DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Masters' Thesis in Robotics, Cognition, Intelligence

Spatially adaptive Density Estimation with the Sparse Grid Combination Technique

Markus Fabry

DEPARTMENT OF INFORMATICS

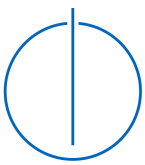
TECHNISCHE UNIVERSITÄT MÜNCHEN

Masters' Thesis in Robotics, Cognition, Intelligence

Spatially adaptive Density Estimation with the Sparse Grid Combination Technique

Räumlich-adaptive Dichteschätzung mit der Dünngitter Kombinationstechnik

Author:	Markus Fabry
Supervisor:	Univ-Prof. Dr. Hans-Joachim Bungartz
1^{st} Advisor:	Michael Obersteiner; M.Sc.
2^{nd} Advisor:	Paul Christian Sarbu; M.Sc. (Hons.)
Submission Date:	September 15, 2020



I confirm that this masters' thesis in robotics, cognition, intelligence is my own work and I have documented all sources and material used.

Munich, September 15, 2020 Markus Fabry

Acknowledgments

I want to thank my advisors, Michael Obersteiner and Paul Sarbu, for their invaluable support, advice and feedback.

Thank you dad, for helping me with proofreading and spell-checking.

Abstract

Non-parametric density estimation, especially with high dimensional data sets, presents a problem requiring powerful hardware and sophisticated algorithms. Sparse grids are an approach that offer a computationally feasible solution. However, the usual methods, like the standard combination technique, still struggle with the sheer number of dimensions and data points for some data sets. Recent advances in adaptive combination techniques present new solutions for such challenges. In this thesis one such method - the dimension-wise spatially adaptive refinement - will be analyzed and compared to the standard combination technique and regular kernel density estimation. To determine its effectiveness for density estimation it will be compared to these methods with a variety of data sets. Furthermore, a classification-based comparison of the dimension-wise method and the standard combination technique will be conducted.

Zusammenfassung

Nicht-parametrische Dichteschätzung stellt ein Problem dar, das, besonders bei Datensätzen mit vielen Dimensionen, große Rechenleistung und ausgeklügelte Algorithmen erfordert. Dünngitter Verfahren sind Methoden, die rechnerisch machbare Lösungen bieten. Trotzdem kämpfen die üblichen Dünngitter Verfahren, wie etwa die Standard-Kombinationstechnik, mit der schieren Anzahl an Dimensionen und Datenpunkten mancher Datensätze. Jüngste fortschritte für adaptive Kombinationstechnik ermöglichen neue Lösungen für solche Herausforderungen. In dieser Arbeit wird eine solche Methode - die Dimensionsweise Räumlich-adaptive Kombinationstechnik - analysiert und mit der Standard-Kombinationstechnik und Kernel Dichteschätzung verglichen. Um ihre Effektivität bei Dichteschätzung festzustellen wird sie mit den eben genannten Methoden mit einer Vielfalt von Datensätzen überprüft. Weiterhin wird ein auf Klassifikation basierender Vergleich mit der Standard-Kombinationstechnik unternommen.

Contents

1 Introduction

Data is ubiquitous in the present world. Telescopes observe and examine the stars, weather stations record humidity, wind speed and temperature, phones track locations and much more. The amount of data continues to grow, so efficient methods to disseminate and interpret all this data becomes more and more important.The methods to accomplish this are commonly grouped together under the umbrella of data mining. One such method is density estimation. For this, data is interpreted as the observations produced by probability distributions and the goal is to learn these distribution through the data. The problem is that there are infinitely many possible distributions that could have produced a given data set [1]. The intent is to find a model that closely matches the observed data, while being general enough to also match with future observations and thus allowing predictions to be made. Common approaches to solve this problem are neural networks or kernel density estimation, the latter of which will be relevant for this thesis. Another promising approach is the concept of grids, and more specifically, sparse grids. It is a flexible approach, with many applications including regression, interpolation, solving of partial differential equations, density estimation or classification. The basic principle is using a grid of points on which simple basis functions are centered. This collection of basis functions covers the entire domain and by adjusting the coefficients of the functions numerically, they can be made to fit to the function or data in question.
In this thesis, a recent innovation for sparse grid-based algorithms - dimension wise spatially adapitve refinement - will be examined in the context of the problem of density estimation. Based on the combination of sparse grids, the so called component grids, this algorithm will be compared to the standard method within the family of combination techniques as well as kernel density estimation.
Chapter 2 describes the basic concepts of sparse grids and the combination schemes used for this thesis as well as kernel density estimation, which is used as a comparison to the combination schemes. In chapter 3, the new additions to the sparseSpACE framework and the algorithms for the density estimation with dimension wise spatially adaptive refinement will be covered in detail. The evaluation of the dimension-wise method compared to the standard combination technique and the kernel density estimation will be covered in chapter 4. Lastly, chapter 5 summarizes and discusses the results of the previous chapter as well as future improvements.

2 Theory

In this section the necessary background for the implementation is introduced. Firstly, a general overview of sparse grids and associated concepts and algorithms like adaptivity and the combination technique. Special focus is given to the dimension-wise spatially adaptive combination technique on which this thesis is based on. Secondly a brief introduction to density estimation and the relevant methods for this thesis, kernel density estimation and grid based density estimation. Lastly, the general principle of classification and the approaches used in the later evaluation will be covered.

2.1 Sparse Grids

Sparse grids are a computationally efficient way of accomplishing various tasks like interpolation, regression, classification or solving of partial differential equations. Introduced by Smolyak [21] in 1963, the main idea is to discretize the domain of a data set with a set of points and basis functions. This section explains the basic principles of a grid based approximation of functions, first with a full grid and then with a sparse grid. The differences between nodal and hierarchical basis, as well as the concepts of adaptive refinement and boundary treatment will also be covered.

2.1.1 Basics

Suppose that a function $f : \Omega \to \mathbb{R}$ is given that is to be interpolated on a grid. For a given level l of refinement, the mesh width is defined as

$$h_n = \frac{1}{2^l} \tag{2.1}$$

The level also determines the number of grid points in a full grid as

$$n = 2^l - 1 \tag{2.2}$$

This means that the domain of function f is decomposed into 2^l ranges of length h_n by n grid points, with the grid points located at $i \cdot \frac{1}{2^l},\ i \in [1, |n|]$. Now a basis function

needs to be defined from which the set of basis functions for the grid is derived. One example of such a basis function is the standard hat function

$$\phi(x) = max(1 - |x|, 0) \tag{2.3}$$

from which the set of locally supported basis functions is derived by dilatation and translation

$$\phi_i(x) := \phi(2^l x - i) \tag{2.4}$$

Local support in this case means that the functions are centered on the grid points and dilated to cover the range between neighboring grid points (or the boundary, in case of the first and last grid point). Therefore the domains are defined as

$$\phi_i : \mathbb{R} \to \left[\frac{i-1}{2^l}, \frac{i+1}{2^l}\right],\ i \in [1, |n|] \tag{2.5}$$

Also note that the level for ϕ_i is fixed, since the nodal basis only works with a single level, unlike the hierarchical basis which indexes the functions $\phi_{l,i}$ by level as well.

The interpolation u that is to be fitted to f is constructed through a weighted sum of the basis functions:

$$u := \sum_{2^l-1}^{i=i} \alpha_i \phi_i(x_i, h) = f \tag{2.6}$$

The coefficients α_i, also known as surpluses, act as weights for the functions ϕ_i and are the unknown variables that need to be optimized to get a closely fitting interpolation. To accomplish the fitting, a numerical approach, like gradient descent, can be used. Figure 2.2 shows an example of a one-dimensional interpolation with nodal basis and hierarchical basis, respectively.

Using a full grid is not the most efficient way of representation, since a lot of basis functions only contribute very little to the overall solution, especially in higher dimensions[19]. Switching from a nodal to a hierarchical basis allows to omit basis functions whose contribution is negligible, creating a sparse grid.
To achieve a hierarchical decomposition of the approximation requires introducing a hierarchical index set for the grid points

$$I_l := \{i \in \mathbb{N} : 1 \leq i \leq 2^l - 1,\ i \text{ odd}\} \tag{2.7}$$

with which the hierarchical sub-spaces W_l can be obtained as follows

$$W_l := span\{\phi_{l,i}(x) : i \in I_l\}. \tag{2.8}$$

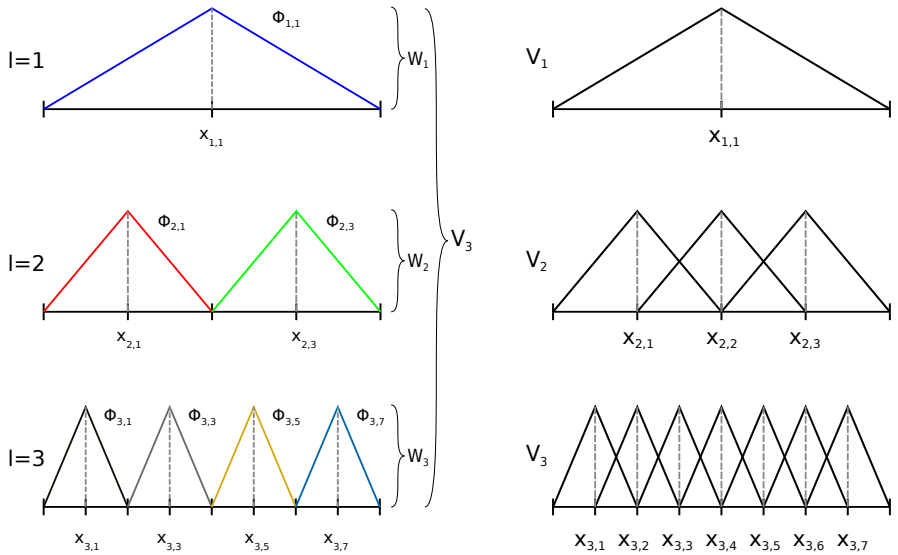


Figure 2.1: One-dimensional basis functions $\phi_{l,i}$ and the corresponding grid points $x_{l,i}$ up to level $l = 3$. The hierarchical basis is shown on the left and the common nodal point basis on the right.

$\phi_{l,i}(x)$ hereby denotes the basis functions translated and dilated

$$\phi_{l,i}(x) = \phi(2^l x - i) \; i \in I_l \tag{2.9}$$

for each level l and index i created for that level. Formulating the space of piece-wise linear functions V_n on a full grid with mesh width h_n for a given level l as a direct sum of W_l

$$V_n = \oplus_{l \leq n} W_l \tag{2.10}$$

gives the full grid with hierarchical basis. See figure 2.1 for a comparison of nodal and hierarchical basis and figure 2.2 for an example of a full grid interpolation with hierarchical and nodal basis.
To obtain a sparse grid from the full grid, only some of the sub-spaces are selected, preferably the ones that contribute most to the overall solution. For a multidimensional grid, the selection could look like this

$$V_n^1 := \bigoplus_{|\mathbf{l}|_1 \leq n+d-1} W_{\mathbf{l}}. \tag{2.11}$$

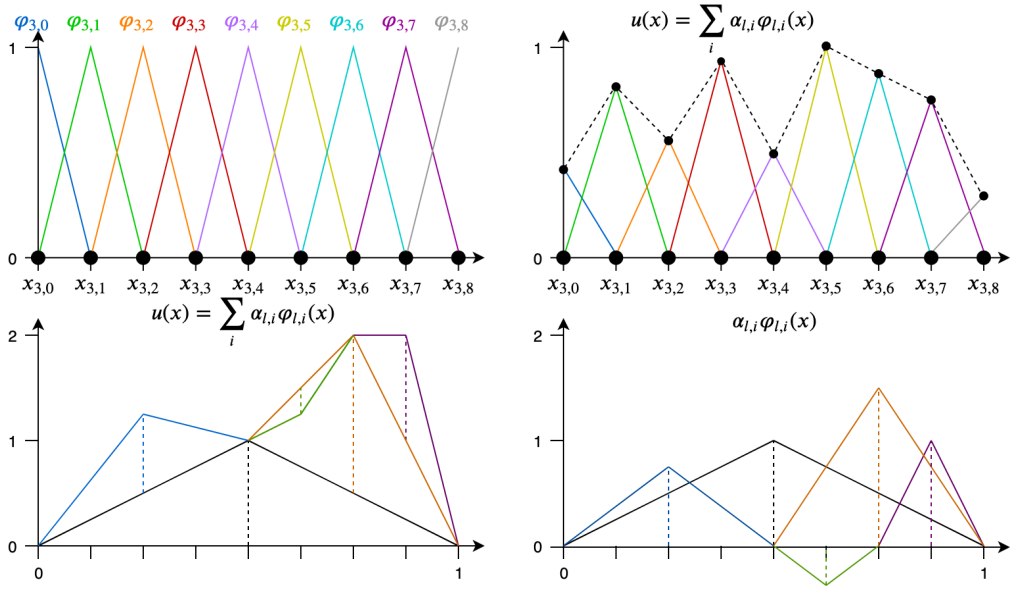


Figure 2.2: Interpolation example with a nodal basis (left) and a hierarchical basis (right)

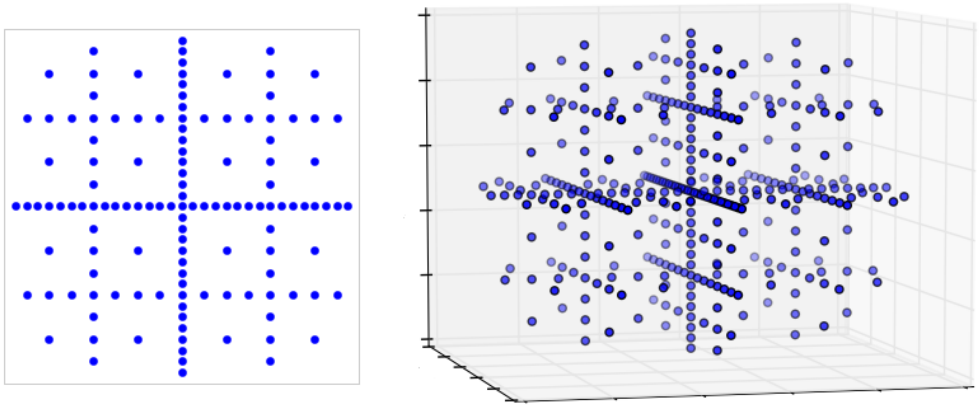


Figure 2.3: Example of a 2D Sparse Grid on the left and a 3D Sparse Grid on the right. Figure sourced from [19]

Here, $\mathbf{l}$ denotes a d-dimensional vector of level indices and $W_\mathbf{l}$ the corresponding subspace with $\mathbf{i}$ as the index in each dimension. Also note the change in the selection from $|\mathbf{l}|_1 \leq n$ to $|\mathbf{l}|_1 \leq n + d - 1$, where n denotes the exponent of the mesh width h_n and d the dimensionality. The concrete choice of sub-spaces depends on the norm

chosen to measure the error.
To extend the basis functions to multiple dimensions, a tensor product approach is used

$$\phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}) := \prod_{j=1}^{d} \phi_{l_j,i_j}(x_j) \tag{2.12}$$

with the d-dimensional level and index vectors $\mathbf{l}$ and $\mathbf{i} \in \mathbf{I}$ with

$$\mathbf{I}_\mathbf{l} := \{\mathbf{i} : 1 \leq i_j \leq 2^{l_j},\, i_j \,\text{odd},\, 1 \leq j \leq d\}. \tag{2.13}$$

The subspaces $W_\mathbf{l}$ in multiple dimensions change to:

$$W_\mathbf{l} := \text{span}\{\phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}) : \mathbf{i} \in \mathbf{I}_\mathbf{l}\} \tag{2.14}$$

and the corresponding interpolant $u(\mathbf{x}) \in V_n$

$$u(\mathbf{x}) = \sum_{|\mathbf{l}|_1 \leq n+d-1,\, \mathbf{i} \in \mathbf{I}_\mathbf{l}} \alpha_{\mathbf{l},\mathbf{i}} \phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}). \tag{2.15}$$

For a more in-depth explanation of multidimensional sparse grids see[19][6].

