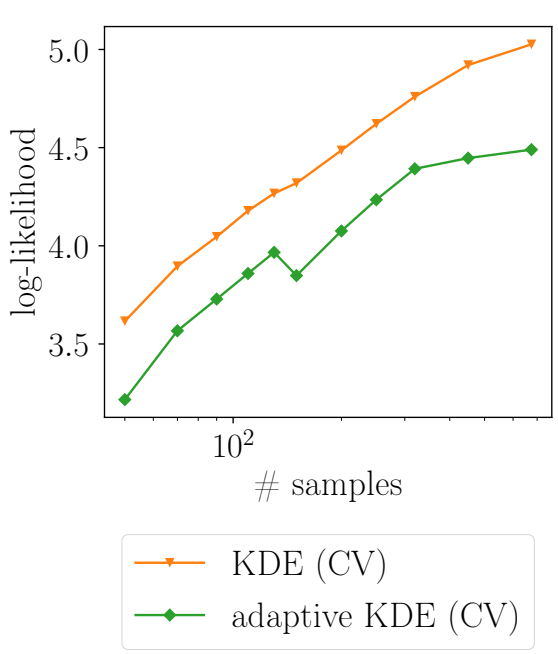

Figure 5.3: Spheres - dataset, KDEs with Gaussian kernel.

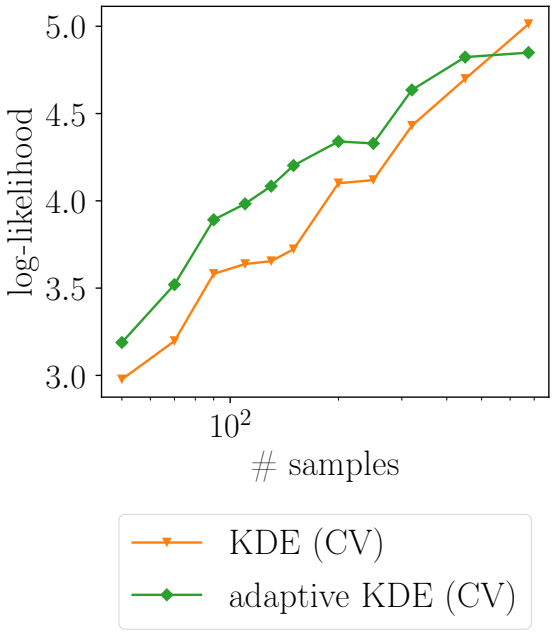

Figure 5.4: Spheres - dataset, KDEs with Epanechnikov kernel.

5.2 Comparison with Benjamin Peherstorfers' *Density Estimation with Adaptive Sparse Grids*

In this section, we compare the results of our adaptive kernel density estimator, using the Epanechnikov kernel, with and without sub-sampling, to state-of-te-art methods, used in Benjamin Peherstorfers' [PPB14]. He used multiple real world data sets to match his sparse grid density estimation method against two other established density estimators, namely the KDE from the *libagf* library (see [Mil11]) and a density tree method. Within the scope of this thesis, the largest of the used datasets are the most relevant, though analysing the smaller ones, especially those of higher dimensionality, are still worth scrutinizing.

Table 5.2 lists the datasets, including the number of dimensions and the size of the training and testing sample set. Additionally there is also the size of the sub-sampled training data, for the dataset where sub-sampling was deployed. The sub-sampling was omitted for the smallest datasets and the ones that could not compete with SGDE anyway. When a sub-sampling process was used, the size of it was determined by either the number of grid points, the corresponding SGDE used, or by choosing a size that is

Dataset	d	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{S}_{\text{sub}}\rvert$	#GP	SGDE	aKDE	sub aKDE
checker	2	100k	50k	1000	769	1.82	-	0.63
spheres	3	607	68	320	351	1.29	4.85	4.6
svmguide	4	3089	4000	500	769	4.22	6.1	5.6
DR6	4	500k	200k	1000	7937	8.37	-	19.3
bupa liver	6	290	55	-	10625	3.97	9.2	-
olives	8	348	88	-	6401	7.30	-12.9	-
oilflow	12	1318	687	-	3249	7.52	1.6	-

Table 5.2: Log-likelihood values for the different estimates on distinct datasets, used by Peherstorfer in [PPB14]. The log-likelihood values for SGDE and numbers of grid points (#GP) are also taken over from Peherstorfer.

not too small in comparison to the original dataset and yet not so large, that it becomes infeasible to conduct the kernel density estimation.