2.1.2 Combination Technique

When dealing with sparse grids, a popular approach to calculate the surpluses of the basis functions centered on the grid points is the so called combination technique [19][7][4]. In addition to working directly in the hierarchical base, the combination technique can be used to compute a sparse grid representation of a function, whereby a specific sequence of small anisotropic full grids represented in the conventional nodal basis, which are also called component grids, is combined linearly.
To this end, let

$$V_\mathbf{l} := \bigoplus_{\mathbf{k} \leq \mathbf{l}} \hat{W}_\mathbf{k} \tag{2.16}$$

be the full grid space of level $\mathbf{l}$, $u_\mathbf{l} \in V_\mathbf{l}$ the interpolant of some function f. The function f is discretized on a sequence of anisotropic grids $\Omega_\mathbf{l} = \Omega_{l_1}, \ldots, \Omega_{l_d}$ with uniform mesh sizes $h_t = 2^{\mathbf{l}t}$ in the t-th coordinate direction. The following grids $\Omega_\mathbf{l}$ are considered:

$$|\mathbf{l}|_1 := l_1 + \cdots + l_d = n + (d-1) - q,\, q = 0, \ldots, d-1,\, l_i > 0,\, l_i \in |\mathbf{l}|_1 \tag{2.17}$$

Incorporating a finite element approach with piecewise d-linear functions $\phi_{\mathbf{l},\mathbf{i}}(\mathbf{x})$ on each grid $\Omega_\mathbf{l}$ gives the representation in the nodal basis as follows:

$$f_\mathbf{l}(\mathbf{x}) = \sum_{j_1=0}^{2^{l_1}} \cdots \sum_{j_d=0}^{2^{l_d}} \alpha_{\mathbf{l},\mathbf{j}} \phi_{\mathbf{l},\mathbf{j}}(\mathbf{x}), \tag{2.18}$$

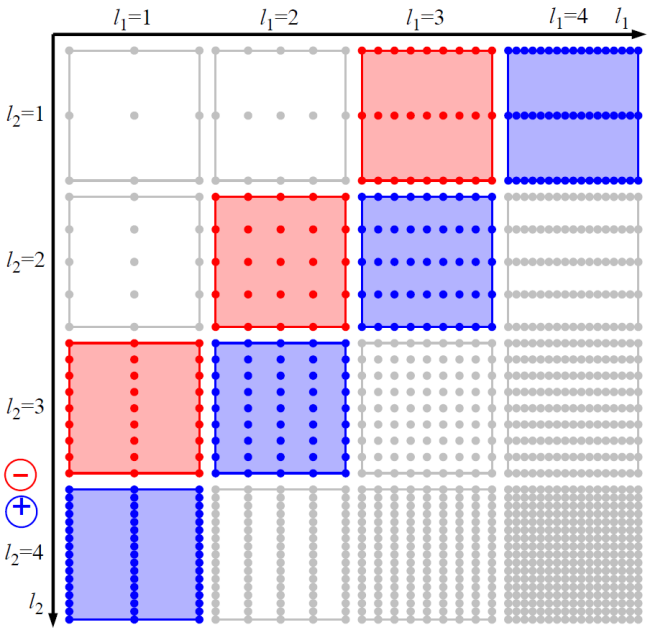
linearly combining the discrete partial functions $f(\mathbf{x})$ from the different grids $\Omega_\mathbf{l}$ according to the previous combination formula.

$$f_n^c(\mathbf{x}) := \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{|\mathbf{l}|_1 = n+d-1-q} f_\mathbf{l}(\mathbf{x}) \tag{2.19}$$

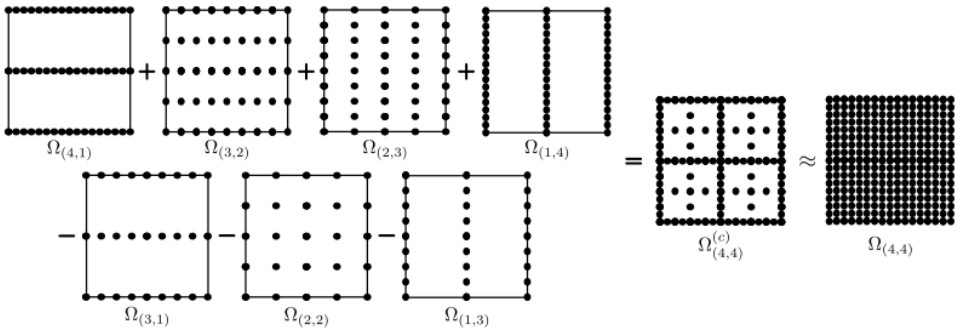
A general combination scheme can be written as,

$$f_\mathbf{I}^c(\mathbf{x}) := \sum_{\mathbf{l} \in \mathbf{I}} c_\mathbf{l} f_\mathbf{l}(\mathbf{x}). \tag{2.20}$$

The resulting function f_n^c exists in the sparse grid space, where the combined interpolant is identical with the hierarchical sparse grid interpolant [7]. Note the varying sign of the combination coefficients in 2.20, which signifies that some grid points occur several times within the combination technique.
Figure 2.4 shows the combination technique in two dimensions for a level 4 grid on the top and for a dimensionally-adaptive sparse grid on the bottom. The component grids with $|\mathbf{l}|_i = 4$ (blue) are added, while the component grids with $|\mathbf{l}|_i = 3$ (red) are subtracted. The combination technique has been extended to allow for dimension-adaptive refinement [4], dimension-wise spatially adaptive refinement [13] and another, so called split-extend method [14]. This enables to adapt to the underlying function in an a posteriori way [19], where more grid points are spent local to the data and making computation with higher dimensionalities feasible.



(a) Standard combination components



(b) Standard combination Result

Figure 2.4: Standard combination for level 4 with points on the boundary. Figure (a) shows the individual component grids involved in the combination (Sourced from [19]). Blue component grids are added, red ones are subtracted. Figure (b) shows the resulting sparse grid and the full grid equivalent.

2.1.3 Adaptivity

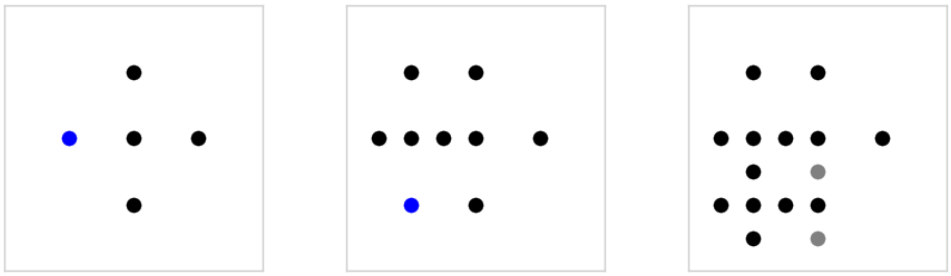


Figure 2.5: An example of a spatially adaptive sparse grid (with hierarchical basis) in 2 dimensions. The blue points in the left and middle figure denotes in which area the refinement will take place. From the first step to the second step we can see that that there are multiple points being added (colored in grey). These points are necessary, since they are the parent points in the hierarchical basis necessary for the grid to stay valid.

If the problem at hand requires higher local resolutions, for example a density with a highly irregular shape, the sparse grids can be expanded through adaptive refinement. This simply means that grid points don't get placed in a regular fashion over the entire domain compared to a regular sparse grid like in figure 2.3, but instead the points are placed in the regions where the data is actually located. The advantage of such an approach is that it avoids placing a large amount of grid points in regions that contribute very little to solving the problem. This also means that adaptive approaches inherently lend themselves to iterative methods, because they simplify minimizing grid point usage for a given accuracy tolerance or prevent overfitting. (While a non-iterative approach is still possible by, for example, determining the data locality during pre-processing and then creating a grid based on that, finding an optimal grid point "budget" for a given accuracy tolerance would still require repeated evaluations.)
Reducing the grid point count will also reduce calculation time for the solution, simply due to the lower amount of functions involved, although the overhead introduced by the iterative process and the adaptivity itself can increase the computation time by a larger margin, as can be seen later in chapter 4.
A requirement for adaptivity is that all backward neighbors of a grid point need to be present. More specifically, to create a grid point of level l and index $i \in I_l$, all grid points with level $\widehat{l}$, $\widehat{l} < l$ and index $\widehat{i} \leq i$, $\widehat{i} \in I_{\widehat{l}}$, also called ancestors, must be present. If one or more of those grid points are not present, they need to be created first before

the wanted grid point can be created.
For the combination technique, the usual approach to adaptivity is the dimension adaptive approach [4]. Instead of using 2.17 as the grid choice, it is generalized using an index set $\mathbf{l}$ with the admissability criterion:

$$\mathbf{k} \in \mathbf{l} \text{ and } \mathbf{j} \leq \mathbf{k} \implies \mathbf{j} \in \mathbf{l}. \tag{2.21}$$

Furthermore, a refinement strategy is necessary to decide which grid point are to be refined. A naive approach is to consider all possible refinements and determine how much they would reduce the error of the regression and only refine the one with highest error reduction in each iteration. Unfortunately, this approach requires considerable amount of computation and is therefore not feasible for many applications [19].
Another aspect which has to be considered is the number of grid points to refine within one refinement step. Depending on the problem at hand, refining several grid points at once can be beneficial, e.g. to avoid the refinement to focus on only a single feature. In the context of a combination technique, such an approach requires not only keeping a valid grid, but a valid combination as well [13].
For a more in-depth view on adaptivity see [18][19].

Spatial Adaptivity with Dimension-wise Refinement

In [13] Obersteiner et al. introduced an advanced a dimension-wise refinement scheme (not to be confused with the dimension adaptive combination scheme [4]) for the sparse grid combination technique. In contrast to the adaptive refinement with a regular sparse grid in hierarchical basis, where points are refined individually (while creating the necessary parent nodes, of course), this scheme instead refines the component grids of the combination in each dimension individually. Although this scheme still requires the creation of necessary parent nodes for the grids to keep the combination valid, it allows grid points to be non-equidistant to their neighbors, unlike the standard combination technique or regular full grids.
The scheme itself is based on rectilinear grids generated form 1D grids by a tensor product construction, which enables the use common 1D refinement algorithms [13].

1D Refinement As mentioned before, the refinement happens for each individual dimension. Refining a single dimension thus increases the level for that dimension only, but grid points are not only added for a single "axis" in the domain, but multiple. See Figure 2.6 for an illustration of the grid points added by increasing levels in single dimensions. If the levels in other dimensions are not equal to 1, grid points are added

in "stripes" along the grid points of the other dimension(s). The amount of points added thus depends on both the level in the dimension to be refined and the level in the other dimensions, including the hierarchical parents needed for a valid sparse grid.

The global combination scheme For the refined component grids to combine into a valid sparse grid, a combination scheme is needed. This scheme needs to fulfill the following criteria:

$$\mathbf{P}^{k,\mathbf{i}} \subseteq \mathbf{P}^{k,\mathbf{j}} \text{ for } \mathbf{i},\mathbf{j} \in \mathbf{I},\ k \in [d],\ \mathbf{j} \geq \mathbf{i} \tag{2.22}$$

and

$$\mathbf{P}^{k,\mathbf{i}} = \mathbf{P}^{k,\mathbf{j}} \text{ for } \mathbf{i},\mathbf{j} \in \mathbf{I},\ k \in [d],\ i_k = j_k \tag{2.23}$$

where $\mathbf{P}^{k,\mathbf{l}}$ denotes the set of 1D grid coordinates in dimension k for level vector $\mathbf{l}$ and I is the index set of the combination technique. The index set for the combination is based on the maximum levels $\mathbf{l}^{max}$ per dimension where $l_k^{max} = max(\mathbf{L}^k)$. The index set is defined by

$$I = \{\mathbf{l} \in \mathbb{N}^d | \left\|\mathbf{l}\right\|_1 \leq max(\mathbf{l}^{max}) + d - 1, l_i < l_i^{max} \vee (l_i = l_i^{max}, l_k = 1, k \in [d]/i)\} \tag{2.24}$$

which is the common d-dimensional simplex definition with level $l = max(\mathbf{l}^{max})$, where the scheme is cut off in case a dimension has lower maximum level. Now $\mathbf{P}^{k,\mathbf{l}}$ needs to be defined for each level vector $l \in I$. At the same time the validity of the combination scheme needs to be ensured. The component grid is then constructed by a tensor product of all 1D grids $\mathbf{P}^{k,\mathbf{l}}, k \in [d]$.
[13] includes several proposed schemes for the combination. For this thesis, the balanced scheme that tries to preserve points as well as interactions between dimensions was chosen:

$$\mathbf{P}^{k,\mathbf{l}} = \{P_j^k | j \in [|\mathbf{P}^k|], L_j^k \leq 1, (L_j^k < \max(D_{lv}(j)) \vee l_k = l_k^{\max})\} \tag{2.25}$$

with

$$\tilde{c}_j^{k,\mathbf{l}} = \max(\{m \in N | g_{k,l}(\sum_{i=0}^{m-1} g_{k,l}(c_j^k - i, d)) \leq c_j^k\} \cup \{0\}), \tag{2.26}$$

$$g_{k,l}(i,j) = \sum_{n=1}^{j} h(i,n), \text{ and} \tag{2.27}$$

$$h(x,k) = \begin{cases} 1 \text{ if } \max(\mathbf{c}^{k,\mathbf{l}}) \geq x \\ 0 \text{ otherwise} \end{cases} \tag{2.28}$$

The second condition of 2.25 guarantees that leaves of $\mathbf{P}^k$ are only added if the level vector is at the maximum level in dimension k [13]. In 2.26 the term that is subtracted

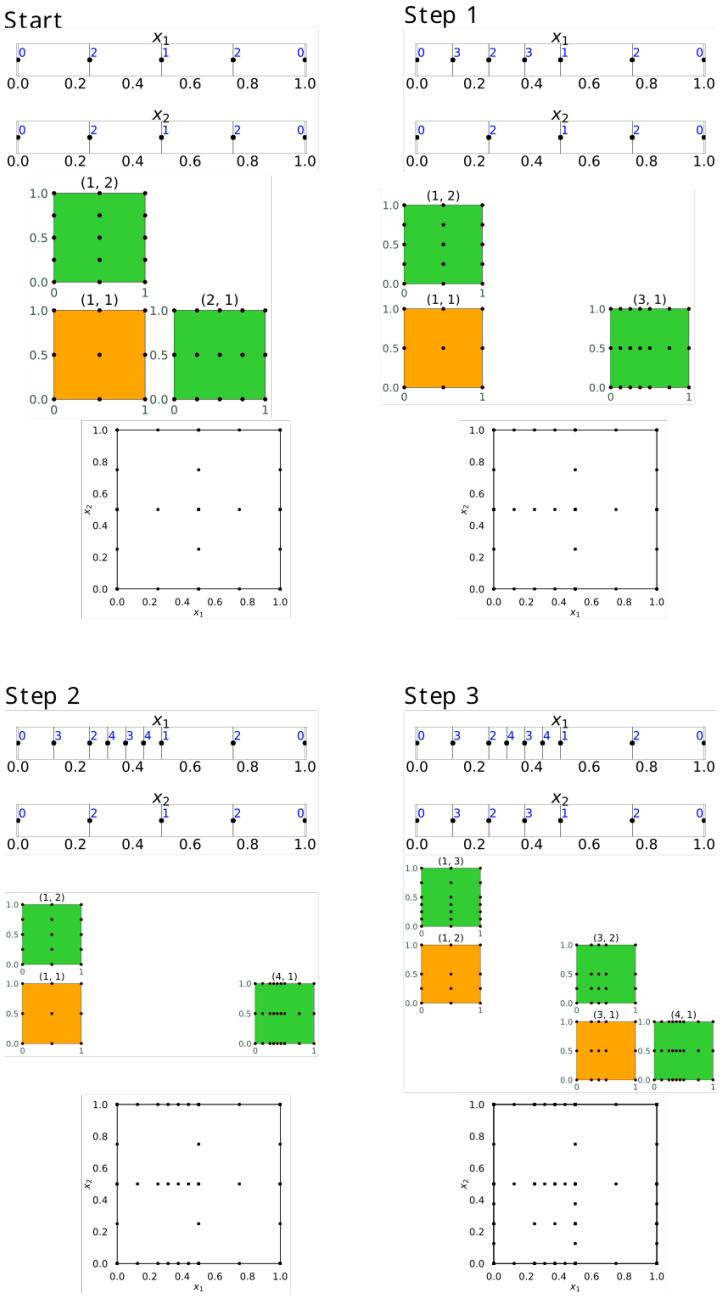


Figure 2.6: Three refinement steps of the dimension-wise spatial refinement. Each step shows $\mathbf{P}^k$ and $\mathbf{L}^k$ (blue) for each dimension x_k, the combination scheme, and the resulting sparse grid. Green component grids are added and orange ones are subtracted. From [13].

from the level vector is defined. g and h are helper functions that sum up the coarsenings that are performed on all dimensions, where $h(x,k)$ indicates if dimension k has a coarsening value of at least x.

Error estimator To determine where the dimension-wise spatially adaptive scheme should refine the grid an error measure is needed. The goal of this measure is to steer the refinement to locales where the data is located by resulting in a higher error in those regions with the most data and where the grid still has too few grid points to properly approximate it. Ideally the measure should lead to a fast convergence with the fewest points necessary. One such measure is given by

$$\epsilon_p^k = \left|\left| \sum_{\mathbf{l} \in I} c_{\mathbf{l}} \cdot \epsilon_p^{k,\mathbf{l}} \right|\right|, \tag{2.29}$$

where $\epsilon^{k,\mathbf{l}}(p)$ is the error estimate from the component grid with level vector $\mathbf{l}$ and $c_{\mathbf{l}}$ the respective combination coefficient. In each component grid, the measure is calculated for each point $p \in \mathbf{P}^k$ in dimension $k \in [d]$. $\epsilon^{k,\mathbf{l}}(p)$ denotes the error estimate from the component grid with level vector $\mathbf{l}$ and combination coefficient $c_{\mathbf{l}}$. To obtain the error for the whole component grid a 1D hierarchization is utilized to solve a system of linear equations. See [13] for the complete calculation.
This error estimator splits error values equally among the descendents of the hierarchical surplus they are associated with. In [13], the set of these descendents is denoted as $leaves(p)$ for the ancestor p. Leaves of level ≤ 1 are ignored.

Tree Rebalancing Although the error measure should ensure a balanced but precise refinement, the grid can nonetheless be overfitted towards certain areas. This could occur for example if there is a preponderance of data in one area together with small limit of grid points or a high error tolerance. The algorithm would then refine towards the area with the most data while neglecting other areas and run out of allowed grid points or pass the error threshold before these other areas can be refined. To counteract such imbalances, an algorithm to rebalance the grid was introduced [13]. This algorithm is based on the balancing of binary search trees. See figure 2.7 for an example. For the sparse grid this means moving level 1 from the center of the domain towards the boundary. This is applied recursively through the hierarchy to "flatten" it, i.e. decrease the maximum level. A safety factor s is included to prevent premature rebalancing, which would result in redundant back and forth rebalancing [13].