DR6 - dataset

The DR6 dataset is with 500000 training samples and 100000 testing samples by far the biggest of the datasets, used in these experiments. Due to the size we could not use the whole dataset to train our KDE, therefore we had to use the proposed sub-sampling strategy. Since the goal of the sub-sampling is to achieve very low calculation time, we evaluated different sizes of sub-sample sets up to the size 1000. As shown in table 5.2, the log-likelihood of the kernel density estimate is much higher that for the SGDE, even though the SGDE used a lot more grid points than the size of the sub-sample set. Furthermore it is shown in figure 5.5, that the adaptive KDE also performs better than the fixed ones.

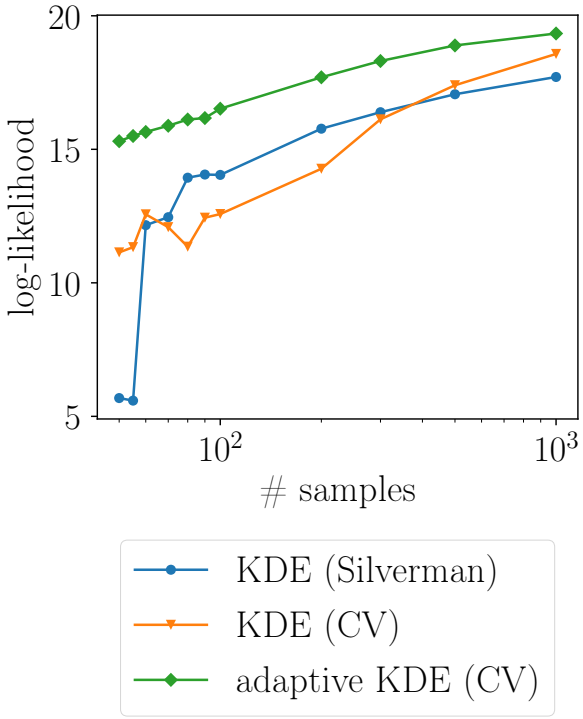

Figure 5.5: DR6 dataset. Log-likelihood of different KDEs against size of sub-sampled training set.

Checker - dataset

With 100000 training samples and 50000 testing samples, the checker dataset is the second largest one. Very different from the DR6 dataset, the log-likelihood indicates that the estimate, using our adaptive KDE, is worse that SGDE. According to Terrell and Scott ([TS92]) adaptive KDE tend to perform worse on low dimensions. Since the checker dataset is two dimensional this might be an explanation, though the fixed KDE also did not perform much better. It seems that kernel based methods in general have issues with this dataset.

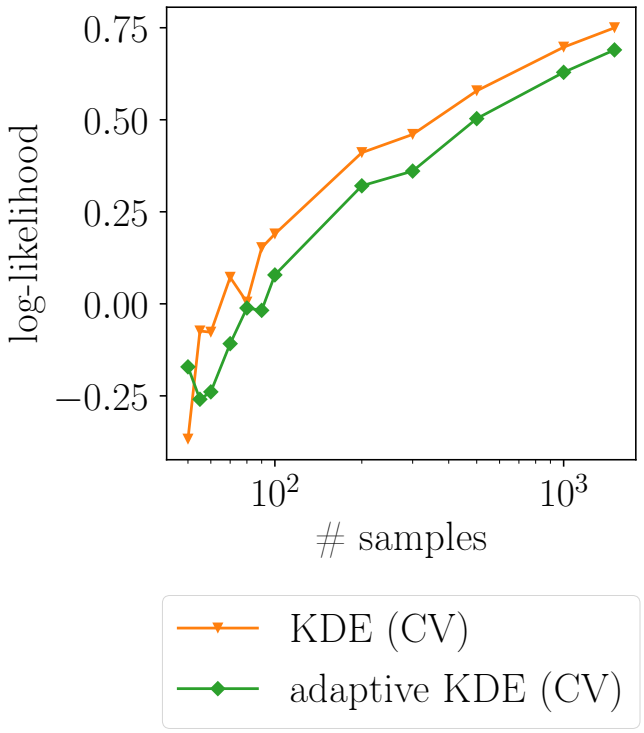

Figure 5.6: Checker dataset. Log-likelihood of different KDEs against size of sub-sampled training set.

Svmguide1 - dataset

The svmguide dataset consists of 3089 train-samples and 4000 test-samples of dimensionality 4. Among the datasets it is a larger one, albeit not exactly vast and it is not excessively time-consuming to process. Since the plot in figure 5.7 is repeatedly smoothed, there are not many differences in comparison to plot of the sub-sampled one, therefore it is omitted.

It is worthy to note that a subset of not even 100 train-samples archives higher log-likelihood values than SGDE (see table 5.2). With that in mind, if time is of the essence, a sub-sampling approach with kernel density estimators can very well be considered. Furthermore we observe again, that the adaptive KDE has the edge over the fixed variants.

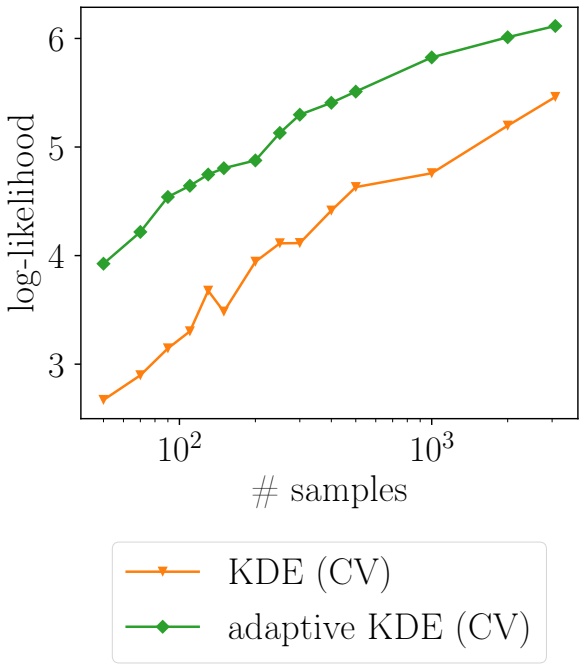

Figure 5.7: Svmguide1 dataset.
Log-likelihood of different KDEs against size of sub-sampled training set.

Liver Disorder - dataset

While this is, with 290 training samples and 55 test samples, the smallest dataset, it is also the one with the highest dimensionality and therefore very costly to process for SGDE. Hence it is a very favourable dataset for KDE, which is reflected by the log-likelihood values in table 5.2.