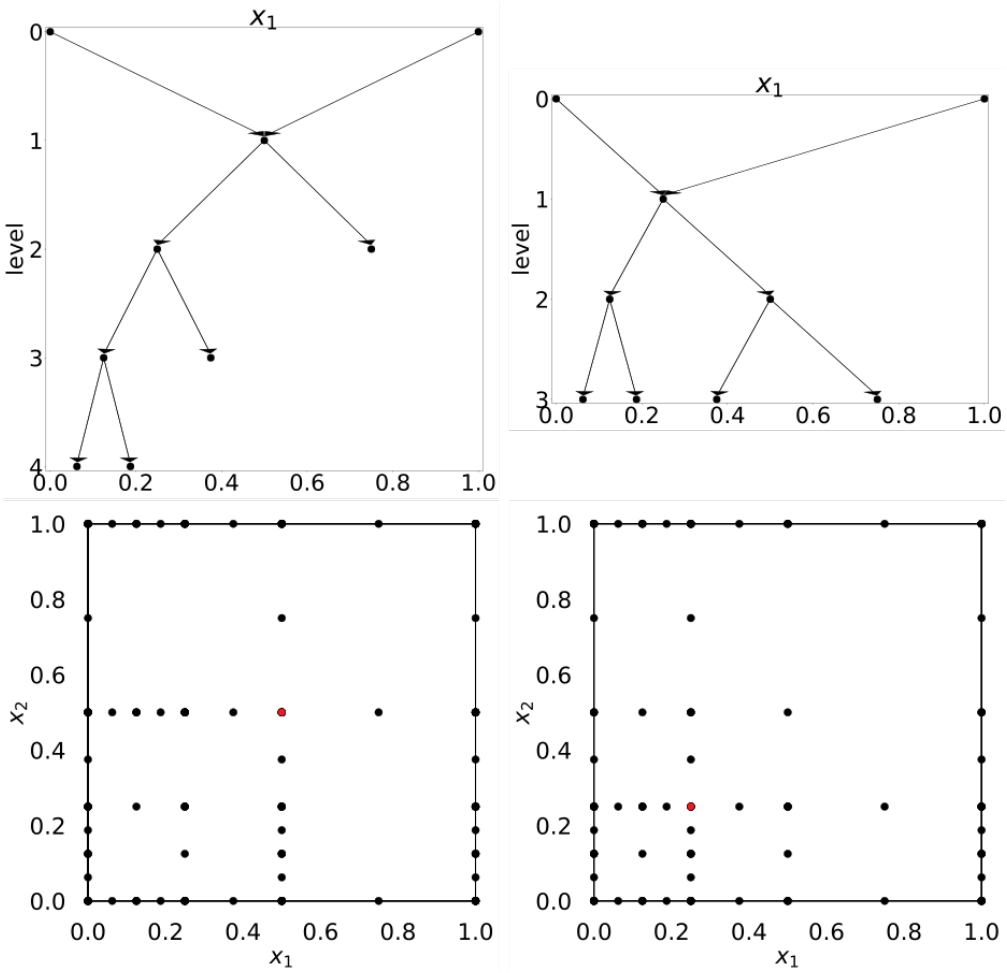


Figure 2.7: Rebalancing of the refinement trees. Left: Initial refinement tree (equal for both dimensions) with clear unbalance and corresponding sparse grid below. Right: Refinement tree after rebalancing (for both dimensions) and corresponding sparse grid below. The point in the subspace with level vector $\mathbf{l} = (1, 1)$ is marked in red for both situations[13].

2.2 Density Estimation

Density estimation is the act of estimating the probability density function f underlying a data set $S = x_1, \ldots, x_M \subset \mathbb{R}^d$. Generally, density estimation approaches can be distinguished between parametric and non-parametric estimation [17][11][1]. Parametric estimation relies on known or assumed properties of the underlying density. Such an assumption might be that the density can be described using a mixture or combination of standard density functions like the Gaussian or binomial distribution. Non-parametric approaches do not rely on such knowledge or assumptions, which makes them more suited in cases where aforementioned properties are not known or can not be reasonably assumed. These approaches still rely on so-called hyperparameters - the learning rate in a neural network for example - and can be quite sensitive to the tuning of these hyperparameters to achieve good results [1].

Kernel Based Density Estimation

In kernel based density estimation, which is a common type of non-parametric density estimation[9], the estimation $\hat{f}$ of the underlying density is constructed by placing a non-negative kernel function K onto the data points x_i :

$$\hat{f} = \frac{1}{M} \sum_{i=1}^{M} K\left(\frac{x - x_1}{h}\right). \tag{2.30}$$

A common choice for a kernel function is the Gaussian kernel [16]

$$K(x) = (2\pi)^{\frac{-1}{2}} e^{\frac{-x^2}{2}}. \tag{2.31}$$

The parameter h is the smoothing coefficient, also called the bandwidth. The bandwidth of a kernel determines the "reach" that the kernel placed on a data point has to its surrounding. See Figure 2.8 for a comparison of two densities estimated with low and high bandwidth, respectively. The performance of the kernel density estimator is influenced by the choice of the kernel function K and the bandwidth h. Furthermore, the evaluation of $\hat{f}$ depends on the number of data points M in the data set S. Density estimation with sparse grids on the other hand is dependent on the number of grid points, which are usually far fewer than the data points.

Density Estimation with Sparse Grids

For density estimation with spare grids the basis of the approximation is the basis function ϕ centered on the grid points. The general idea is to start with an initial guess

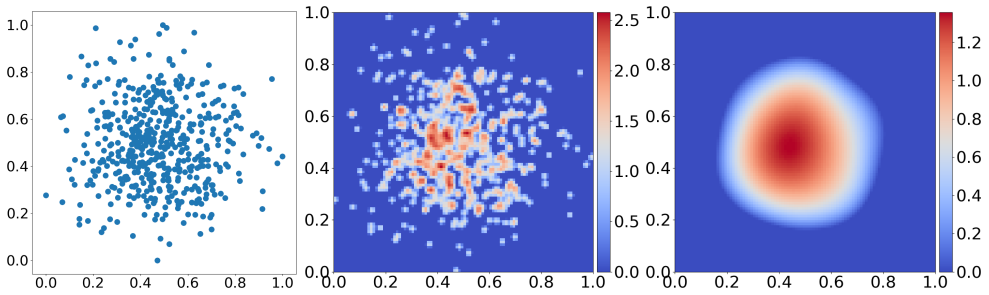


Figure 2.8: Kernel density estimation of a gaussian. On the left is the data set to be estimated, the middle shows the kernel density estimation with a bandwidth of 0.01, the right with bandwidth 0.1.

f_e with

$$f_\epsilon := \frac{1}{M}\sum_{i=1}^{M}\delta_{x_i}, \tag{2.32}$$

where δ_{x_i} is the Dirac delta function which is centered on the data points x_i, similar to how the kernel is placed in kernel density estimation. This guess is then refined using spline smoothing to obtain a better approximation $\hat{f}$ such that

$$\hat{f} = \arg\min_{f\in V}\int_\Omega (f(x) - f_e(x)^2)dx + \lambda \left\|\Lambda f\right\|_{L^2}^2 \tag{2.33}$$

Here the regularization term $\|\Lambda f\|_{L^2}^2$ is used to control the smoothness of the resulting density function, while the coefficient $\lambda > 0$ controls the balance between accuracy and smoothness. Λ represents a differential operator. with $W^{(1)}$ as the set of basis functions of the sparse grid space $V^{(1)}$, a function $\hat{f}(x) \in V^{(i)}$ is to be determined so that

$$\int_\Omega f(x)\phi(x)dx + \lambda\int_\Omega \Lambda f(x)\cdot\Lambda\phi(x)dx = \frac{1}{M}\sum_{i=1}^{M}\phi(x_i) \tag{2.34}$$

holds for all $\phi \in W^{(1)}$. This equation can be expressed as the system of linear equations

$$(R + \lambda C)\boldsymbol{\alpha} = \boldsymbol{\beta}, \tag{2.35}$$

with $R_{ij} = (\phi_i, \phi_j)_{L_2}$, $C_{i,j} = (\Lambda\phi_i, \Lambda\phi_j)_{L_2}$ and $\beta_i = \frac{1}{M}\sum_{j=1}^{M}\phi_i(x_j)$. Here $(\cdot,\cdot)_{L_2}$ denotes the standard L_2-inner product of two basis function on the domain Ω^d:

$$(\phi_{\mathbf{l},\mathbf{i}}, \phi_{\mathbf{l},\mathbf{i}})_{L_2} = \int_{\Omega^d}\phi_{\mathbf{l},\mathbf{i}}(\mathbf{x})\phi_{\mathbf{l},\mathbf{i}}(\mathbf{x})d\mathbf{x} \tag{2.36}$$

By choosing a suitable regularization operator Λ, the system of linear equations can be simplified to

$$(R + \lambda I)\boldsymbol{\alpha} = \boldsymbol{\beta}, \tag{2.37}$$

where C has been replaced with the identity matrix I. While this regularization operator no longer preserves moments, these properties are not needed for estimating density functions [16]. Solving the system of equations 2.37 for $\boldsymbol{\alpha}$ gives the coefficients of the basis functions in the grid. Since the calculation of the R-matrix depends solely on the sparse grid and is independent of the data set, the calculaton of the R-matrix and the β-vector can be split into a separate online/offline phase. In the offline phase, the matrix $R + \lambda I$ can be pre-calculated and stored for reuse in other (component-)grids. In the online phase the system of equations 2.37 is solved. This reduces the complexity of solving 2.37 from $O(N^3)$ to $O(N^2)$ [17].

2.3 Classification

Classification denotes the separation of data into classes based on one or more criteria. These criteria can include various feature or patterns of the data points. Usual approaches for classification are either supervised or unsupervised. In supervised approaches, the algorithm or method in question is given a training set, usually a partial subset of the complete data set with class labels, to learn the features and patterns of the data set.
The goal is then to construct a function $\hat{p} : \mathbb{R}^d \to K$ that assigns class labels from the the set of classes $K = \{1, \ldots, k\} \subset \mathbb{N}$ to the data points [1] [5] [11]. Classification with sparse grid density estimation can be accomplished with two approaches, one based on estimating the densities of the classes themselves and one based on estimating the boundaries between classes.
The first approach starts by separating the training data based on classes. Then the density of each class is estimated separately, each with their own grid[15]. The classification of the test data is performed by evaluating the data points from the test data with each grid and assigning the point to the class whose grid evaluated with the highest value. Figure 2.9 gives an example for such a classification with the two moon data set.
The second approach uses a binary one-vs-others split of the training data instead. As before, the training data is not separated per-se, but data points not in the class to be estimated are given the same label to designate as the "other" in the one-vs-other approach. The resulting densities consist of "hills" and "valleys", with the hills being the estimated class in question and the valleys being the regions of the other classes. Determining the class label for the test data is done by evaluating on the grid and choosing the highest value, as before.

An advantage of using this approach is that adaptive methods, like the dimension-wise adaptive refinement, allows focusing specifically on the regions with class boundaries. This can be accomplished by using a misclassification based error estimator for the refinement, because it gives a higher error in regions where class boundaries meet/overlap. An example for the one-vs-others approach is in figure 2.9.

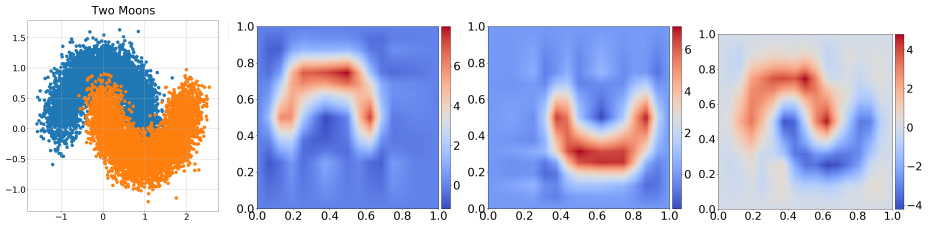


Figure 2.9: Single class and one-vs-others classification of a two moons data set. The leftmost figure is the data set, the rightmost figure is the one-vs-other classification. The middle figures are the single class density estimations.

3 Implementation

In this section the Framework used will be covered. Firstly, a general introduction of the SparseSpACE and its capabilites. Secondly, the new additions to the Framework will be explained in-depth with pseudo code for better illustration of the functionality.

3.1 SparseSpACE Framework

The SparseSpACE framework [22] - created by Michael Obersteiner et al. [13] - is a modular, Python based framework for working with a variety of Sparse Grid algorithms. Currently implemented methods include the Sparse Grid Combination Techniques in standard and adaptive configurations, featuring advanced adaptive algorithms like Single Dimension Adaptivity - which was introduced in [13] - and the split-extend algorithm introduced in [14]. The standard combination technique, as well adaptive methods, such as the single dimensional adaptive approach introduced in [13] and the split-extend method introduced in [14], can be combined with a variety of operations and error measures for the adaptive algorithms.
Operation options currently include integration, interpolation and density estimation, the latter of which was previously implemented for the standard Combination technique in [20] and was expanded to also work with dimension wise refinement for this thesis.
Error measures for the adaptive algorithms include variations of volume-, surplus and misclassification based ones, where the misclassification based ones have been implemented for this thesis and will be explained in more detail in subsection 3.2. The framework also supports several different grid types like `Trapezoidal-`, `ClenshawCurtis-` or `GaussLegendreGrids`. These grids provide useful methods for implementing density estimation, such as checking whether a point lies on the boundary of the grid or obtaining the coordinates of a point. For the implementation of density estimation for the dimensionally adaptive algorithm, a global trapezoidal grid was the solution most suited within the given timeframe.
A recent addition by Cora Moser [10] is a manager class for the purpose of handling classification within the framework. It allows easy execution of multi-class classification with the standard combination technique. For this thesis, the manager was extended to allow usage of single dimensional refinement, as well as using a binary one-vs-others

approach in the density estimation. For a more comprehensive look at this classification management, refer to [10]. Furthermore, with the help of Obersteiner the calculations for the linear system in 3.1 have been vectorized using NumPy [12].

3.2 Density Estimation for dimension-wise Refinement

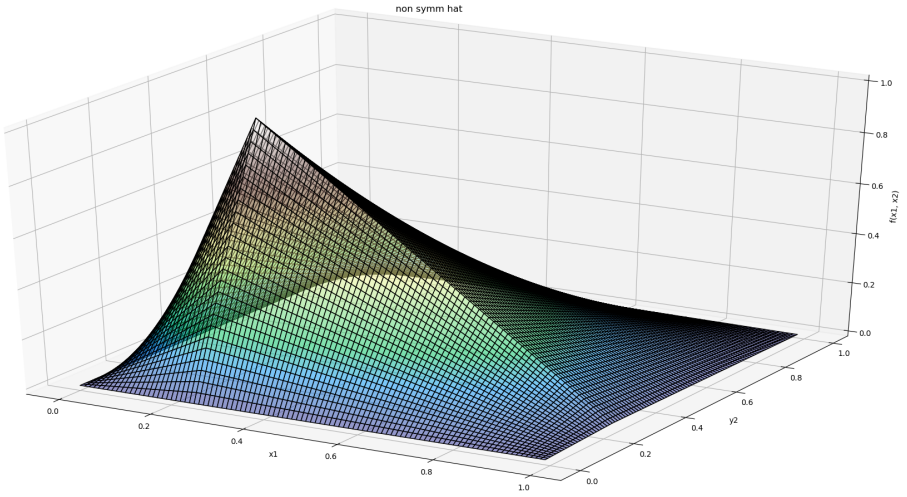


Figure 3.1: Contour plot of a 2-dimensional non-symmetric hat function. The domains in each dimension are $([0,1],[0,1])$, with the function centered on $(0.25, 0.25)$

In the case of dimension-wise refinement, the algorithm for density estimation works very similar to the standard combination technique with a few differences. The main effort is again in obtaining the solution for the linear System

$$(R + \lambda I) = \boldsymbol{\alpha}. \tag{3.1}$$

Since the points on the component grids are not necessarily equidistant, the basis function centered on the points and spanning to the neighboring points are not guaranteed to be symmetric. An example of such an asymmetric function is in figure 3.1, depicting the asymmetric hat function in 2D.
While calculating the values of the R matrix using the inner product

$$R_{ij} = (\phi_i, \phi_j)_{L_2} \tag{3.2}$$

remains unchanged, obtaining the supports has become more complicated. Unlike with the standard combination technique, the supports can no longer be obtained by using the meshsize, but must instead be explicitly found. This is achieved by determining the coordinates of the neighboring points and by determining the domains of the grid points involved in the calculation of each R-value. Given the set of grid point coordinates $\mathbf{C} \in \mathbb{R}^d \times \mathbb{R}^{|l_d|}$, where $|l_d|$ denotes the number of grid points for the given level of the component grid in the respective dimension. The domains for a single grid point $\mathbf{p} \in \mathbf{C}$ can be obtained by building the set of domains in each dimension $\mathbf{D}$ with

$$D_i = \{[a,b] \mid a = \arg\max_{c_i}(0, c_i),\ c_i \in \mathbf{C}_i,\ c_i \leq p_i,\ b = \arg\min_{c_i}(1, c_i),\ c_i \in \mathbf{C}_i,\ c_i \geq p_i\} \tag{3.3}$$

Where p_i is point $\mathbf{p}$'s coordinate in the i-th dimension. The neighbors of $\mathbf{p}$ are simply the coordinates at the boundary of the domain of $\mathbf{p}$ that are also in the set of grid point coordinates C. They are obtained by building the set

$$\mathbf{N} = \{\mathbf{n} \mid \mathbf{n} = (n_1, \ldots, n_d)^T,\ a_i \leq n_i \leq b_i,\ [a_i, b_i] \in D_i,\ \mathbf{n} \in \mathbf{C},\ \mathbf{n} \neq \mathbf{p}\}, \tag{3.4}$$

where D_i are the domains of the point $\mathbf{p}$ in each dimension. Calculating the inner product is also changed, since the basis function is sometimes non-symmetric, but since the non-symmetric hat function is linear, it can still be easily solved analytically. The following equation shows the integral used in calculating the inner product for a non-symmetric hat function:

$$\int_b^c \int_a^b \phi_i(x)\phi_j(x)dx = \begin{cases} \gamma_a(b, m_i, b) - \gamma_a(a, m_i, b) + \gamma_b(c, m_j, b) - \gamma_b(b, m_j, b) & \text{full overlap} \\ \rho(b, m, a, b) - \rho(a, m, a, b) & \text{partial overlap} \\ 0 & \text{no overlap} \end{cases} \tag{3.5}$$

where

$$\rho(x, m, p, q) = \frac{m^2x^2(p+q)}{2} - \frac{m^2x^3}{3} - x(mp+1)(mq-1) \tag{3.6}$$

$$\gamma_a(x, m, p) = \frac{(m(p-x)-1)^3}{3m} \tag{3.7}$$

$$\gamma_b(x, m, p) = \frac{(m(p-x)+1)^3}{3m} \tag{3.8}$$

The framework now also gives the option of calculating the integral of the inner product numerically. Algorithm 3.2 shows the construction of the R matrix modified for the dimension wise refinement case.