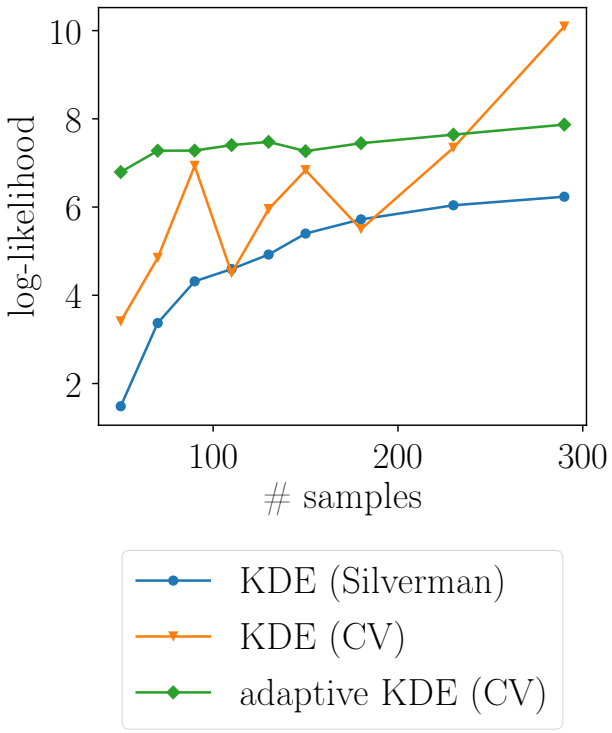

Figure 5.8: Liver-disorder. Log-likelihood of different KDEs against size of sub-sampled training set.

Remaining Datasets and Remarks

As for the remaining datasets, our KDE did quite well for the spheres dataset, but yielded abysmal results for the olives dataset and very modest ones for the oilflow dataset (see table 5.2).

Even though our adaptive KDE beat the SGDE only in one of the two really large datasets, the overall results were decent and at the end of the day, using adaptive kernel density estimation on a reduced sample set seems to be worth considering when dealing with such datasets.

5.3 Rosenblatt Transformation

The Rosenblatt transformation is an interesting operation in the context of UQ, where large and (highly) multidimensional datasets frequently occur. Hence it is a fine application example, which might benefit from the proposed density estimation method

and furthermore serves as another validation scenario, as we can see if the transformed random variable is approximately uniformly distributed.

For this example, we used a synthetic sample set, containing 40000 samples. This data set is called two moons and a subset of 2000 samples can be seen in figure 5.9. The adaptive KDE, which is to be used to conduct the Rosenblatt transformation, according to the method described in 3.9, is constructed by 1000 samples, chosen by utilizing a repeated sub-sampling of the dataset.

Figure 5.10 shows the transformed subset. Evidently, the samples are approximately uniformly distributed, thus the transformation was successful.

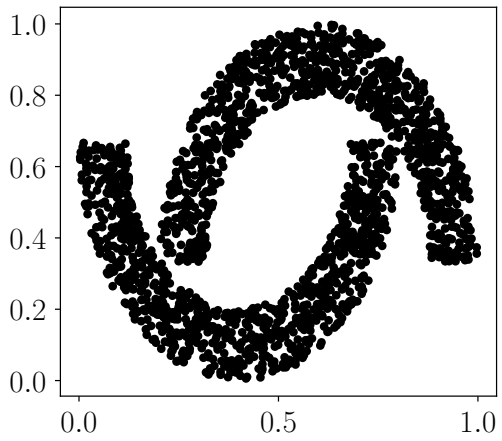

Figure 5.9: Two Moons dataset, before the Rosenblatt transformation.

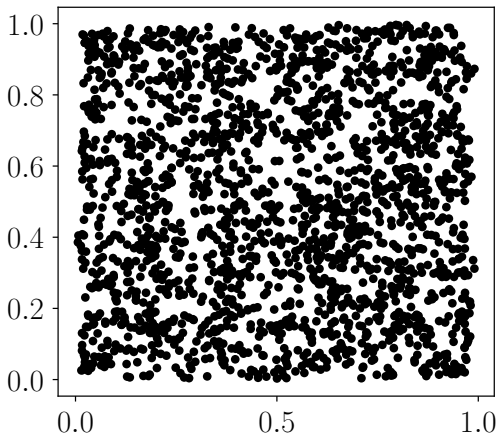

Figure 5.10: Two Moons dataset, after the Rosenblatt transformation.

6 Summary and Prospects

After introducing some background information, we discussed several approaches to kernel density estimation and how to optimize these estimates. The goal of this thesis was to use an adaptive kernel density estimator, to estimate a probability density function, utilizing a large dataset, without the caveat of a gigantic runtime, for the runtime of kernel density estimators are strongly dependant on the size of the training sample set. Ultimately we chose a simplified version of the Meißel and Prucell *sample point* estimator, though in order to efficiently process large datasets, some kind of sub-sampling is needed. In the experiments it became apparent that a simple method, such as repeated sub-sampling in combination with an optimization technique, such as cross validation, can yield very decent results, even in comparison with SGDE, another density estimation method that balances quality of the approximation and runtime complexity. To show off another application example, we additionally introduced the Rosenblatt transformation and described how to conduct this transformation, using an adaptive KDE of the *sample point* estimator type. This was also tested and yielded satisfactorily results.