Calculation of the $\boldsymbol{\alpha}$-vector is virtually unchanged. As with the standard combination,

the value of each element is obtained by summing up the contributions of each data point x_j in the data set and scaling by $\frac{1}{M}$, where M is the size of the data set:

$$b_i = \frac{1}{M} \sum_{j=1}^{M} \phi_i x_j \tag{3.9}$$

The only difference here is that ϕ_i uses a non-symmetric version of the basis function if the domain of grid point i is not symmetric. See 3.1 for the complete algorithm.

Reusing old values Since the changes for building the R matrix add considerable overhead by having to explicitly determine the domain of each grid point involved, a few performance optimizations were implemented.
By exploiting the fact that, firstly, only smaller and smaller regions within the Grid get refined with each iteration, secondly that the basis functions are centered on the grid points, thirdly they often have the same domain size and shape and fourthly, that the integral of the inner product of two basis functions is invariant to rotation, a large amount of values computed in previous iterations and for component grids within the same iteration can be reused.
For reusing values for the calculation of the R matrix, the domain overlap of the functions and its neighbors need to be determined pairwise. If the hat function is symmetrical, the overlap between all neighbors within the same manhattan distance along the dimension wise grid point indices are the same. Even if the domain is non-symmetric, the integral of the inner product is still invariant to rotation, meaning that all grid point pairs whose distances in each dimension are a permutation of each other can be reused. This means that for every grid point, up to $2^d - 1$ fewer calculations need to be made. This is especially useful, if the R-values need to be calculated numerically.
For reusing values calculated for the $\boldsymbol{\alpha}$-vector, the regions affected by the last iterations refinement need to be determined. Any $\boldsymbol{\alpha}$-values within or neighboring a new point, need to be recalculated, while the rest can be reused. To avoid running through the entire data set and having to check if each point is within the affected domain, d-many sorted index sets - where d is the dimensionality of the grid - were used to allow narrowing the data set to points only within the affected domain.
The old α-values are stored per component grid in a python dictionary, and indexed by the coordinate stripes of their grid points, where coordinate stripes refers to the d-many vectors $\mathbf{c}^{(d)} = (c_0^{(d)}, \ldots, c_{2^l-1}^{(d)})^T$ containing the coordinates of each grid point for each single dimension.
To determine the closest matching point set, the coordinate stripes of the new and old component grids are compared in each dimension and any points not in the old

point set are collected in a point difference set. The coordinate stripe set that produced the point difference set with the lowest magnitude denotes the closest matching set. Algorithm 3.4 gives the pseudo code for this procedure.
The dictionary with old b values gets updated with every refinement step, with only the values of the previous refinement step being saved, while older values are discarded.

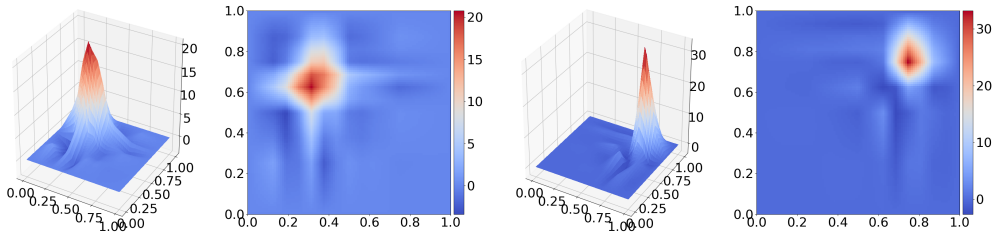


Figure 3.2: Density estimation for two separate Gaussians within the same domain. Note the value increases of the z-axis from left to right.

One-vs-others density estimation Another addition is the a binary one-vs-others split for density estimation. The class labels are hereby separated into $\{1, -1\}$ to be used as signs for the calculation of the $\boldsymbol{\alpha}$-vector of 3.1. When working with the classification manager, the label/sign of the "others" set O is further weighted by the ratio $\frac{\|C\|}{\|O\|}$, where $\|C\|$ is the number of samples for the class to be estimated and $|O|$ the number of samples for the other classes, changing the label set to $\{1, -\frac{\|C\|}{\|O\|}\}$. See Algorithm 3.1 for when this (weighted) sign is used in the calculation.
The main purpose of the one-vs-others split is to mitigate misclassifications at coarse resolutions, near class boundaries or lopsided class distributions. Since the class assignment for a sample is done by selecting the maximum evaluation of the respective grid classifier, a sample can be mislabeled, because the domain of a grid point may extend past the class boundary and into the neighboring class density. Which density has the higher value in that case comes down to which class has a higher amount of samples in that specific region. The values of the density increase overall with larger and larger amounts of data - see Figure 3.2 - which means that for lopsided class distributions, this problem can become more pronounced if the classes are not separated well enough. The one-vs-others split mitigates this problem by steepening the slopes near class boundaries.

Misclassification error measure A new error measure for refinement based on the misclassification rate of a validation set was added for this thesis. Usage of this error measure requires class labels to be passed along with the data set and the one-vs-others flag of the density estimation to be set.
At the start of each refinement, a validation set is split off from the training data. The validation set is sampled randomly from the training data, with an equal amount of samples taken for the class to be estimated and the "others" label. At the end of the refinement the validation set is added back into the training data. Subsequent refinement steps sample a new validation set each time.
To calculate the misclassification rate, the validation set is first evaluated on the current grid. Then all samples falling within the currently considered (sub-)domain are labeled based on the sign of their evaluation. The labels are compared to the ground truth and the number of wrongly labeled points are summed up. The final sum then weighted by the size of the currently considered (sub-)domain.

Algorithm 3.1 calculate α vector

P $\leftarrow$ Grid Points
$\mathcal{P} \leftarrow$ Old Grid Points
$M \leftarrow |Data|$
$N \leftarrow |GridPoints|$
$\boldsymbol{\alpha} \leftarrow (0, \ldots, 0)^T \in \mathbb{R}^N$
$\hat{\boldsymbol{\alpha}} \leftarrow$ old_α_vector

point_domains $\leftarrow$

$$\{[a,b] \in \mathbb{R}^d |$$
$$a = \underset{|p_d - p'_d|}{\arg\min}(\{p' | p'_d <= p_d, p' \in P, p \in P\}, d \in [0, \ldots, \dim]),$$
$$b = \underset{|p_d - p'_d|}{\arg\min}\left(\{p' | p'_d >= p_d, p' \in P, p \in P\}, d \in [0, \ldots, \dim]\right)\}$$

old_point_domains $\leftarrow$

$$\{[a,b] \in \mathbb{R}^d |$$
$$a = \underset{|p_d - p'_d|}{\arg\min}\left(\{p' | p'_d \leq p_d, p' \in P, p \in P\}, d \in [0, \ldots, \dim]\right),$$
$$b = \underset{|p_d - p'_d|}{\arg\min}\left(\{p' | p'_d \geq p_d, p' \in \mathcal{P}, p \in \mathcal{P}\}, d \in [0, \ldots, \dim]\right)\}$$

for $i \in [0, \ldots, N]$ **do**
  $dom_i \leftarrow$ point_domain$_i \in$ point_domains
  if $dom_i \in$ old_point_domains **then**
    $\alpha_i = \hat{\alpha}_i$
  end if
end for
for $i \in [0, \ldots, N]$ **do**
  if $\alpha_i = 0$ **then**
    for $x \in M$ **do**
      $\alpha_i = \alpha_i + \phi_i(x) \cdot sgn(x)$
    end for
  end if
end for

Algorithm 3.2 calculate R matrix

N ← number_of_grid_points
$R \leftarrow ((0,\ldots,0)^T,\ldots,(0,\ldots,0)^T) \in \mathbb{R}^d \times \mathbb{R}^d$
$R' \leftarrow$ old_R_values
for $i \in [0, N]$ **do**
  for $j \in [i, N]$ **do**
    overlap ← get_overlap(p_i, dom_i, p_j, dom_j)
    if overlap $\in R'$ **then**
      $R_ij \leftarrow R'(overlap)$
      $R_ij \leftarrow R'(overlap)$
    else
      $r \leftarrow$ integral(p_i, dom_i, p_j, dom_j)
      $R_ij \leftarrow r$
      $R_ji \leftarrow r$
      $R'(overlap) \leftarrow r$
    end if
  end for
end for

Algorithm 3.3 calculate overlap between grid point domains

N ← number_of_grid_points
$G =$ Grid point set
$p_i \leftarrow (p_{i,1},\ldots,p_{i,d}) \in G$
$p_j \leftarrow (p_{j,1},\ldots,p_{j,d}) \in G$
$dom_i \leftarrow (a_i, b_i)$, where (a_i, b_i) is the domain of ϕ_i centered on p_i
$dom_j \leftarrow (a_j, b_j)$, where (a_i, b_i) is the domain of ϕ_j centered on p_j
widths ← {}
distances ← {}
for $m \in [0,\ldots,d]$ **do**
  w ← $max(a_i, a_j) - min(b_i, b_j)$
  x ← $|p_{i,m} - p_{j,m}|$
  widths.append(w)
  distances.append(x)
end for
sort(widths)
sort(distances)
return (widths, distances)

Algorithm 3.4 Find closest old α vector

$C \leftarrow$ old_grid_coordinate_stripes
$\hat{C} \leftarrow$ new_grid_coordinate_stripes
dim $\leftarrow$ grid_dimensions
for $\hat{c} \in [0, \ldots, |\hat{C}|]$ **do**
  for $d \in [0, \ldots, |\text{dim}|]$ **do**
    for $i \in [0, \ldots, |\hat{c}|]$ **do**
      if $\hat{c}_i \notin C$ **then**
        $\text{new\_points}_{c,d,i} = c_i$
      end if
    end for
  end for
end for
differences $\leftarrow \{|\text{p}| \mid \text{p} \in \text{new\_point\_set}, \text{new\_point\_set} \in \text{new\_points}\}$
closest_match $\leftarrow \underset{|diff| \in differences}{\arg\min} (differences)$
return old_α_values(closest_match

4 Evaluation

This chapter will cover evaluations on model and real world data. The dimension-wise density estimation is evaluated based on a variety of criteria. Firstly, how well the estimated density matches the given model data in shape compared to the standard combination technique and kernel density estimation and how it compares to kernel density estimation within various metrics, like average sample difference, KL-divergence and Pearsson correlation. Secondly, a classification based evaluation. The single dimensional refinement method is compared against the standard combination technique with a variety of data sets, like the two moon data set, Gaussian quantiles and the iris data set. The results are evaluated based on accuracy for the amount of points used and the runtime for the amount of points used.

4.1 Density Estimation

4.1.1 Cross data set

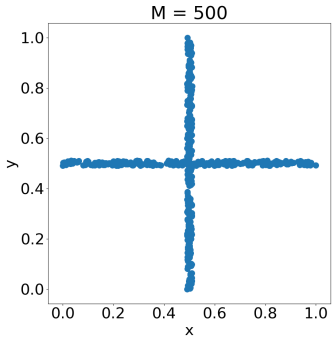


Figure 4.1: Cross data set.

The cross data set is generated with "lines" by using a uniform density in each dimension, with parameters $[a, b] = [0.45, 0.55]$ and $[a, b] = [0, 1]$, respectively. The Cross data set is simply two different line sets overlapping. The purpose is mainly to test the adaptivity of the single dimensional refinement approach. The standard

combination and kernel density estimation act as control and comparison. Figure 4.1 shows line and cross data set, Figures 4.2, 4.3 and 4.4 show the results for the standard combination technique, the single dimensional refinement and kernel density estimation, respectively.
The standard combination technique performs very well. One can see that the even distribution of points across the domain is a bit wasteful, since the data is only within an narrow range of the domain. This is expected behaviour, since the standard combination technique has no adaptive capabilities. The total number of grid points used (including all component grids) for level 5 is 273 and 95 for level 4.
As Figure 4.3 shows, the single dimensional refinement has no trouble adapting to this simple data set. Since the cross is simply two lines overlapped, the density of points in the overlap is roughly double that of the rest of the domain covered by a single line, which results in a higher density of grid points in the center. With the last refinement of the dimension-wise method, the grid consists of 166 points. Comparing the grid with level 4 and 5 of the standard method, the dimension-wise method offers a good compromise between the level 5 and 4.
The kernel density estimation gives good results as well. The only caveat is, that the bandwidth of the kernel may need to be increased, if the data set is too sparse, to obtain a sufficiently smooth approximation. Figure 4.4, left, shows what happens when the bandwidth is set too low. Setting the bandwidth too high on the other hand can result in an over-smoothed approximation, which can be seen in Figure 4.4 on the right.

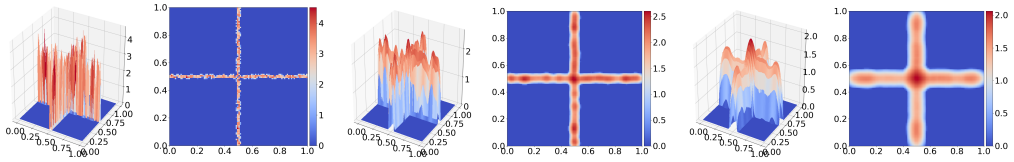


Figure 4.4: Density estimation for the cross data set with KDE. From left to right the kernel bandwidths are 0.01, 0.05 and 0.1.

$\begin{pmatrix}\text{Avg sample difference}\\ \text{KL-divergence}\end{pmatrix}$	kde bandwith=0.01	kde bandwith=0.05	kde bandwith=0.1
dimension-wise	$\begin{pmatrix}-\\-\end{pmatrix}$	$\begin{pmatrix}0.075\\0.119\end{pmatrix}$	$\begin{pmatrix}0.048\\0.106\end{pmatrix}$

Table 4.1: Average sample difference and KL-divergence between the solution of dimension-wise method and the kernel density estimation. Pearson correlation is not applicable for this data set. Lower average sample distance and KL-divergence indicate a closer match to the kernel density estimation.

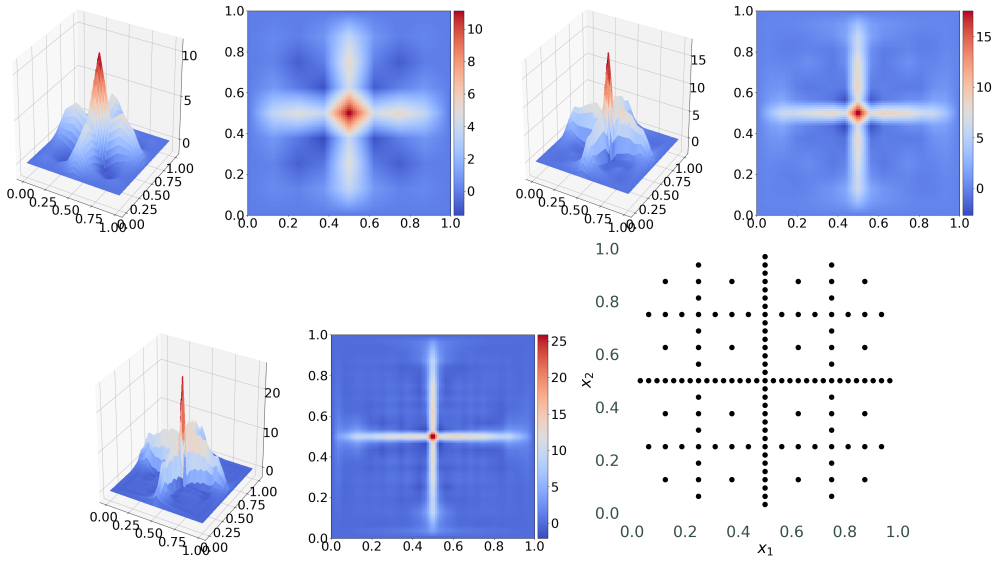


Figure 4.2: Density estimation for the cross data set with the standard combination technique with max levels 3 to 5 and the final grid for level 5.

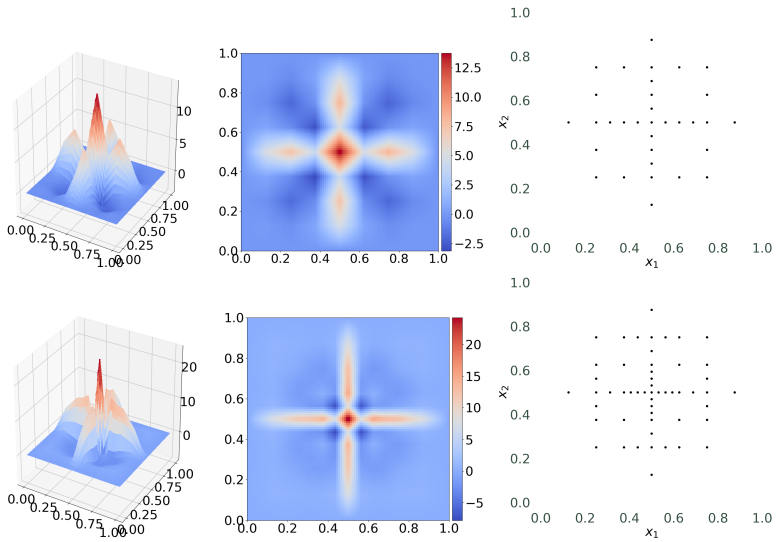


Figure 4.3: Density estimation for the cross data set with the dimension-wise spatially adaptive method for the first and second refinement.

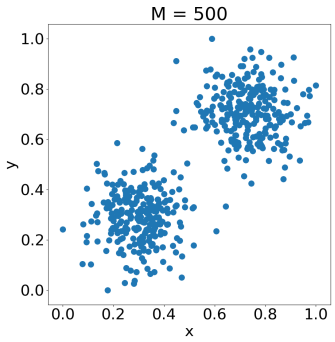


Figure 4.5: Two Gaussian data set.

4.1.2 Two Gaussians

Two Gaussian distributions with the same standard deviation, but differing means are generated and used as a single dataset. The number of samples drawn is 250 for each Gaussian, for a total of 500. This setup is to explore how the dimension-wise spatial refinement behaves when the density is split up into multiple localities.
The standard combination technique performs well in capturing the shape and boundary of the Gaussians, as can be seen in figure 4.6. Level 3 already captures the shape of the Gaussians. Further refinement merely refines the boundary of the density. The drawback here is that there are a lot of points in the upper left and lower right corner that are wasted, since there is not much data there.
Since the dimension-wise refinement is adaptive, one would expect the grid points to be distributed more local towards the data compared to the standard combination technique. Looking at figure 4.7, some local refinement can be observed in the component grid plots, but the final sparse grid depicted Figure 4.7 reveals, that this refinement is not concentrating on the upper right and lower left corners, where the data is located. Since the granularity of the refinement is only in the dimension and not in the location itself and because of the parent node requirements for valid sparse grids, the final grid ends up very similar to the standard combination technique.
Still, the final grid 4.7 appears a bit lopsided towards one side, but repeating the same trial with newly generated data might end up with the grid leaning towards the other side, depending on how the drawn samples are distributed for each Gaussian. As for the amount of points used, the final grid of the dimension-wise method has 177, a middle ground between the 95 and 273 points of level 4 and 5 of the standard method. Since level 3 already captures the shape of the Gaussians fairly well, increasing to level 4 is only worth it if the boundary of the Gaussians is of importance. The 177 points used by the dimension-wise method are already too much for such a simple data set.

As one would expect, the kde with Gaussian kernel has no problem estimating Gaussians, as can be seen in Figure 4.8. As before, the choice of bandwidth relative to the amount of samples greatly influences the smoothness of the final estimation. In this case a bandwidth of 0.1 captures the original distributions the best.

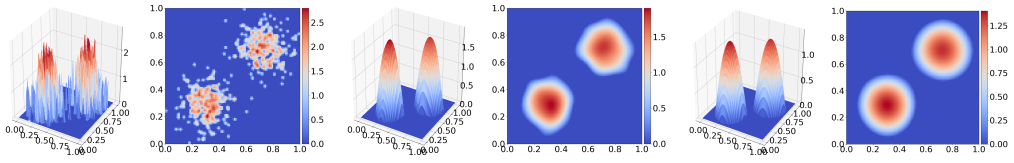


Figure 4.8: Density estimation for the two Gaussians data set with KDE. From left to right the kernel bandwidths are 0.01, 0.05 and 0.1.

$\begin{pmatrix}\text{Avg sample difference}\\ \text{Pearsson correlation}\end{pmatrix}$	kde bandwith=0.01	kde bandwith=0.05	kde bandwith=0.1
dimension-wise	$\begin{pmatrix}0.261\\0.476\end{pmatrix}$	$\begin{pmatrix}0.039\\0.847\end{pmatrix}$	$\begin{pmatrix}0.023\\0.849\end{pmatrix}$

Table 4.2: Average sample difference and KL-divergence between the solution of dimension-wise method and the kernel density estimation. KL-divergence is not applicable for this data set. Lower average sample distance and high Pearson correlation indicate a closer match to the kernel density estimation.

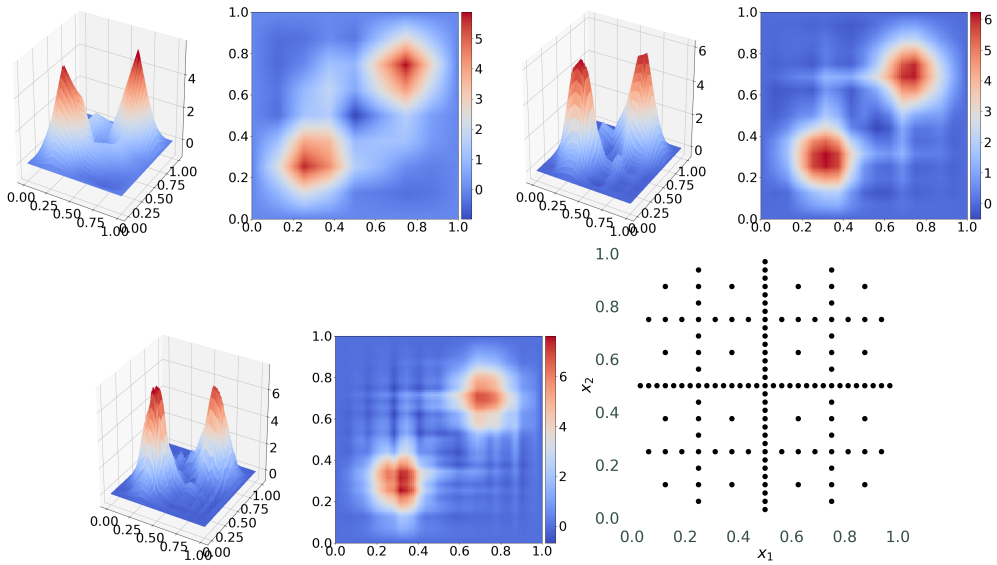


Figure 4.6: Density estimation for the two Gaussians data set with the standard combination technique with max level from 3 to 5 and the final grid for level 5.

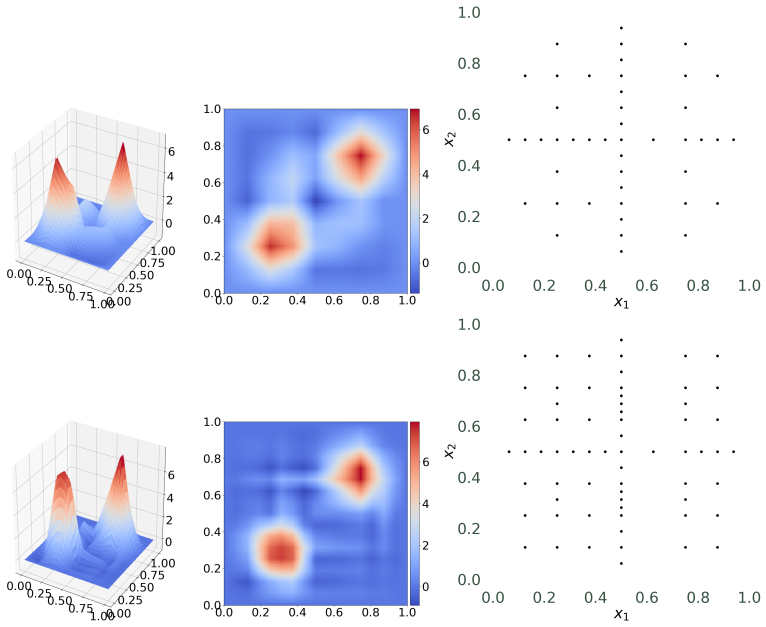


Figure 4.7: Density estimation for the two gaussians data set with the dimension-wise spatially adaptive method for the first and second refinement.

4.1.3 Circle

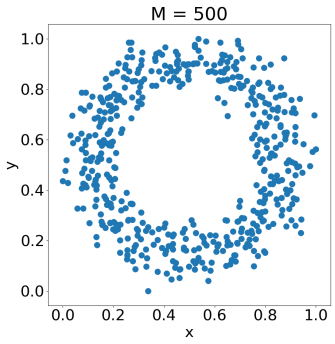


Figure 4.9: Cirle data set.

The circle data set was created with scikit-learn, again with a sample size of 500 data points. This data set presents much greater challenge for the standard combination technique and the dimension-wise method compared to the previous ones. This is because the grid based approach is less well suited for non-axis aligned data. The donut shape introduces further complications, since there is a large area of the domain not covered, but the data is still distributed over the whole range when considering each dimension seperately. So the data requires fine local resolution to properly capture the smooth, narrow curve of the circle which is spread over the whole domain, while simultaneously large areas contain no data at all.
Figure 4.10 shows that the standard combination technique performs adequately at higher levels. At lower levels, the shape of the estimation is more square in shape than round, due to the fact that distribution of grid points themselves are, of course, grid based. With higher levels, a rounder shape for the estimation starts to emerge.
For the dimension-wise method, the contour for each refinement are depicted in Figure 4.11. The result of the dimension-wise spatially adaptive method looks very similar to standard combination technique. While there is a little bit of refinement towards the edges and a few "missing" points in the middle of the domain compared to the standard combination technique, the overall result is still fairly regular. Further refinement would grow the number of points near the edge more than the number of points at the center, but at the refinement level depicted in Figure 4.11 signs of overfitting can already be observed, so further refinement would be counterproductive.
The final count of grid points is 188 for the dimension-wise method, 273 for level 5 and 95 for level 4 of the standard combination technique. While the result for the dimension-wise method is more "rounded" in the corners compared to level 4 of the standard method, it needs more than twice as many points to achieve that result. Due

to the low amount of data points, the regular shape and coverage of the domain the standard method is the more efficient option in this case.
As for the kde estimation, given a decent choice for the bandwidth, like 0.05 or 0.1 in the middle and right of Figure 4.12, the results capture the original distribution best. Since the kernels are placed directly on the data points, the locality can be captured much more closely.

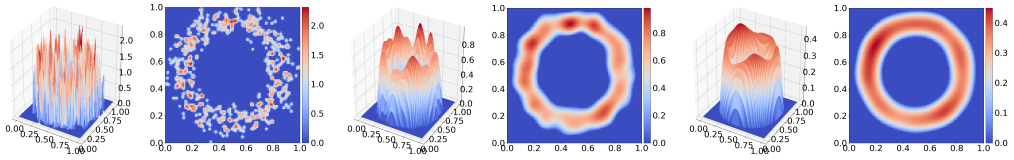


Figure 4.12: Density estimation for the circle data set with KDE. From left to right the kernel bandwidths are 0.01, 0.05 and 0.1.

$\begin{pmatrix}\text{Avg sample difference}\\ \text{Pearsson correlation}\end{pmatrix}$	kde bandwith=0.01	kde bandwith=0.05	kde bandwith=0.1
dimension-wise	$\begin{pmatrix}0.143\\0.480\end{pmatrix}$	$\begin{pmatrix}0.015\\0.816\end{pmatrix}$	$\begin{pmatrix}0.006\\0.767\end{pmatrix}$

Table 4.3: Average sample difference and KL-divergence between the solution of dimension-wise method and the kernel density estimation. KL-divergence is not applicable for this data set. Lower average sample distance and high Pearson correlation indicate a closer match to the kernel density estimation.

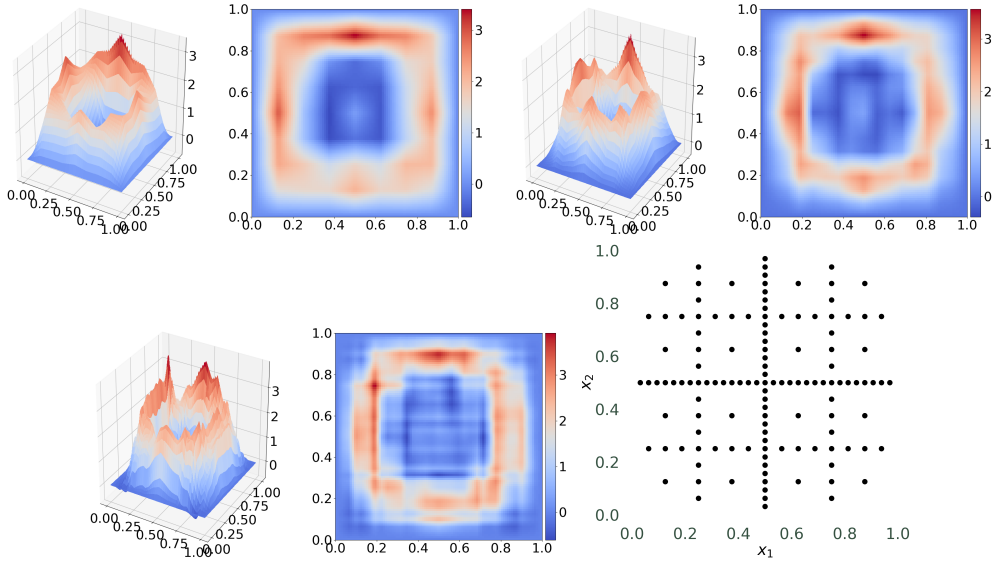


Figure 4.10: Density estimation for the circle data set with the standard combination technique with max level from 3 to 5 and the final grid for level 5.

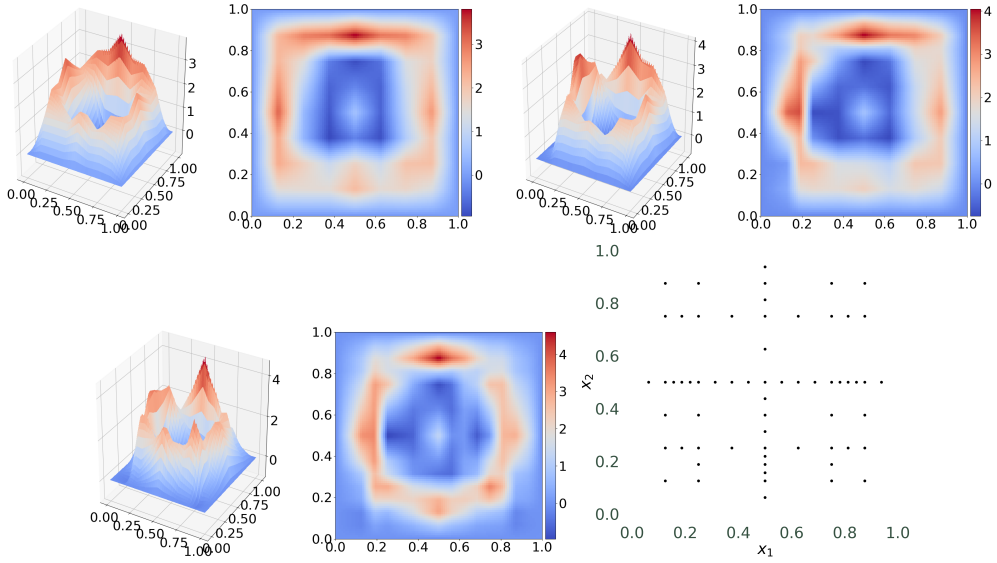


Figure 4.11: Density estimation for the circle data set with the dimension-wise spatially adaptive method for all three refinement steps and the final sparse grid.

4.2 Classification

This section covers classification evaluation with both artificial and real data sets. The Classification is done using the classification manager added by Cora Moser [10]. For each data set, both the standard combination technique and the dimension-wise method were evaluated with two different configurations. The first configuration was density estimation on a single class, with the dimension-wise method using the guided volume error measure for refinement. The second configuration used the one-vs-other approach, with the dimension-wise method using the misclassification error measure for refinement.
All data sets were also evaluated using successive max levels and point limits for the standard combination technique and dimension-wise spatially adaptive refinement, respectively. The results were examined in terms of the accuracy per number of points and the number of points generated for the runtime. Although, if the data set in question is in 2D, meaning that it can be properly visualized, these plots were omitted in favour of contour plots. Also note that the dimension-wise method was not executed iteratively, meaning that for each level increase for the standard combination technique, the dimension-wise method was redone from scratch with a new point limit. This was done for three reasons. Firstly, so that the overhead of the dimensional refinement can be observed. Secondly, for higher point limits/levels the runtime comparison favors the dimension-wise method by default since it is iterative. Thirdly, since the framework is still unoptimized, the absolute values of the runtimes are of little interest.

4.2.1 Two Moons

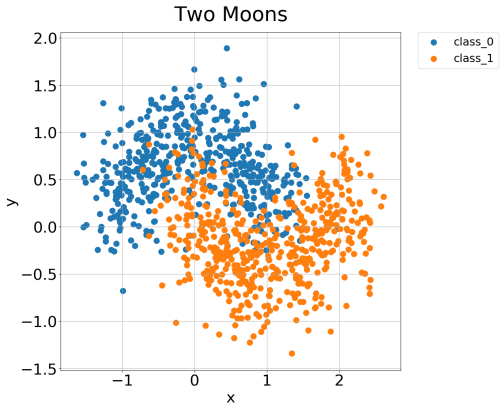


Figure 4.13: Two moons data set. The first class is colored blue, while the second class is in orange.

The two moon data set - a staple set for classification algorithms - consists of two interleaving half-circles. The set was created using the scikit learn framework with a sample size of 1000, with 500 samples for each class. 100 randomly chosen samples were split off as a test set. Figure 4.14 depicts the solution obtained with the standard combination technique with a maximum level of 4. The solutions on the top are for the first class and the bottom for the second class. The left side was obtained with the single class density estimation, while the right side used the one-vs-others approach. Figure 4.16 gives the classification results for the first configuration on the left, with the second configuration on the right.
The density estimation for the dimension-wise method are depicted in Figure 4.15. As with the standard combination technique, the first and second class are on the top and bottom, respectively, while the first configuration is on the left and the second configuration on the right. Figure 4.17 shows the classification results for the obtained solutions also with the the first configuration is on the left and the second configuration on the right.
Test accuracy for all methods and configurations was at $89\% \pm 0.5$, with the dimension-wise method in the first configuration performing the best. The number of grid points used for level 4 are, as before, 95. The dimension-wise method used 62 and 64 points for classes 1 and 2 in the first configuration. In the second configuration, the numbers increase to 90 and 74, making the dimensional refinement the better option in terms of efficiency.

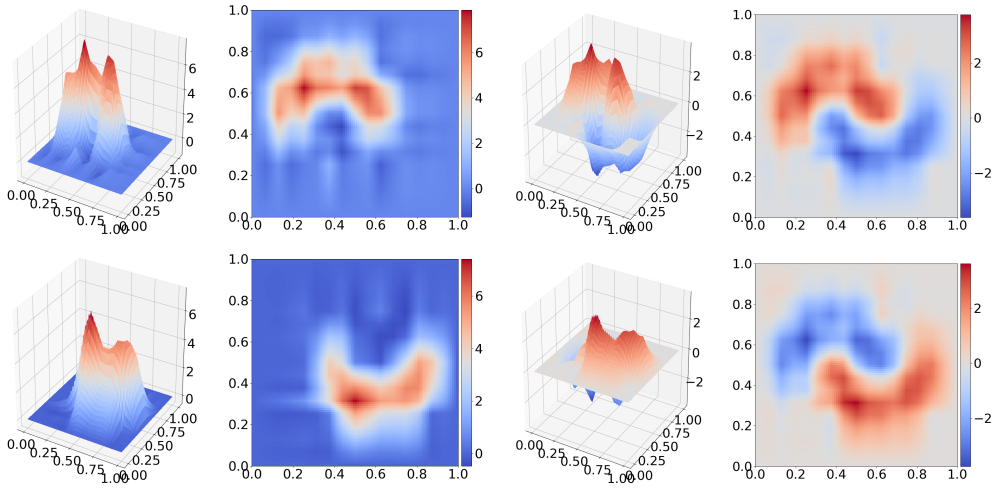


Figure 4.14: Contour plots of the standard combination solutions. One-vs-others approach with misclassification error measure is on the right, standard approach on the left.

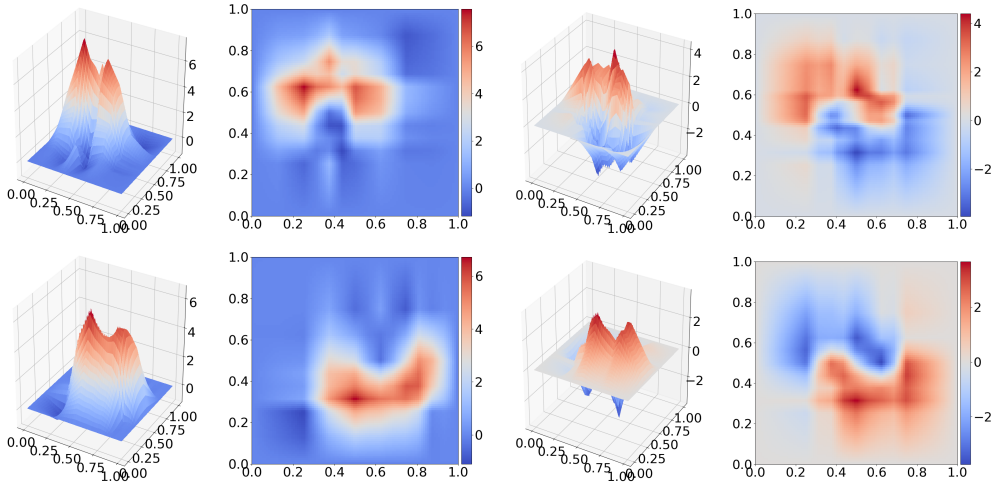


Figure 4.15: Contour plots of the dimension-wise spatially adaptive solutions. One-vs-others approach with misclassification error measure is on the right, standard approach on the left.

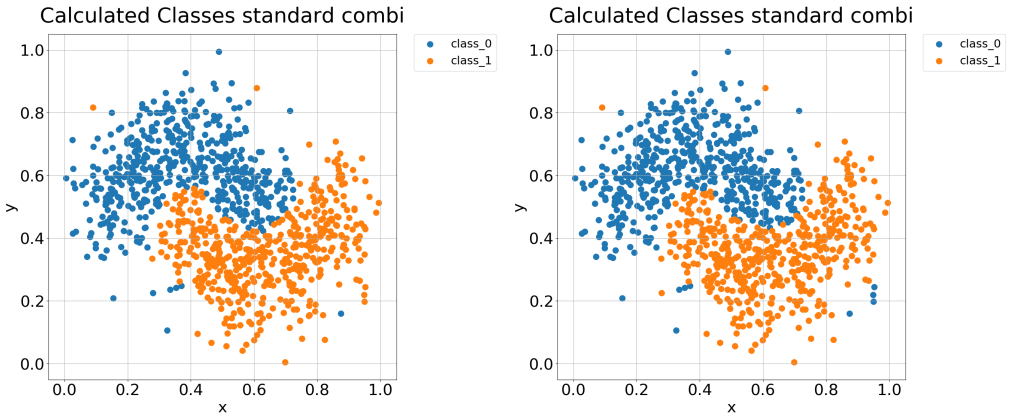
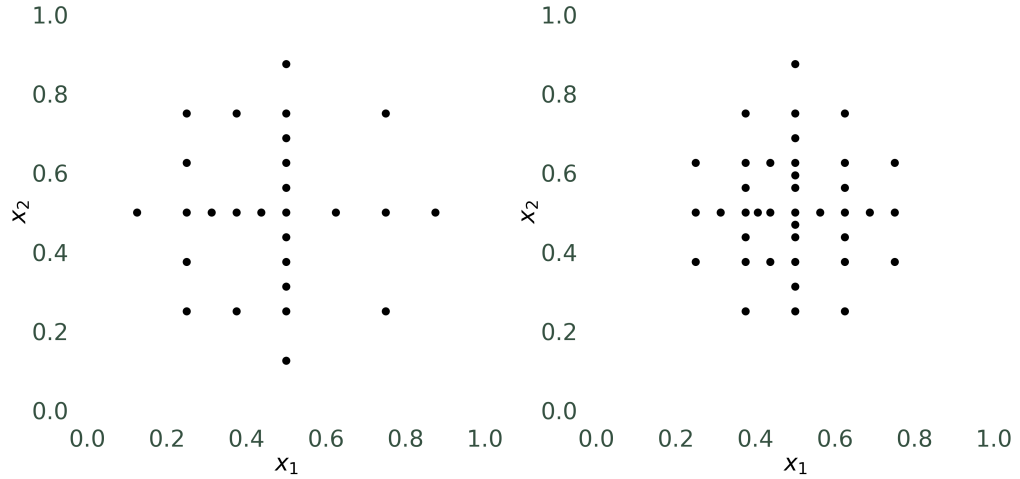


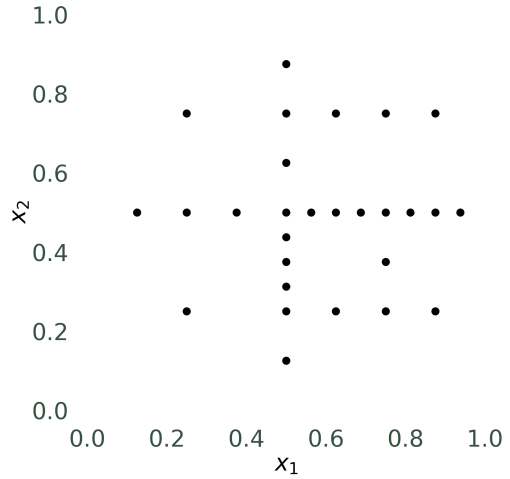
Figure 4.16: The calculated classes for the standard combination solution. The left side is for the single class density estimation, while the right is for the one-vs-others approach.

Figure 4.17: The calculated classes for the dimension-wise method. The left side is for the single class density estimation with the guided volume error measure, while the right is for the one-vs-others approach with the misclassification error measure.

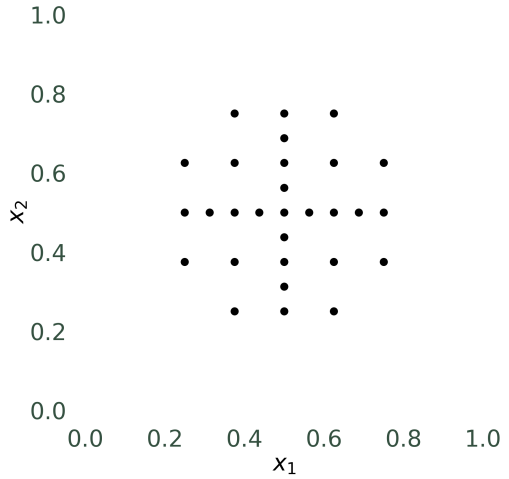


(a) First class. Single class estimation.

(b) First class. One-vs-others approach.



(c) Second class. Single class estimation.



(d) Second class. One-vs-others approach.

Figure 4.18: Final sparse grids of the dimension-wise spatially adaptive solutions. The top grids are for the estimation of the first class, the bottom for the second class. The left side is for the single class density estimation with the guided volume error measure, while the right is for the one-vs-others approach with the misclassification error measure. The right side shows a clear focus on towards the middle, where the boundary between the two classes is located.

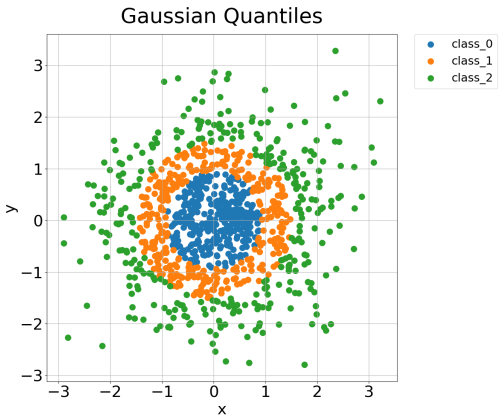
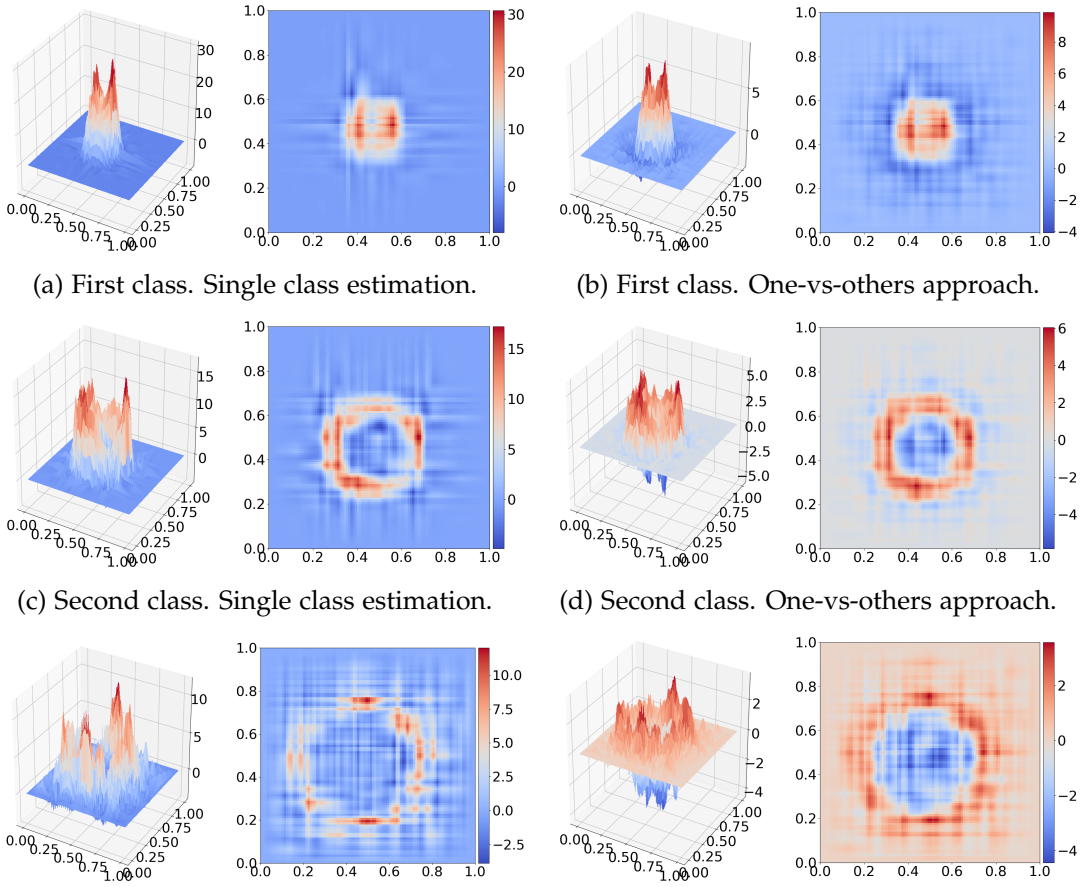


Figure 4.19: Data set for Gaussian quantile classification. The first class is in blue, the second in orange and the third in green.

4.2.2 Gaussian Quantiles

The set of Gaussian quantiles consists of the quantiles for a given d-dimensional Gaussian separated into c-many classes. It consists of a set of (hyper-)circles, with each successive one placed within the previous and a blob in the middle. See Figure 4.19 for a 2D example of the data set. The data set was evaluated in 2 dimensions. The data set was created with a sample size of 1000 and a test set size of 20%, meaning 200 test samples. Note that the test set samples are split evenly across all classes. Figure 4.20 shows the resulting densities for the standard combination technique and the single dimensional refinement, respectively. The classes are ordered from top to bottom, starting with `class_0`, while the first and second configuration are on the left and right, respectively.
Both the standard combination technique and the dimension-wise method achieved above 90% test accuracy for the 2D data set, with the standard combination technique achieving 91% and 92.5% accuracy with the first and second configuration, while the dimension-wise method achieved 91.5% and 93.5% with the first and second configuration, respectively.
The amount of points for the standard combination technique are 723 at level 6, while the dimension-wise method only used an average of 392 points in the first configuration and an average of 348 points in the second configuration. In terms grid point efficiency for achieved accuracy, the dimensional refinement is the clear winner in this case.



(a) First class. Single class estimation.

(b) First class. One-vs-others approach.

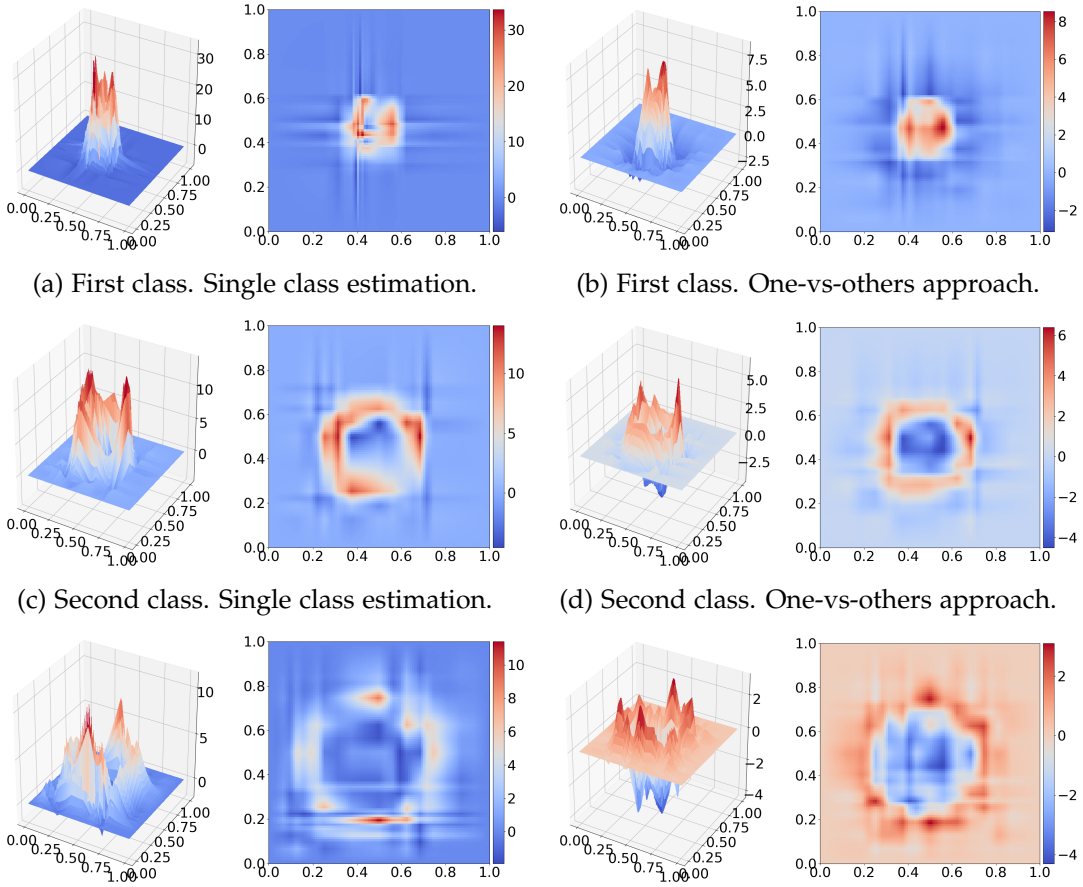
(c) Second class. Single class estimation.

(d) Second class. One-vs-others approach.

(e) Third class. Single class estimation.

(f) Third class. One-vs-others approach.

Figure 4.20: Contour plots of the standard combination solutions for Gaussian quantile classification. The single class estimation is on the left, the one-vs-others approach with misclassification error measure is on the right.



(a) First class. Single class estimation.

(b) First class. One-vs-others approach.

(c) Second class. Single class estimation.

(d) Second class. One-vs-others approach.

(e) Third class. Single class estimation.

(f) Third class. One-vs-others approach.

Figure 4.21: Contour plots of the dimension-wise spatially adaptive solutions for Gaussian quantile classification. The single class estimation is on the left, the one-vs-others approach with misclassification error measure is on the right.

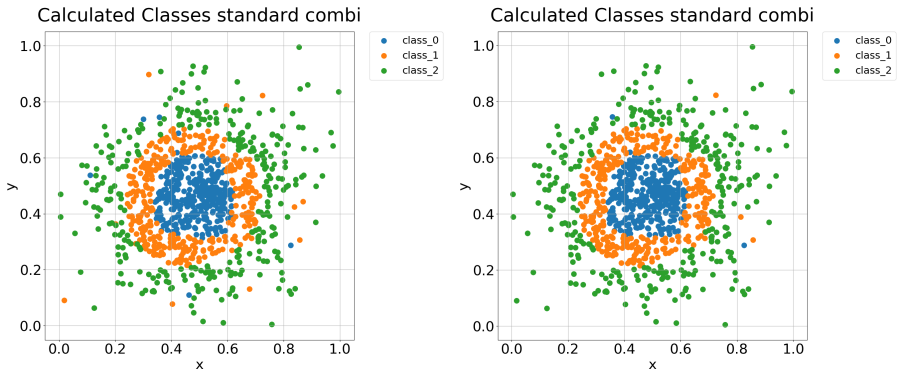


Figure 4.22: Classification results for Gaussian quantile classification for the standard combination technique. The result for the first configuration is on the left, and the right is for the second configuration.

Figure 4.23: Classification results for Gaussian quantile classification for the dimension-wise method. The result for the first configuration is on the left, and the right is for the second configuration.

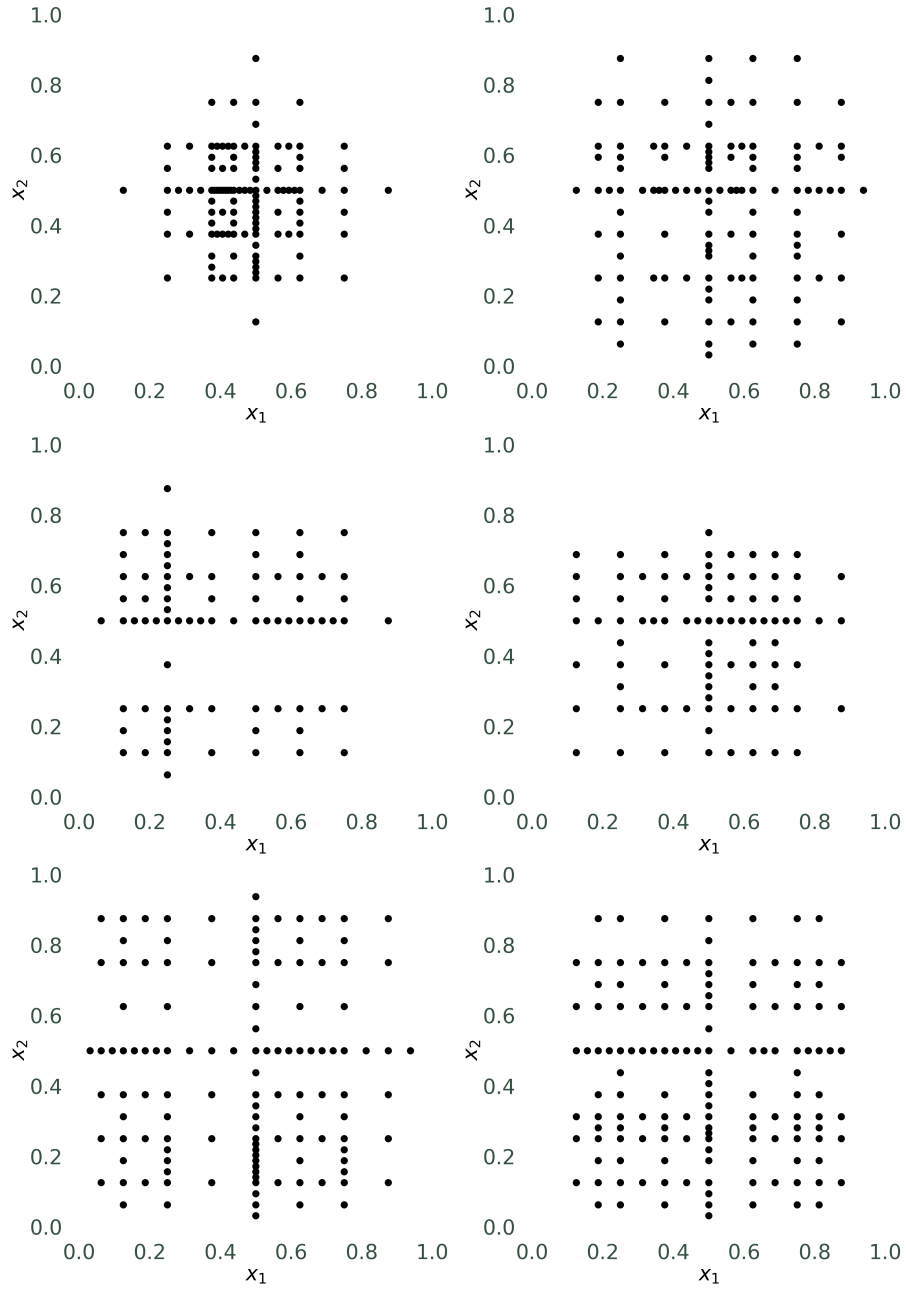


Figure 4.24: Final sparse grids of the dimension-wise spatially adaptive solutions for Gaussian quantile classification.

4.3 Classification on Real Data

4.3.1 Iris flower data set

The iris flower data set was collected by Sir Ronald A. Fisher for his 1936 paper "The use of multiple measurements in taxonomic problems" [3]. The data set consists of 50 samples from each of three species of iris (iris setosa, iris virginica and iris versicolor)[8]. Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. This puts the dimensionality of the data set at 4, with a sample size of 150, divided into 3 classes.
Figure 4.25 shows the resulting accuracy per points used for the first configuration - with and without tree rebalancing on the right and left, respectively - for the dimension-wise method. The results for the second configuration are depicted in figure 4.26. Note that the accuracy for the dimension-wise method is at 100% in figure 4.26, which makes the red line hard to distinguish from the top of the graph. Even at the lowest amount of points used, the accuracy is above 90%. Comparing figures 4.25 and 4.26 reveals that the standard combination technique achieves more consistent results in the first configuration than the second one, where the accuracy "dips" before recovering with the next higher level. The reverse is true for the dimension-wise method.

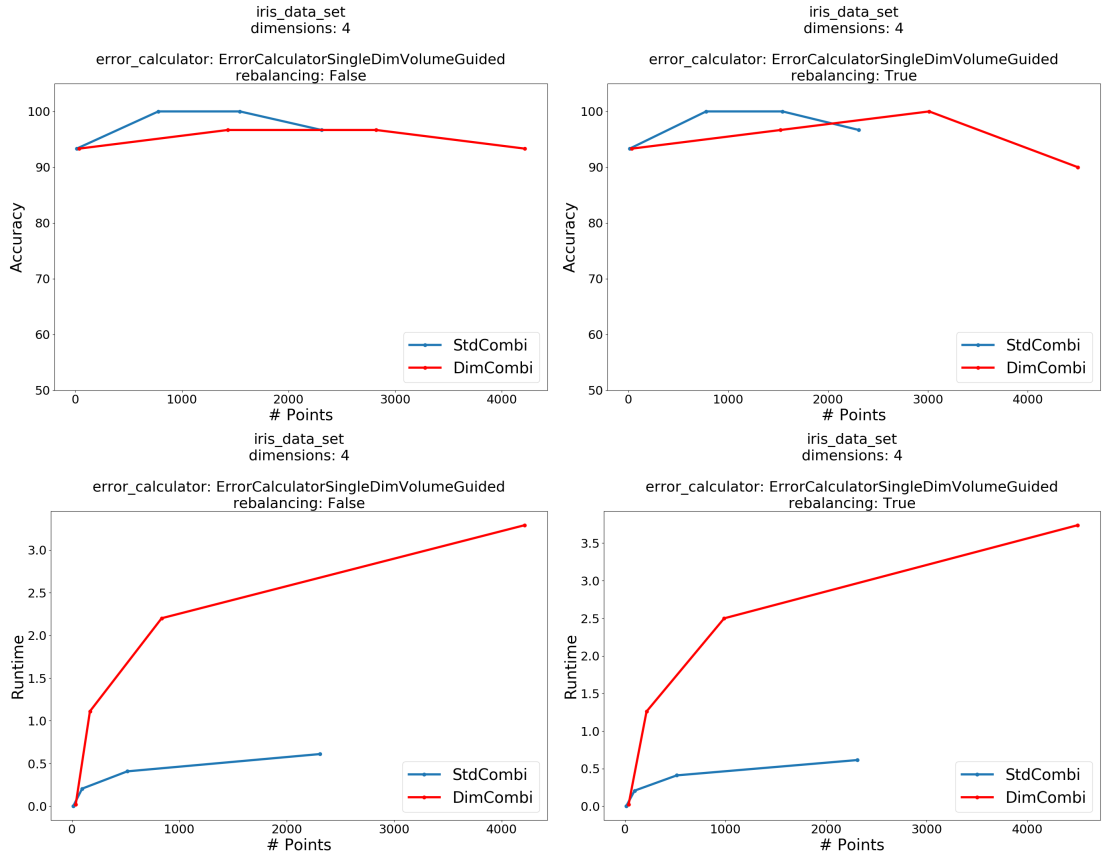


Figure 4.25: Comparison of the results for the iris flower data set with the second configuration. The top two graphs compare points used to accuracy achieved without tree rebalancing on the left and with rebalancing on the right. The bottom two graphs compare the accuracy for the amount of points used also with and without rebalancing on the left and right. For this data set, the rebalancing slightly increases the amount of points used.

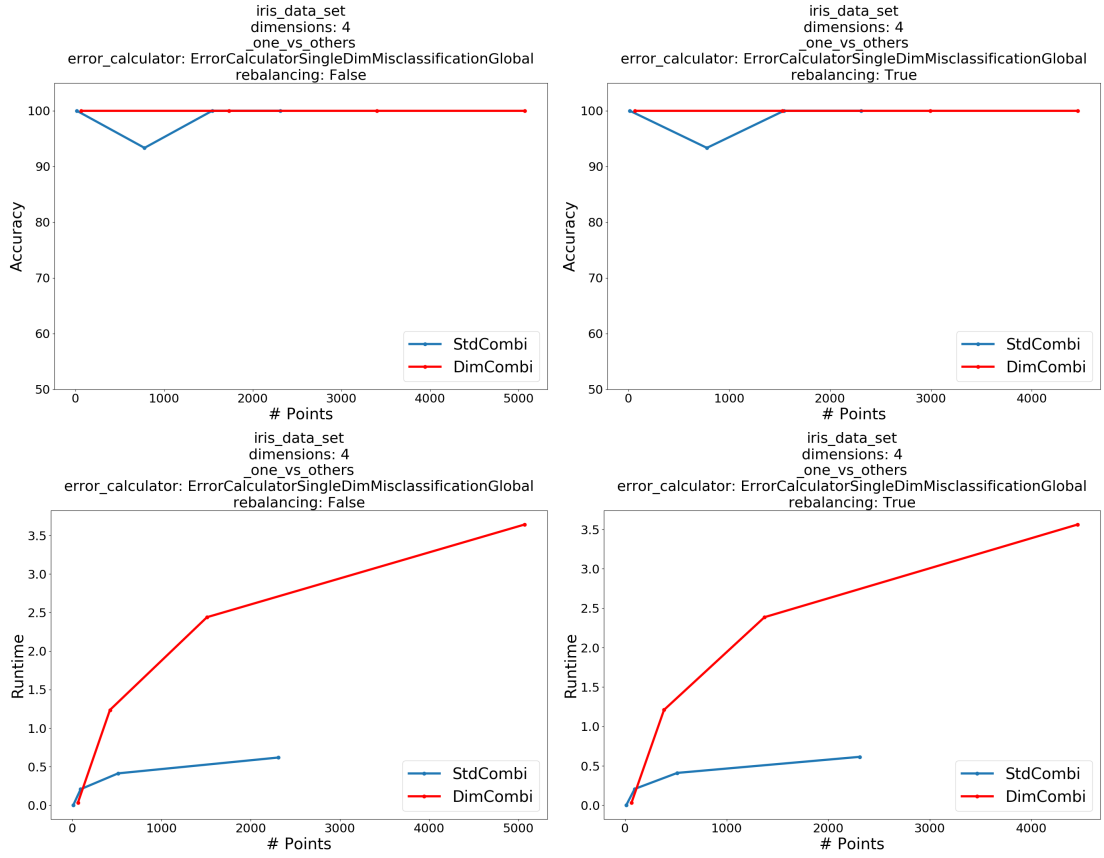


Figure 4.26: Comparison of the results for the iris flower data set with the second configuration. The standard combination technique is in blue, the dimension-wise method in red. The top two graphs compare points used to accuracy achieved without tree rebalancing on the left and with rebalancing on the right. The bottom two graphs compare the runtime for the amount of points used also with and without rebalancing. Despite fewer points being used with rebalancing, the runtime is not affected.

4.3.2 Italian wine data set

These data are the results of a chemical analysis of wine grown in the same region in Italy but derived from three different cultivars[23]. The analysis determined the quantities of 13 constituents found in each of the three types of wine. The 13 features are in order: Alcohol, Malic acid, Ash, Alcalinity of ash, Magnesium, Total phenols, Flavanoids, Nonflavanoid phenols, Proanthocyanins, Color intensity, Hue, OD280/OD315 of diluted wines and Proline. The set is divided into 3 classes, with 59, 71 and 48 samples, respectively. The challenge for the sparse grid algorithms lies mostly in the low number of samples for such a high dimensional data set.
The figures are set up the same as in the iris flower data set. Figure 4.27 is for the first configuration, with and without tree rebalancing, while figure 4.28 is for the second configuration. Again, the plots compare the accuracy per point and the runtime per point used. Test accuracy is good in lower levels and for a low amount of points for both algorithms. Increasing the number of points actually decreases the accuracy for the dimension-wise method, while for the standard combination technique it continues to rise up until the last level. Considering that the number of points used in the dimension-wise method rises much faster than the standard combination technique it indicates that the dimension-wise method quickly overfits the data set. Using tree rebalancing only exacerbates the problem in this case, since it increases the number of points used, as opposed to decreasing them like in the iris data set.
In the second configuration, the amount of points used by the dimension-wise method is far greater than in the first configuration, even though the limit for the number of points was to the same for both, the amount of points the current maximum level of the standard combination technique would use. While the dimensional refinement can exceed this limit to complete its last refinement step it is not allowed to refine past the next higher higher level in each component grid. Thus the likely reason for this excessive use of points is that many more component grids were added in during the refinement steps. Due to the high dimensionality of the data set and the fact that the one-vs-others approach includes the entire data set, refinement in many locations across many component grids can quickly grow the amount of points.

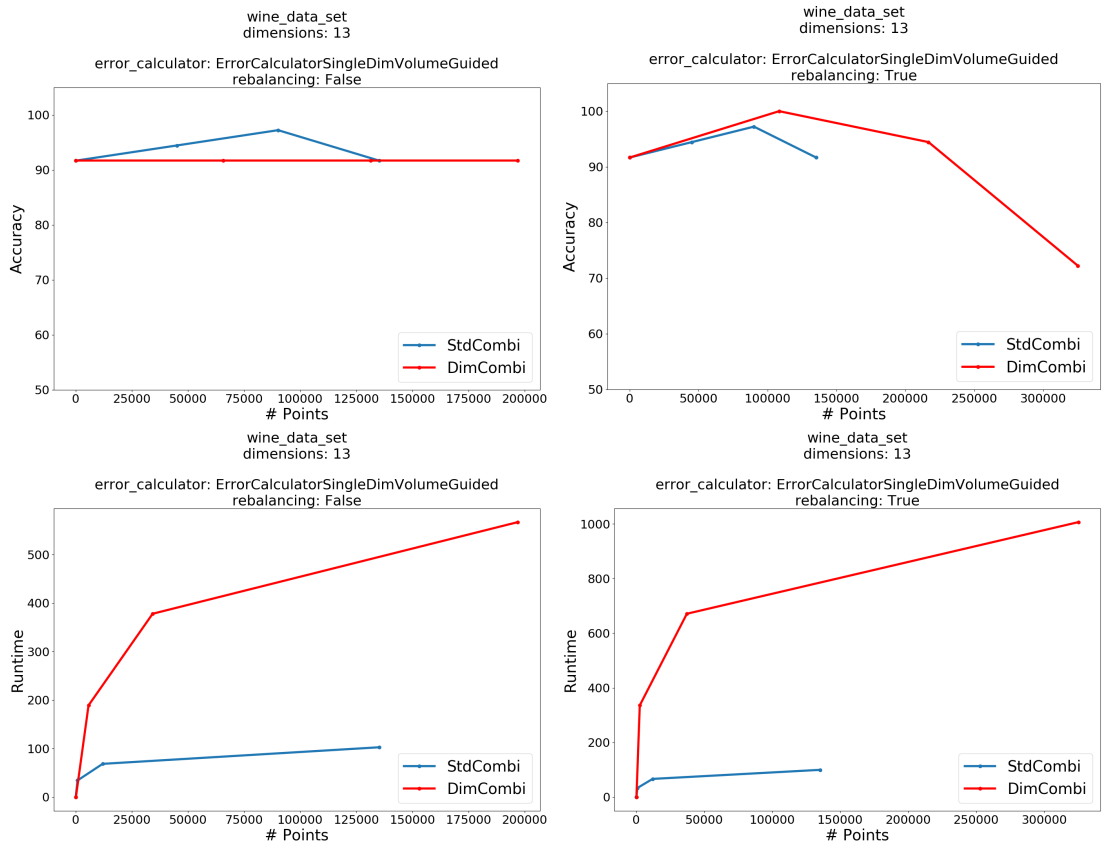


Figure 4.27: Result comparison for the italian wine data set with the first configuration. The standard combination technique is in blue, the dimension-wise method in red. The top two graphs compare points used to the accuracy that was achieved. The bottom two graphs compare the runtime for the amount of points used. Left and right is without and with rebalancing, respectively. For this data set rebalacing greatly influences accuracy and the amount of overfitting for higher point numbers.

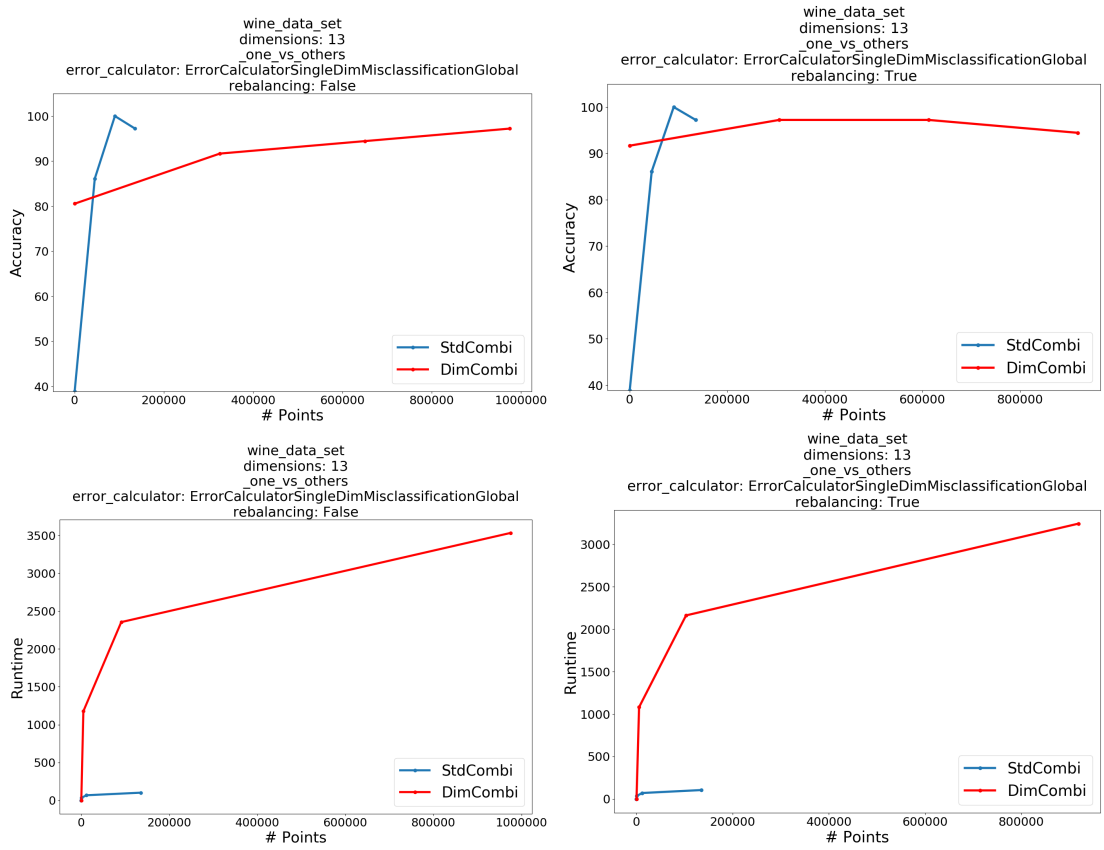


Figure 4.28: Result comparison for the italian wine data set with the second configuration. The top two graphs compare points used to the accuracy that was achieved. The bottom two graphs compare the runtime for the amount of points used also with and without rebalancing on the left and right. Left and right is without and with rebalancing, respectively. Here rebalancing decreases the amount of points used while raising the accuracy.

4.3.3 Breast cancer data set

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass[2]. They describe characteristics of the cell nuclei present in the image. Ten real-valued features are computed for each cell nucleus: radius (mean of distances from center to points on the perimeter), texture (standard deviation of gray-scale values), perimeter, area, smoothness (local variation in radius lengths), compactness ($\frac{\text{perimeter}^2}{\text{area}-1.0}$), concavity (severity of concave portions of the contour), concave points (number of concave portions of the contour), symmetry, fractal dimension ("coastline approximation" - 1).The total amount of attributes is 32. The data set has 569 samples, separated into the two classes benign with 357 samples and malignant with 212.
The standard combination technique achieves good accuracy in the first configuration, but below 50% (roughly 37%), accuracy with the one-vs-others approach. As for the dimension-wise method, the accuracy is similar with around 73% to 82%. When rebalancing is turned off it performs slightly better than when it is is enabled. The results for the second configuration are similarly bad as for the standard combination technique. The accuracy stays around 37%, with only the very last evaluation for the dimension-wise method jumping to around 84%. When calculating the ratios for each class - 37.25% for malignant and 65.9% for benign - the results for the second configuration match closely, indicating that in the failed evaluations all samples are assigned to the first class only.
Like with the wine data set, the amount of points used in the later evaluations of the dimensional refinement grow excessively large.

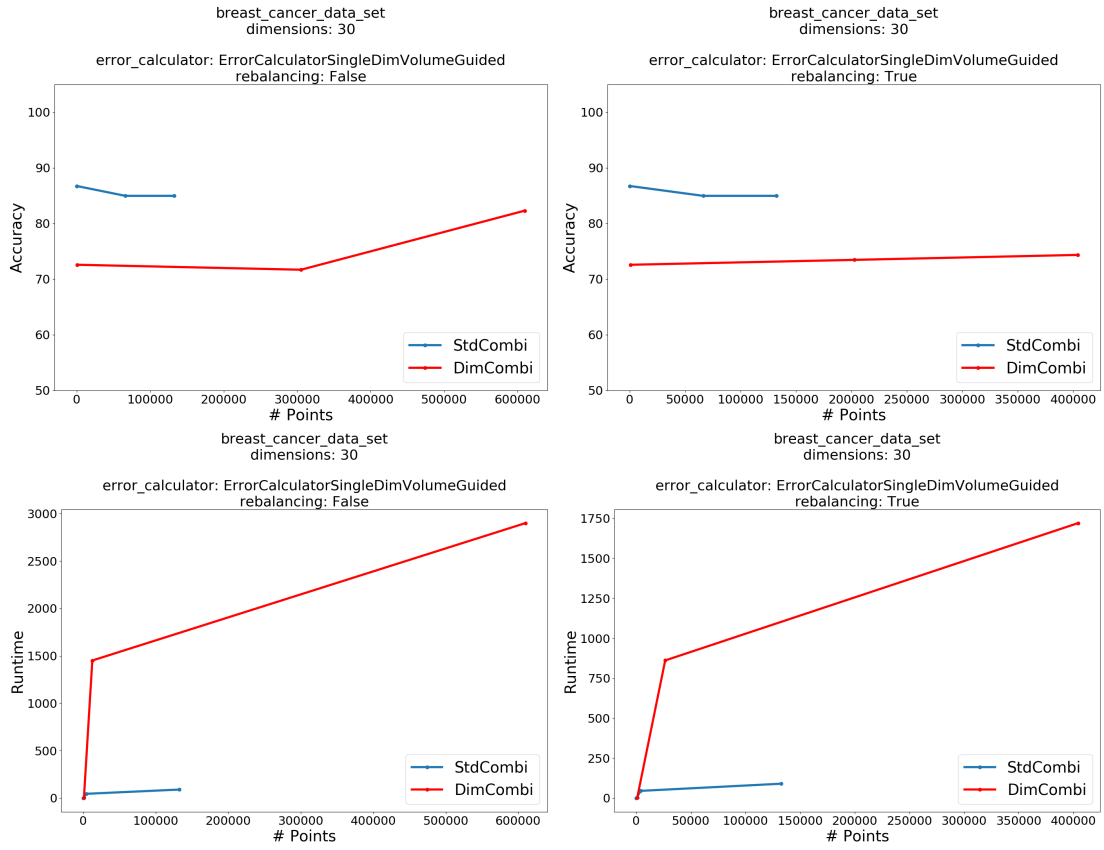


Figure 4.29: Result comparison for the breast cancer data set with the first configuration. The top two graphs compare points used to accuracy achieved. The bottom two graphs compare the runtime for the amount of points used also with and without rebalancing on the left and right. Left and right is without and with rebalancing, respectively. Here rebalancing derceases accuracy significantly in evaluations with higher points limits.

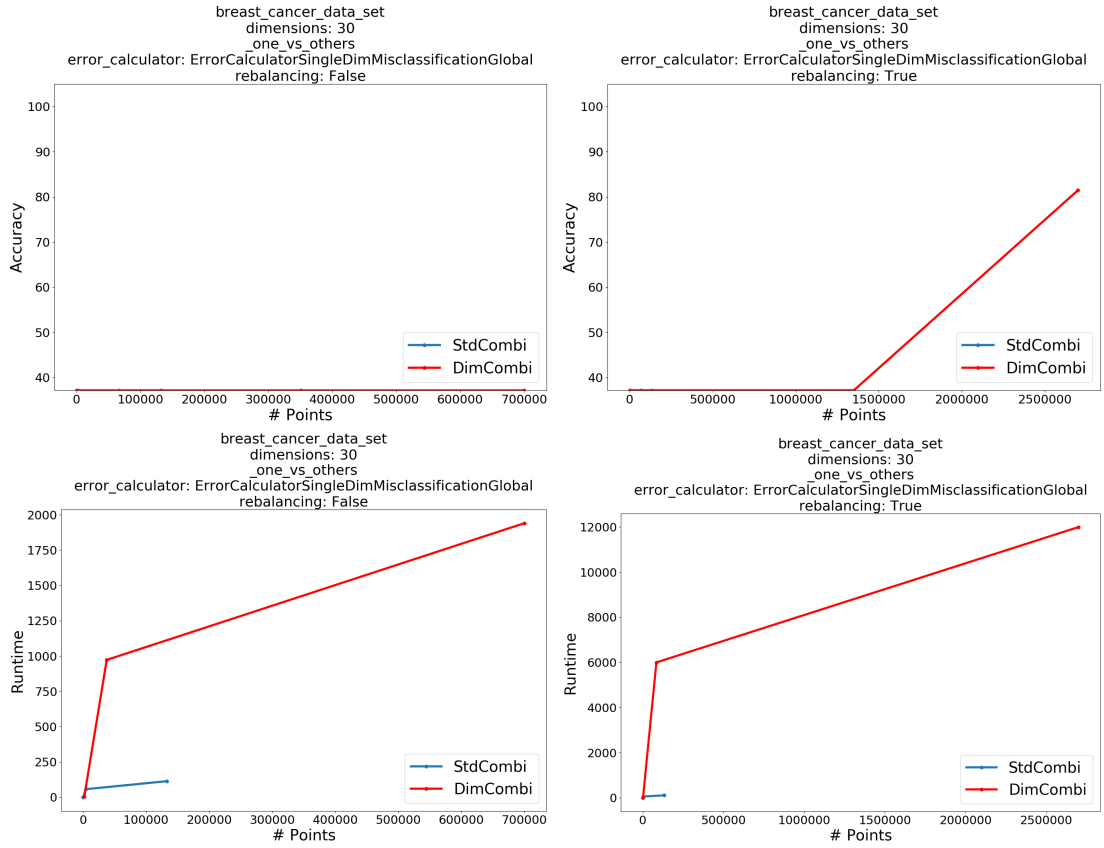


Figure 4.30: Result comparison for the breast cancer data set with the second configuration. The top two graphs compare points used to accuracy achieved. The bottom two graphs compare the runtime for the amount of points used. Left and right is without and with rebalancing, respectively. Since both methods have very low accuracy and because the point count for the dimension-wise method is so much higher, the line for the standard combination technique is hidden in the top two graphs.

5 Conclusion and Outlook

In this thesis density estimation was implemented for the dimension-wise spatially adaptive refinement combination scheme. This method was then evaluated and compared to density estimation with the standard combination technique and the kernel density estimation using a variety of model and real world data. For evaluating the density estimation, the ability of each grid-based approach to match the given data distribution was evaluated by visual similarity and amount of points used. Furthermore, the dimension-wise method was compared to the kernel density estimation with the measures of the average sample distance, the Kullbeck-Leibler divergence and the Pearson correlation. Additionally, classification on model and real world data were conducted to compare the performance of the dimension-wise spatially adaptive refinement to the standard combination approach.
The density estimation with dimension-wise refinement gave good results, but was not as efficient for the simple data sets with few samples when compared to the standard approach. Depending on the shape of the data, the refinement might result in grids that are very similar to the standard combination technique, like for the circle data set. Measuring the dimension-wise method against the kernel density estimation with the average sample distance, the Kullbeck-Leibler divergence and the Pearson correlation also produced favorable results.
The classification results for the two-moons and Gaussian quantile data sets show the strength of the dimensional refinement, being able to achieve similar or superior results with fewer grid points. The classification of the real world data produced mixed results. For the iris data set the one-vs-others approach with the misclassification error measure performed perfectly with the dimension-wise method, while the standard combination technique experienced a small dip in accuracy. When using the single class estimation with the guided volume error, the standard combination approach outperformed the dimensional refinement in terms of accuracy per point used. With the wine data set, the dimension-wise method achieved better accuracy in the first configuration than the standard method, but only with rebalancing enabled. Using the second configuration, the standard combination technique was better, although the dimension-wise method came close in terms of accuracy when using rebalancing, but with a far higher amount of points used.
For the breast cancer data set the dimensional refinement performed poorly compared

to the standard method, except for the highest point limit in the second configuration, where the accuracy inexplicably jumped to over 80% from around 37%.
All in all, the dimension-wise method achieves good results, but should be investigated further to find the reason for the excessive refining for the breast cancer and wine data sets.
While iterative refinement with successive points limits or stopping criteria, like an accuracy threshold, were not used for this thesis, they are already implemented in the framework. Using this can greatly speed up finding a satisfactory solution for a given problem compared to the standard combination technique since it requires recalculation of every component grid with each level. Thus for practical purposes, the dimension-wise spatially adaptive refinement is superior.
Further possible improvements to the framework would be the implementation of the reuse of grid values across the classes. Currently, when using the classification manager, each grid is built and refined independently. Since the calculation of the R-matrix values are independent of the data, they can easily be shared across multiple classes, saving significant amounts of calculations across all component grids for each separate grid. Also, since they all use the same data set in the one-vs-others configuration, with only parts of the data being relabeled for each class, they could share α-vector values across classes.
Another avenue to explore would be the use of pre-processing. For example, since the dimensional refinement works better with axis-aligned data, principal component analysis might improve accuracy.
More generally, since the framework is python based, there are inherent efficiency limitations to the software. Python does not feature robust parallelization capabilities and is restricted to single core execution. Extending the framework with Cython or porting the algorithms to the C++-based SG++ framework would allow larger and more complex data sets to be processed in a reasonable time frame. This might also make it a more competitive alternative to other methods that have enjoyed great popularity and thus sophisticated optimizations, like neural networks for example.

Bibliography

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2009. ISBN: 978-0387310732.

[2] *Breast Cancer Data Set*. 1995. URL: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic).

[3] Sir Ronald A. Fisher. "The use of multiple measurements in taxonomic problems." In: *Annals of Eugenics* (1936), pp. 179–188. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1936.tb02137.x.

[4] J. Garcke. "A dimension adaptive sparse grid combination technique for machine learning." In: (2006).

[5] J. Garcke, M. Griebel, and M. Thess. "Data Mining with Sparse Grids." In: (2001).

[6] M. Griebel. "Sparse Grids and Related Approximation Schemes for Higher Dimensional Problems." In: (2005). URL: http://www.iam.uni-bonn.de/sfb611/.

[7] M. Griebel, M. Schneider, and C. Zenger. "A Combination Technique for the solution of Sparse Grid Problems." In: (1990).

[8] *Iris Data Set*. 1936. URL: https://archive.ics.uci.edu/ml/datasets/Iris/.

[9] A. J. Izenman. "Review Papers: Recent Developments in Nonparametric Density Estimation." In: *Journal of the American Statistical Association* (1991), pp. 205–224. DOI: 10.1080/01621459.1991.10475021.

[10] C. Moser. "Machine Learning with the Sparse Grid Density Estimation using the Combination Technique." Bachelor Thesis. Technische Universität München, 2020.

[11] K. P. Murphy. *Machine Learning - A Probabilistic Perspective*. 2012.

[12] *NumPy*. URL: https://numpy.org/.

[13] M. Obersteiner and H-J. Bungartz. "A generalized spatially adaptive sparse grid combination technique with dimension-wise refinement." Article. Technische Universität München, 2019.

[14] M. Obersteiner and H.-J. Bungartz. "A Spatially Adaptive Sparse Grid Combination Technique for Numerical Quadrature." Article. Technische Universität München, 2019.

[15] B. Peherstorfer, D. Pflüger, and H.-J. Bungartz. "Classification with Probability Density Estimation on Sparse Grids." In: *Sparse Grids and Applications - Munich* (2014), pp. 255–270.

[16] B. Peherstorfer, D. Pflüger, and 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). DOI: 10.1137/1.9781611973440.51.. URL: https://epubs.siam.org/doi/10.1137/1.9781611973440.51.

[17] B. Peherstorfer, D. Pflüger, and H.-J. Bungartz. "Model Order Reduction of Parametrized Systems with Sparse Grid Learning Techniques." Dissertation. Technische Universität München, 2013.

[18] D. Pflüger. "Spatially Adaptive Refinement." In: *Sparse Grids and Applications of Lecture Notes in Computational Science and Engineering*. 2014.

[19] D. Pflüger. "Spatially Adaptive Sparse Grids for High-Dimensional Problems." Phd. Technische Universität München, 2010.

[20] L. Schulte. "Sparse Grid Density Estimation with the Combination Technique." Bachelor Thesis. Technische Universität München, 2020.

[21] S. A. Smolyak. "Quadrature and interpolation formulas for tensor products of certain classes of functions." In: *Dokl. Akad. Nauk SSSR* (1963), pp. 1042–1045.

[22] *sparsSpACE framework*. URL: https://github.com/obersteiner/sparseSpACE.

[23] *Wine Data Set*. 1991. URL: https://archive.ics.uci.edu/ml/datasets/wine.