Apart from performance improvements, it would be worthwhile to try the not simplified Meißel/Prucell *sample point* estimator, as well as the Abramson estimator. But testing other sub-sampling methods, especially more complex ones, would be an endeavour most interesting.

Bibliography

[Abr82] I. S. Abramson. “On bandwidth variation in kernel estimates-a square root law.” In: *The annals of Statistics* (1982), pp. 1217–1223 (cit. on pp. 21, 23).

[BMP77] L. Breiman, W. Meisel, E. Purcell. “Variable kernel estimates of multivariate densities.” In: *Technometrics* 19.2 (1977), pp. 135–144 (cit. on pp. 21, 23).

[CKS08] E. Cramer, U. Kamps, A. Steland. *Grundlagen der Wahrscheinlichkeitsrechnung und Statistik*. Springer, 2008 (cit. on p. 13).

[EM12] H. C. Elman, C. W. Miller. “Stochastic collocation with kernel density estimation.” In: *Computer Methods in Applied Mechanics and Engineering* 245 (2012), pp. 36–46 (cit. on pp. 14, 18).

[HPK11] J. Han, J. Pei, M. Kamber. *Data mining: concepts and techniques*. Elsevier, 2011 (cit. on p. 15).

[LQ+65] D. O. Loftsgaarden, C. P. Quesenberry, et al. “A nonparametric estimate of a multivariate density function.” In: *The Annals of Mathematical Statistics* 36.3 (1965), pp. 1049–1051 (cit. on p. 21).

[Mil11] P. Mills. “Efficient statistical classification of satellite measurements.” In: *International Journal of Remote Sensing* 32.21 (2011), pp. 6109–6132. DOI: 10.1080/01431161.2010.507795. eprint: http://dx.doi.org/10.1080/01431161.2010.507795 (cit. on pp. 11, 18, 21, 33).

[NM65] J. A. Nelder, R. Mead. “A simplex method for function minimization.” In: *The computer journal* 7.4 (1965), pp. 308–313 (cit. on p. 23).

[PPB12] B. Peherstorfer, D. Pflüger, H.-J. Bungartz. “Clustering based on density estimation with sparse grids.” In: *KI 2012: Advances in Artificial Intelligence* (2012), pp. 131–142 (cit. on p. 11).

[PPB14] B. Peherstorfer, D. Pflüger, H.-J. Bungartz. “Density Estimation with Adaptive Sparse Grids for Large Data Sets.” In: *Proceedings of the 2014 SIAM International Conference on Data Mining*. 2014, pp. 443–451 (cit. on pp. 11, 14, 18, 28, 31, 33, 34).

[Ros52] M. Rosenblatt. “Remarks on a Multivariate Transformation.” In: *The Annals of Mathematical Statistics* 23.1952 (1952), pp. 470–472. DOI: 10.1214/aoms/1177729394 (cit. on p. 24).

[Sai94] S. R. Sain. “Adaptive kernel density estimation.” PhD thesis. Rice University, 1994 (cit. on pp. 18, 20).

[SDNM10] K. Sargsyan, B. Debusschere, H. Najm, O. L. Maître. “Spectral representation and reduced order modeling of the dynamics of stochastic reaction networks via adaptive data partitioning.” In: *SIAM Journal on Scientific Computing* 31.6 (2010), pp. 4395–4421 (cit. on p. 24).

[TS92] G. R. Terrell, D. W. Scott. “Variable Kernel Density Estimation.” In: *Ann. Statist.* 20.3 (Sept. 1992), pp. 1236–1265. DOI: 10.1214/aos/1176348768 (cit. on pp. 20, 21, 23, 35).

All links were last followed on August 03, 2017.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature