Density Estimation with Adaptive Sparse Grids for Large Data Sets

Benjamin Peherstorfer* Dirk Pflüger† Hans-Joachim Bungartz‡

Abstract

Nonparametric density estimation is a fundamental problem of statistics and data mining. Even though kernel density estimation is the most widely used method, its performance highly depends on the choice of the kernel bandwidth, and it can become computationally expensive for large data sets. We present an adaptive sparse-grid-based density estimation method which discretizes the estimated density function on basis functions centered at grid points rather than on kernels centered at the data points. Thus, the costs of evaluating the estimated density function are independent from the number of data points. We give details on how to estimate density functions on sparse grids and develop a cross validation technique for the parameter selection. We show numerical results to confirm that our sparse-grid-based method is well-suited for large data sets, and, finally, employ our method for the classification of astronomical objects to demonstrate that it is competitive to current kernel-based density estimation approaches with respect to classification accuracy and runtime.

1 Introduction

Suppose we have a data set $\mathcal{S} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\} \subset \mathbb{R}^d$ of samples drawn from an unknown distribution with unknown probability density function f. The task is to construct an estimated density function $\hat{f}$ of f based on the data $\mathcal{S}$. Estimated density functions can be either used to present, visualize and retrieve information about the data at hand [21], or they can be a means for other common tasks of data mining and statistics such as density-based clustering [3] and Bayesian classification [15]. We refer to [21] for a general study on density estimation and its applications.

In general, we can distinguish between parametric and nonparametric density estimation methods. A parametric density estimation method assumes that the form of the underlying distribution is known and that only a small number of parameters has to be estimated.

Here, we consider nonparametric density estimation which uses only the given data samples to estimate the density and does not require any additional information about the data. There is a variety of nonparametric density estimation methods [9]. Kernel density estimation has become the most widely used method of this kind. The (one-dimensional) estimator $\hat{f}(x) = 1/M \sum_{i=1}^{M} K\left((x - x_i)/h\right)$ is a linear combination of kernel functions K centered at the data points $x_i \in \mathcal{S}$. The performance of the estimator depends on the choice of the kernel function K and the bandwidth $h > 0$. Whereas usually simply the Gaussian kernel $K(x) = (2\pi)^{-1/2}\mathrm{e}^{-x^2/2}$ is used, the selection of the bandwidth h is a far more delicate matter. Approaches to determine a good value for h reach from rules of thumb to highly sophisticated methods requiring a good amount of computational effort, see, e.g., [10, 2] and the references therein. Furthermore, the bandwidth can be selected for each kernel individually, leading to kernel density estimators with adaptive bandwidths [12, 13]. However, this is even more expensive. Besides this issue of selecting the bandwidth, kernel density estimators can become costly to evaluate for large data sets. The evaluation of $\hat{f}$ depends on the number M of (training) data points $\mathcal{S}$. Thus, in order to evaluate the estimated density function $\hat{f}$, all M kernel functions centered at all data points have to be evaluated. One remedy is to divide the data into a small number of bins and place a kernel function at each bin (also called "gridding the data"). However, the number of bins increases exponentially with the dimension of the data points (curse of dimensionality), and this is thus only feasible in up to, say, four dimensions. Note that approaches based on FFT also rely on grids and thus are hardly considered in more than four dimensions [6]. Recently, tree-based approaches have been proposed which introduce a preprocessing step to represent the data in a tree so that they can boost the evaluation or query time [19, 6]. Depending on the data (size, dimension) the pre-processing step can become costly.

We develop a density estimation method based on sparse grids that overcomes these two drawbacks—bandwidth selection, evaluation costs—of kernel density estimation to some extent. Our method is based on the idea in [7] to start with a highly-overfitted guess f_ϵ of

*Aerospace Computational Design Laboratory, AeroAstro, Massachusetts Institute of Technology

†Simulation of Large Systems Group, Institute for Parallel and Distributed Systems, University of Stuttgart, Germany

‡Scientific Computing Group, Department of Computer Science, Technische Universität München, Germany

the density function and then use spline smoothing to obtain a smoother and more generalized approximation $\hat{f}$. The trade-off between fidelity and smoothness can be controlled by a regularization parameter λ. For that, we develop here a simple but efficient cross validation technique to select a good regularization parameter. Furthermore, in contrast to kernel-based methods which represent a function as a linear combination of kernels leading to high evaluation costs, we discretize the estimated density function $\hat{f}$ on basis functions centered at grid points. Since the number of grid points is usually orders of magnitude smaller than the number of data points, and the costs of evaluating the estimator depend only on the number of grid points, the method becomes scalable in the number of data points. However, just as binning for kernel density estimation, a straightforward grid-based discretization suffers from the curse of dimensionality: The number of grid points grows exponentially with the dimension of the data points. That is why we employ sparse grids instead of full (dense) grids [1]. Sparse grids allow a grid-based discretization also in moderately high-dimensional settings. Thus, the result is a sparse-grid-based density estimation method that is scalable in the number of data points and the number of dimensions.

In Sec. 2 we briefly summarize the spline smoothing approach presented in [7] before we discuss sparse grids in Sec. 3 and how to employ them for density estimation in Sec. 4. We show that the estimator is consistent, give details on how to compute the estimator, and present a cross validation technique to select the regularization parameter. With the numerical experiments in Sec. 5 we demonstrate that our method is competitive to kernel density estimators with adaptive bandwidths [13] with respect to accuracy and to tree-based methods [19] with respect to runtime.

2 Density estimation with spline smoothing

We briefly summarize the idea of [7] to employ spline smoothing to derive a smoother and more generalized estimator $\hat{f}$ of a highly-overfitted initial guess f_ϵ.

Suppose we have an initial guess f_ϵ of the density function underlying the data $S = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_M\}$. Following spline smoothing, we are then looking for $\hat{f}$ in a suitable function space V such that

$$\hat{f} = \operatorname*{argmin}_{u \in V} \int_\Omega \left(u(\boldsymbol{x}) - f_\epsilon(\boldsymbol{x})\right)^2 \mathrm{d}\boldsymbol{x} + \lambda \|\mathrm{L}u\|_{L^2}^2. \tag{2.1}$$

Whereas the left term ensures that the function $\hat{f}$ closely fits the initial guess f_ϵ, the right term $\|\mathrm{L}u\|_{L^2}^2$ is a regularization or penalty term imposing a smoothness constraint. The regularization parameter $\lambda > 0$ controls the trade-off between fidelity and smoothness. After some transformations (see [7]) we obtain the variational equation

$$\int_\Omega u(\boldsymbol{x}) s(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} + \lambda \int_\Omega \mathrm{L}u(\boldsymbol{x}) \cdot \mathrm{L}s(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \frac{1}{M} \sum_{i=1}^{M} s(\boldsymbol{x}_i), \tag{2.2}$$

for all test functions $s \in V$ and $f_\epsilon = \frac{1}{M} \sum_{i=1}^{M} \delta_{\boldsymbol{x}_i}$ where $\delta_{\boldsymbol{x}_i}$ is the Dirac delta function centered on $\boldsymbol{x}_i$. Note that $\delta_{\boldsymbol{x}_i}$ is only well-defined if considered within an integral as in (2.2); we refer to [7] for details. Here, we want to find a solution of the variational problem (2.2) with Galerkin projection. Therefore, we have to define a finite-dimensional function space $V_N \subset V$ as the span of the basis functions in $\Phi = \{\phi_1, \dots, \phi_N\}$ centered at grid points. In the following section, we consider sparse grid spaces for this purpose because standard discretizations as employed in [7] become computationally infeasible in high-dimensional settings.

3 Sparse grids

In data mining it is very common to represent a function as a linear combination of kernel functions centered at data points. However, in a naive implementation (cf. Sec. 1), we have to iterate over all data points for each function evaluation. This becomes computationally expensive for large data sets or a large number of function evaluations. In contrast, grid-based approaches represent a function $f_N \in V_N$ as a linear combination with coefficients α_i,

$$f_N(x) = \sum_{i=1}^{N} \alpha_i \phi_i(x)\,, \tag{3.3}$$

where the basis $\Phi = \{\phi_1, \dots, \phi_N\}$ comes from a grid and spans the function space V_N. Hence, the number of basis functions does *not* increase with the number of data points in contrast to classical approaches with kernels. Unfortunately, a straightforward conventional discretization with mesh width $h_\ell = 2^{-\ell}$ in each dimension suffers the curse of dimensionality: The number of grid points is of the order $\mathcal{O}(h_\ell^{-d})$, depending exponentially on the dimension d. In our case, the dimension d equals the dimension of the data points. For functions with bounded mixed, weak derivatives up to order two, i.e., functions in the mixed Sobolev space H^2_{mix}, sparse grids enable one to reduce the number of grid points by orders of magnitude to $\mathcal{O}(h_\ell^{-1} |\log_2(h_\ell)|^{d-1})$ while keeping an interpolation accuracy of $\mathcal{O}(h_\ell^2 |\log_2(h_\ell)|^{d-1})$ in the L^2 norm, instead of $\mathcal{O}(h_\ell^2)$ as in the full grid case, see [1] and Section 4.3.

In the following, we describe the basics of sparse grids as briefly as possible, see [1, 18] for more details. The underlying principle of sparse grids is a one-

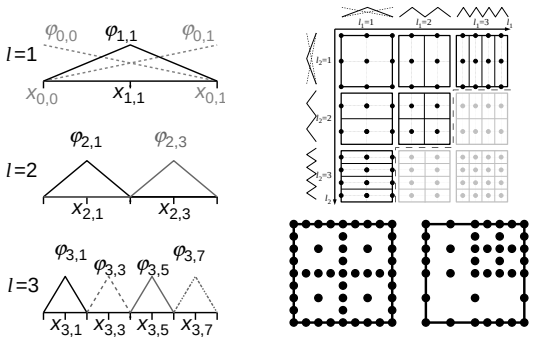


Figure 1: One-dimensional hierarchical basis (left), hierarchical scheme of increments (right top) and a regular and refined sparse grid (right bottom).

dimensional hierarchical system of basis functions (see Fig. 1, left) which is then extended to the d-dimensional case via a tensor product approach. We start with the reference hat function $\phi(x) := \max\{1 - |x|, 0\}$ and obtain the one-dimensional hierarchical hat functions $\phi_{l,i}$ depending on the level l and index i via translation and scaling as $\phi_{l,i}(x) := \phi(2^l x - i)$.

In the d-dimensional case the level $\boldsymbol{l} = (l_1, \ldots, l_d)$ and index $\boldsymbol{i} = (i_1, \ldots, i_d)$ become vectors and the corresponding basis function $\phi_{\boldsymbol{l},\boldsymbol{i}} := \prod_{k=1}^{d} \phi_{l_k,i_k}(\boldsymbol{x})$ is the product of the respective one-dimensional basis functions. This leads to a set of subspaces $W_{\boldsymbol{l}}$ for which the grid points are the Cartesian product of the one-dimensional ones with level l_k in dimension k. Fig. 1 (right top) shows the grids of the two-dimensional hierarchical increments $W_{\boldsymbol{l}}$ up to level 3 in each dimension. Starting from a hierarchical scheme of increments we can select only those subspaces $W_{\boldsymbol{l}}$ that contribute most to the overall solution. The solution to this optimization problem is to cut-off the tableau in Fig. 1 (right top) along the diagonal. The result is the sparse grid space $V_\ell^{(1)}$ of level ℓ [1].

Efficient methods to construct sparse grids are available [1]. Here we do not discuss sparse grid construction further because this is a standard task in the context of sparse grids and several libraries support corresponding methods and provide appropriate data structures, see, e.g., [18, 14, 11].

To further reduce the number of grid points, we can use spatial (local) adaptivity. We start with a rather coarse sparse grid and use a suitable adaptivity criterion to add points in those regions of the domain that are most important, see Fig. 1 (right bottom). A simple (though typically very effective) criterion for adaptive refinement, which we use in the following, is to select the refinement candidates with the highest absolute values of their coefficient $\alpha_{\boldsymbol{l},\boldsymbol{i}}$ weighted with the function value at the corresponding grid point [18].

In the following, we define I as the set of all level-index pairs corresponding to a sparse grid space $V_\ell^{(1)}$ and write a sparse grid function $f_N \in V_\ell^{(1)}$ as the linear combination

$$f_N(\boldsymbol{x}) = \sum_{(\boldsymbol{l},\boldsymbol{i}) \in I} \alpha_{\boldsymbol{l},\boldsymbol{i}} \phi_{\boldsymbol{l},\boldsymbol{i}}(x), \tag{3.4}$$

with $N = |I|$ basis functions $\phi_{\boldsymbol{l},\boldsymbol{i}}$ and coefficients $\alpha_{\boldsymbol{l},\boldsymbol{i}}$ instead of the more common formulation (3.3).

4 Density estimation with sparse grids

In this section, we show how to employ sparse grids to solve the spline smoothing problem (2.2) introduced in Sec. 2. This leads to a sparse-grid-based density estimation method. We discuss the computational procedure as well as asymptotic properties of our method. Finally, we present a cross validation technique to select the regularization parameter λ and show how to marginalize sparse grid density functions.

4.1 Density estimation with adaptive sparse grids We solve the minimization problem (2.2) with Galerkin projection to obtain an estimated density function. Therefore, we define the function space $V_N \subset V$ as the span of the basis functions in $\Phi = \{\phi_1, \ldots, \phi_N\}$ and as usual also set the test space to V_N. The approximation $\hat{f}_N \in V_N$ of the density estimator $\hat{f}$ is a linear combination $\hat{f}_N = \sum_{i=1}^{N} \alpha_i \phi_i$ where the coefficients $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)$ are the solution of a system of linear equations $\boldsymbol{A}\boldsymbol{\alpha} = \boldsymbol{b}$ stemming from (2.2). In particular, the system matrix $\boldsymbol{A}$ and the right hand side $\boldsymbol{b}$ depend on the choice of the discretization, i.e., the basis functions $\phi_1, \ldots, \phi_N$ and thus the space V_N. We employ a sparse grid discretization as discussed in the previous section.

Let Φ be the set of hierarchical basis functions of the (adaptive) sparse grid space $V_\ell^{(1)} \subset H^2_{\text{mix}}$ of level $\ell \in \mathbb{N}$, cf. Sec. 3. Following the usual Galerkin approach, we are then looking for $\hat{f}_N \in V_\ell^{(1)}$ such that

$$\int_\Omega \hat{f}_N(\boldsymbol{x}) \cdot \phi(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} + \lambda \int_\Omega \mathrm{L}\hat{f}_N(\boldsymbol{x}) \cdot \mathrm{L}\phi(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \frac{1}{M} \sum_{i=1}^{M} \phi(\boldsymbol{x}_i), \tag{4.5}$$

holds for all $\phi \in \Phi$. Because the sparse grid function $\hat{f}_N$ is a linear combination (3.3) of the basis functions in Φ with the coefficients $\boldsymbol{\alpha}$, we solve (4.5) by solving the system of linear equations

$$(\boldsymbol{R} + \lambda \boldsymbol{C})\, \boldsymbol{\alpha} = \boldsymbol{b}, \tag{4.6}$$

with $R_{ij} = (\phi_i, \phi_j)_{L^2}$, $C_{ij} = (\mathrm{L}\phi_i, \mathrm{L}\phi_j)_{L^2}$ and $b_i = \frac{1}{M} \sum_{j=1}^{M} \phi_i(\boldsymbol{x}_j)$, where we used an arbitrary ordering of

the N sparse grid basis functions $\phi_{\boldsymbol{l},\boldsymbol{i}}$ and coefficients $\alpha_{\boldsymbol{l},\boldsymbol{i}}$. Thus, to obtain an estimated density function $\hat{f}_N$, we define the sparse grid level $\ell \in \mathbb{N}$—or the number of refinement steps—and solve the system (4.6) to get the coefficient vector $\boldsymbol{\alpha}$ of the linear combination (3.3) corresponding to the sparse grid function $\hat{f}_N$. Note that the level of the sparse grid and the number of refinement steps are parameters of the method which have to be chosen adequately, see the numerical examples in Sec. 5.

4.2 Computational properties It is important that the matrices $\boldsymbol{R}$ and $\boldsymbol{C}$ of the system of linear equations (4.6) are of size $N \times N$ and that the right hand side $\boldsymbol{b}$ is a vector of size N. Thus, in contrast to most kernel-based approaches, the number of unknowns does *not* depend on the number of data points M but only on the number of grid points N. Furthermore, if we want to evaluate the estimated density function $\hat{f}_N$, we only have to iterate over all N basis functions $\phi_{\boldsymbol{l},\boldsymbol{i}} \in \Phi$ and *not* over all M data points in $\mathcal{S}$ as it is the case for (naive) kernel density estimators. Of course, it is crucial to keep the number of grid points at a minimum. With sparse grids we ensure that the number of grid points grows only moderately with the number of dimensions. Employing adaptivity, we can even further reduce the needed grid points, cf. Sec. 3.

We solve the system of linear equations (4.6) with the conjugate gradient method. Hence, we do not have to form the matrices $\boldsymbol{R}$ and $\boldsymbol{C}$ but only need to provide the matrix-vector product. For the matrix $\boldsymbol{R}$, algorithms especially designed to cope with the hierarchical basis and the structure of sparse grids are available, see [18]. There are also highly efficient, parallel versions [8]. The matrix $\boldsymbol{C}$ depends on the operator L which imposes a smoothness constraint on the solution $\hat{f}_N$. A common choice is $\mathrm{L} = \nabla$ leading to $\|\nabla u\|_{L^2}^2$ so that non-smooth functions are penalized. Besides that it has been shown that in the case of the hierarchical basis and sparse grids the term $\sum_{(\boldsymbol{l},\boldsymbol{i}) \in I} \alpha_{\boldsymbol{l},\boldsymbol{i}}^2$ is a good choice for $\|\mathrm{L}u\|_{L^2}^2$ in (2.1) because the coefficients $\boldsymbol{\alpha}$ are a measure for the second derivatives of the function, cf. [18]. The advantage of this choice is that the matrix $\boldsymbol{C}$ becomes the identity matrix and thus the corresponding matrix-vector product is for free.

4.3 Asymptotic properties In the following, we assume that the (exact) probability density function f has bounded mixed, weak derivatives up to order two, i.e., $f \in H_{\mathrm{mix}}^2$, and thus the sparse grid discretization error bounds hold, see Sec. 3. Even though this is not a very restrictive assumption—every twice continuously differentiable function is in H_{mix}^2—we note that this assumption is only important for the following theoretical considerations. Sparse grids, in particular adaptively refined sparse grids, have been shown to work well in more general settings too, see, e.g., [18, 5]. We first show the consistency of our sparse grid estimator $\hat{f}_N$ and then that $\hat{f}_N$ has unit integrand.

THEOREM 4.1. *Let $\mathcal{S} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\} \subset \mathbb{R}^d$ be samples drawn from the distribution with probability density function f, and let $\hat{f}_N \in V_\ell^{(1)}$ be the estimated density function which is the solution of the system of linear equations* (4.6)*. Then, $\hat{f}_N$ is consistent, i.e.,*

$$\Pr\left(\lim_{M\to\infty, N\to\infty} \|\hat{f}_N - f\|_2^2 = 0\right) = 1\,.$$

Proof. We write

$$\|\hat{f}_N - f\|_2^2 = (\hat{f}_N - f, \hat{f}_N - \tilde{f}_N)_2 + (\hat{f}_N - f, \tilde{f}_N - f)_2\,, \tag{4.7}$$

where $\tilde{f}_N \in V_\ell^{(1)}$ is the sparse grid interpolant of the density function f. We consider the second term of (4.7) first, and obtain

$$\begin{aligned} |(\hat{f}_N - f, \tilde{f}_N - f)_2| &\leq \|\hat{f}_N - f\|_2 \cdot \|\tilde{f}_N - f\|_2 \\ &\leq \tfrac{1}{4}\|\hat{f}_N - f\|_2^2 + \|\tilde{f}_N - f\|_2^2 \\ &\leq \tfrac{1}{4}\|\hat{f}_N - f\|_2^2 + \mathcal{O}(h_\ell^2 |\log_2(h_\ell)|^{d-1})\,. \end{aligned}$$

In the second step, we applied Young's inequality. In the third step, we used the L^2 error bound of the sparse grid interpolant of $f \in H_{\mathrm{mix}}^2$, see Sec. 3. Let us now consider the first term of (4.7). We are interested in $M \to \infty$ and thus overfitting does not occur; hence, we can assume $\lambda = 0$. We know that $\hat{f}_N - \tilde{f}_N \in V_\ell^{(1)}$ because both functions are sparse grid functions. Therefore, we denote $s = \hat{f}_N - \tilde{f}_N$ and obtain with (2.2),

$$\begin{aligned} (\hat{f}_N - f, \hat{f}_N - \tilde{f}_N)_2 &= (\hat{f}_N - f, s)_2 \\ &= (\hat{f}_N, s)_2 - (f, s)_2 \\ &= \tfrac{1}{M}\textstyle\sum_{i=1}^M s(\boldsymbol{x}_i) - \mathbb{E}_f\left(s(\boldsymbol{x})\right)\,, \end{aligned}$$

because due to (2.2) and $\lambda = 0$, we have $(\hat{f}_N, s)_2 = \frac{1}{M}\sum_{i=1}^M s(\boldsymbol{x}_i)$ for all $s \in V_\ell^{(1)}$. We denote with $\mathbb{E}_f(X)$ the expected value of X with respect to the density function f. Altogether we obtain

$$\begin{aligned} \|\hat{f}_N - f\|_2^2 \leq\ & \tfrac{4}{3} \cdot \mathcal{O}(h_\ell^2 |\log_2(h_\ell)|^{d-1}) + \\ & \tfrac{4}{3} \cdot \tfrac{1}{M}\textstyle\sum_{i=1}^M s(\boldsymbol{x}_i) - \tfrac{4}{3} \cdot \mathbb{E}_f\left(s(\boldsymbol{x})\right)\,. \end{aligned}$$

For $M \to \infty$, and samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$ drawn from the distribution corresponding to f, we have $\Pr\left(\frac{1}{M}\sum_{i=1}^M s(\boldsymbol{x}_i) = \mathbb{E}_f\left(s(\boldsymbol{x})\right)\right) = 1$, due to the strong law of large numbers. Thus, only the sparse grid discretization error remains. However, the discretization error converges to zero for $N \to \infty$ (i.e., $\ell \to \infty$) because $f \in H_{\mathrm{mix}}^2$. Hence, $\Pr(\|\hat{f}_N - f\|_2^2 = 0) = 1$.

THEOREM 4.2. *The estimator* $\hat{f}_N \in V_\ell^{(1)}$ *has unit integrand if the regularization term* $\|\nabla \hat{f}_N\|_{L^2}^2$ *is used in* (2.2).

Proof. Let $\mathbf{1}_N \in V_\ell^{(1)}$ be the constant map[1] with $\boldsymbol{x} \mapsto 1$. We then have

$$\int \hat{f}_N(\boldsymbol{x})\mathrm{d}\boldsymbol{x} + \lambda \int \nabla \hat{f}_N(\boldsymbol{x}) \cdot \nabla \mathbf{1}_N(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \frac{1}{M}\sum_{i=1}^{M} \mathbf{1}_N(\boldsymbol{x}),$$

because (2.2) holds for all $s \in V_\ell^{(1)}$. It follows immediately that $\int \hat{f}_N(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = 1$ because $\int \nabla \hat{f}_N(\boldsymbol{x}) \cdot \nabla \mathbf{1}_N(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = 0$ and $\frac{1}{M}\sum_{i=1}^{M} \mathbf{1}_N(\boldsymbol{x}) = 1$.

We note that a similar consistency result as in Theorem 4.1 is shown in [20] for a sparse grid extrapolation technique. We also note that the proof of Theorem 4.2 can be easily extended to show that also the first moment is matched for $N \to \infty$. Finally, we note that we cannot guarantee non-negativity of the sparse grid estimator $\hat{f}_N$ because of the discretization error incurred by the sparse grids; however, negative values of $\hat{f}_N$ are limited for large ℓ due to the error bounds given in Sec. 3. Furthermore, most applications of density estimation (e.g., visualization, clustering, classification) can easily cope with values that are, say, -10^{-10} instead of 0. This is confirmed by our numerical results in Sec. 5.

4.4 Selection of regularization parameter The minimization problem (4.5) depends on the regularization parameter λ which controls the trade-off between fidelity and smoothness. It ensures that the estimated density function generalizes to new data, i.e., it prevents overfitting. In this section, we introduce a cross validation technique that allows us to select a parameter λ.

After we have obtained our estimated density function with the training data $\mathcal{S} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$, i.e., we computed the coefficients $\boldsymbol{\alpha}$, we can test the estimator by computing the residual of the system of linear equations (4.6) but now without $\boldsymbol{C}$ and with a new right hand side $\boldsymbol{b}_T$ which has been built using test data. The L^2 norm of the residual $\|\boldsymbol{R}\boldsymbol{\alpha} - \boldsymbol{b}_T\|_2$ is then the accuracy indicator. This is a reasonable indicator: If the estimated density function captures the underlying density function and distribution we should obtain a small value for the test data as well. If our estimated density does not generalize to the test data then the value of the residual of the system of linear equations remains high. We compare the norms of the residuals for different choices of the parameter λ and select the one for which the residual is smallest, i.e., we select the λ for which the corresponding estimated density function generalizes best to the test data.

To ensure that each sample is used for training and testing, we split the training data $\mathcal{S} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$ into k equally sized sets. Each of these k sets is then used for testing and the union of the remaining ones for training. This is standard k-fold cross validation widely used in classification and other data mining tasks.

4.5 Marginalized density function Assume we have estimated the joint probability function $\hat{f}_{X_1,X_2}$ of two random variables X_1 and X_2. The marginal probability density function of X_1 is $\hat{f}_{X_1}(x_1) = \int_{x_2} \hat{f}_{X_1,X_2}(x_1, x_2)\mathrm{d}x_2$. With

$$\hat{f}_{X_1,X_2}(x_1, x_2) = \sum_{(\boldsymbol{l},\boldsymbol{i}) \in I} \alpha_{\boldsymbol{l},\boldsymbol{i}} \phi_{l_1,i_1}(x_1)\phi_{l_2,i_2}(x_2),$$

where I is the set of all level-index pairs of a two-dimensional sparse grid, we have the marginal density function of X_1,

$$\hat{f}_{X_1}(x_1) = \sum_{(\boldsymbol{l},\boldsymbol{i}) \in I} 2^{-l_2} \alpha_{\boldsymbol{l},\boldsymbol{i}} \phi_{l_1,i_1}(x_1),$$

as $\int_x \phi_{l,i}(x)\mathrm{d}x = 2^{-l}$. Because a d-dimensional sparse grid consists of several $d-1$ dimensional sparse grids [1], the marginal density function $\hat{f}_{X_1}$ can be represented without an additional approximation error on a one-dimensional sparse grid. This leads to a dramatic saving in the number of grid points. Due to the tensor product approach of sparse grids this procedure carries over to the d-dimensional case. Note that marginalizing a sparse grid density function does not depend on the number of data points and is only linear in the number of grid points.

5 Numerical Experiments

In this section, we compare our sparse-grid-based density estimation (SGDE[2]) method with two variants of kernel density estimation: a kernel method with adaptive bandwidths (libagf[3]) [13] and a tree-based kernel method (density trees[4]) [19]. Note that adaptive bandwidths estimators are known to achieve high accuracies but are costly to evaluate, and that tree-based methods are well-suited for large data sets but require an expensive pre-processing step, see Sec. 1 and the extensive discussions in [19, 6]. Note further that we are only interested in nonparametric density estimation here and thus do not consider, e.g., Gaussian mixture models.

[1] Sparse grids with boundary points are required for $\mathbf{1}_N$ to be in the sparse grid space $V_\ell^{(1)}$.

[2] SG++ library, http://www5.in.tum.de/SGpp

[3] Version 0.9.6, http://libagf.sourceforge.net/

[4] MLPACK 1.0.6 (with Intel MKL), http://www.mlpack.org/

In the following, we employ the identity regularization term (cf. Sec. 4.2) and cross validation with 10 folds to obtain the regularization parameter λ for the sparse grid density estimator, if not otherwise noted. We also use 10-fold cross validation for the density trees method as suggested in [19].

5.1 Synthetic data sets Let us first consider synthetic data sets where we know the target density function f and thus can compute the averaged L^2 error over a test set $\mathcal{T}$ with $M_{\mathcal{T}}$ test samples as

$$\frac{1}{M_{\mathcal{T}}} \sqrt{\sum_{\boldsymbol{x} \in \mathcal{T}} \left(f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x}) \right)^2}, \tag{5.8}$$

where $\hat{f}$ is the estimated density function. The synthetic data sets are sampled in each direction independently from a normal distribution with mean 0.4 and standard deviation 0.1 and a beta distribution with $\alpha = 3$ and $\beta = 5$. Thus, a data set denoted by "GGBBB" is a five dimensional data set ($d = 5$) where the samples in the first two dimensions are drawn from the normal distribution and in the third, fourth and fifth from the beta distribution.

Before we start the comparison with kernel density estimation, we show the convergence of our SGDE method (nonadaptive sparse grid of level six) for the two-dimensional "BB" (beta distribution) data set with up to 10 million samples in Figure 2a. The error (5.8) was computed over the samples corresponding to an equidistant grid with 2^8 points in each dimension. We observe a convergence rate of about $\mathcal{O}(M^{-1/2})$. Kernel density estimation can achieve a rate of up to $\mathcal{O}(M^{-3/4})$ for the mean integrated square error if the optimal bandwidth is selected; however, the optimal bandwidth is rarely known, and so rates of $\mathcal{O}(M^{-1/2})$ are common for kernel estimators as well [21].

In Figures 2b and 2c we compare the averaged L^2 error (5.8) of the estimators obtained from 100,000 training samples with SGDE, libagf, and density trees over test sets that contain the first 50,000 points of the low-discrepancy Sobol sequence (generated with GNU Scientific Library). For SGDE, we employ an adaptive sparse grid where we start with a grid of level three and refine five times up to 100 grid points. This leads to ≈ 4000 points in five dimensions and to ≈ 5000 points in six dimensions, cf. Table 1. The regularization parameter λ is fixed to 10^{-5} to emphasize that our SGDE method is not very sensitive with respect to λ. The Figures 2b and 2c clearly show that our method is competitive with both kernel density estimation methods even though we keep the regularization parameter λ fixed over all data sets. For five out of seven data sets, our SGDE estimator achieves the highest accuracy.

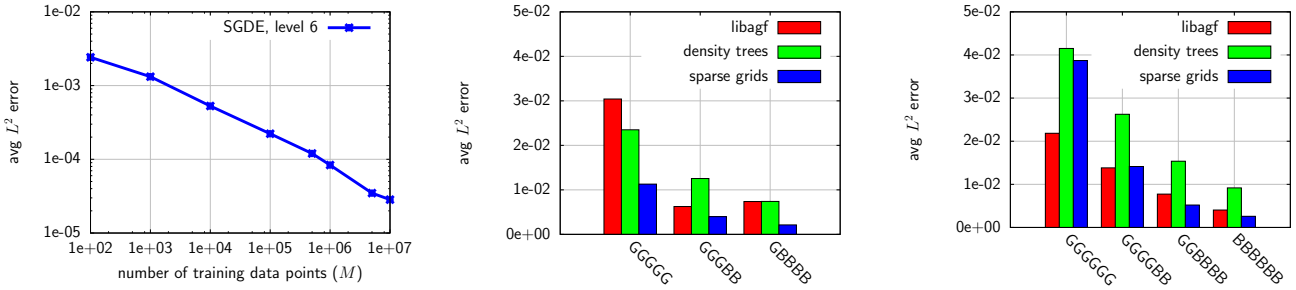
5.2 Runtimes In Table 1 we summarize the runtimes of the libagf, density trees, and our SGDE method for five to ten dimensional synthetic data sets. The cross validation runtime is not included[5]. The estimators are evaluated at 100,000 samples. All measurements were performed on a system with an Intel SandyBridge-EP Xeon E5-2670. No parallelized implementations of the libagf and density trees methods are available. We split the total runtime into the time spent for initializing the estimator (e.g., constructing the trees, solving the system of linear equations) and the time needed to evaluate the estimator $\hat{f}_N$. We employ the same adaptive sparse grid settings as in the previous Sec. 5.1. Again, the regularization parameter λ is fixed to 10^{-5}.

The libagf has no initialization phase except for loading the data; but the evaluation time increases linear with the number of sample points (M). The density trees method has a costly initialization phase (constructing the trees) that strongly grows with the number of training data points, but the evaluation of the estimated density is very fast. Our sparse grid density estimation method also has an initialization phase, where the system of linear equations (4.6) is solved, but the required time is low compared to density trees. The initialization phase of SGDE increases slightly with the number of training data points M because the computation of the right hand side of (4.6) depends on the number of data points. Whereas the sample size is increased by a factor of 50, the runtime to solve the system increases only by a factor of five. The evaluation time of our sparse-grid-based estimator stays constant even if we increase the number of training samples. That is because the evaluation costs only depend on the number of sparse grid points which does not change for more training points.

Let us now consider the runtimes with respect to the dimensions. The runtimes of libagf and density trees are only slightly affected by the dimension of the data set. However, since the number of sparse grid points grows with the number of dimensions (see Sec. 3), we observe a runtime increase of our SGDE method. This emphasizes that our method is well-suited for moderately high-dimensional data sets. Nevertheless, our SGDE method only needs 1,361 seconds on one core to estimate a ten-dimensional density function from 500,000 training samples and to evaluate it on 100,000 test samples. The parallel version on four cores takes only 417 seconds.

5.3 Real-world data sets So far we have evaluated our SGDE method only on synthetic data sets (normal

[5] For the density trees implementation (MLPACK), we used 1-fold cross validation and divided the training runtime by two.



(a) convergence rate (b) five dimensions (c) six dimensions

Figure 2: In (a) the convergence of our SGDE method for a two-dimensional beta distribution. In (b) and (c) the averaged L^2 error of the libagf, density trees, and SGDE estimators for synthetic data sets.

Table 1: Runtime (in seconds, without cross validation) of initializing and evaluating estimated densities with libagf, density trees, and our SGDE method: The time spent for initializing our SGDE estimator depends only slightly on the number of training data points; the evaluation time depends only on the number of sparse grid points and is independent from the number of training data points.

			libagf		d. trees			SGDE, 1 core		SGDE, 4 cores	
data set	d	size M	init[s]	eval[s]	init[s]	eval[s]	#gp	init[s]	eval[s]	init[s]	eval[s]
GGGBB	5	10000	<1	146	**1**	**<1**	4277	16	5	6	2
GGGBB	5	100000	<1	2145	78	<1	4185	**27**	**6**	9	2
GGGBB	5	500000	<1	11488	2567	<1	4285	**87**	**6**	24	2
GGBBBB	6	10000	<1	145	**1**	**<1**	5586	34	10	13	2
GGBBBB	6	100000	<1	1898	75	<1	5183	**52**	**10**	17	3
GGBBBB	6	500000	1	12931	2539	<1	5198	**152**	**9**	44	2
GGGBBBB	7	500000	<1	13251	2593	<1	6823	**293**	**14**	86	3
GGGGBBBB	8	500000	<1	13755	2327	<1	9391	**546**	**22**	159	7
GGGGGBBBB	9	500000	<1	13157	2106	<1	12301	**857**	**30**	256	8
GGGGBBBBBB	10	500000	<1	14123	2095	<1	14764	**1314**	**47**	403	14

and beta distributions). Now we consider seven multi-modal real-world data sets from various application areas which are frequently used to benchmark data mining methods. Description and links to download these data sets can be found in [17], for DR6 see Sec. 5.5. Because we have real-world data sets we do not know the target density function f anymore. Following [22], we split the data sets into a training set $\mathcal{S}$ and test set $\mathcal{T}$, estimate the density function on the training data, and compute the log-likelihood of the test data,

$$\frac{1}{M_{\mathcal{T}}} \sum_{\boldsymbol{x} \in \mathcal{T}} \log\left(\hat{f}(\boldsymbol{x})\right).$$

The log-likelihood of the test samples is then a measure of how well our estimated density function generalizes to before unseen data. Again, we employ adaptive sparse grids with the same setting as in Sec. 5.1. In Table 2, we compare our SGDE method (10-fold CV) with libagf and density trees (10-fold CV). For two out of these seven data sets, our SGDE achieves better results than the adaptive bandwidths estimator libagf, and for five data sets a better result than density trees.

Table 2: Log-likelihood (higher is better) for multi-modal real-world data sets: Our SGDE performs better than libagf for two data sets, and better than density trees for five sets.

data set	d	libagf	d. trees	SGDE
checker	2	1.82	**1.83**	1.82
spheres	3	**1.34**	0.44	1.29
svmguide	4	4.08	2.89	**4.22**
DR6	4	**12.08**	9.19	8.37
bupa liver	6	**4.92**	3.69	3.97
olives	8	4.22	3.35	**7.30**
oilflow	12	**9.66**	7.31	7.52

Figure 3 shows the estimated densities for the old faithful data set. Our sparse grid method captures the two dominating clusters of the data sets and also the

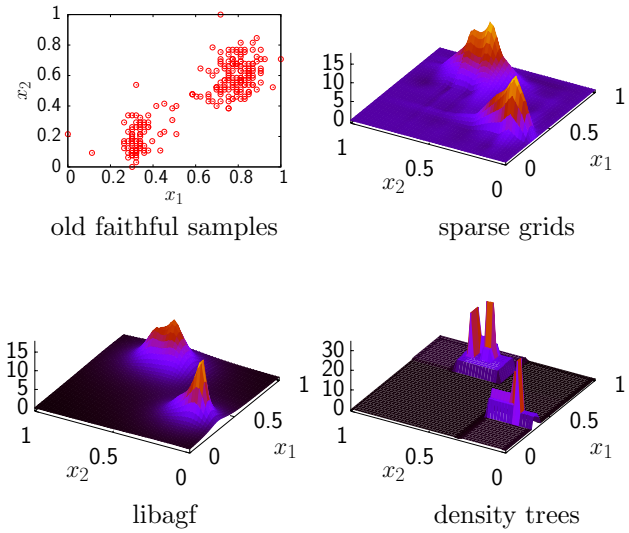


Figure 3: Estimated density functions for the old faithful data set. Our sparse grid method captures not only the two dominating clusters but also the two modes within the cluster centered at $(0.8, 0.6)$.

two modes within the cluster centered at $(0.8, 0.6)$.

5.4 Bayesian classification Density estimation is a common task in Bayesian classification methods where the class $y \in \{1, \ldots, k\}$ of a data point $\boldsymbol{x} \in \mathbb{R}^d$ is determined by evaluating

$$f(Y = y|\boldsymbol{x}) = \frac{f(\boldsymbol{x}|Y = y) \cdot p(Y = k)}{f(\boldsymbol{x})} \tag{5.9}$$

for all $y \in \{1, \ldots, k\}$ and with prior p. The class with the highest probability is assigned to the point $\boldsymbol{x}$. To compute (5.9), the class-conditional density $f(\boldsymbol{x}|Y = y)$ has to be estimated from the data samples [15, 4].

In Table 3 we show the classification accuracy of Bayesian classifiers where the class-conditional densities were estimated with libagf, density trees, and our SGDE method and the prior $p(Y = k)$ was set to the ratio of the data points with class k in the data sets. The sparse-grid-based classifiers achieve competitive results with respect to accuracy. It is also the fastest classifier for the largest data set (checker). For small data sets, the overhead of building the grid and solving the system is not compensated. Note that it is also possible to precompute the matrices $\boldsymbol{R}$ and $\boldsymbol{C}$ of (4.6) once, because they do *not* depend on the data points, and use them over and over again for different data sets [17]. Such an offline/online decomposition can boost the runtime of SGDE by several orders of magnitude, as the "time*" results in Table 3 confirm.

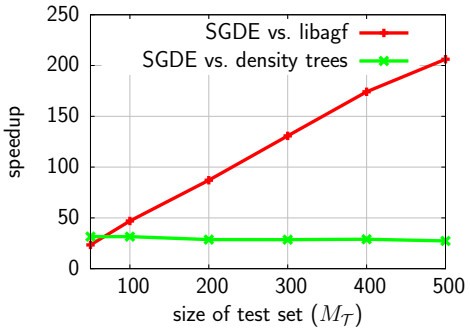


Figure 4: Speedup of the sparse-grid-based Bayesian classifier for the checkerboard data set. The timings include cross validation, training (50,000 points), and prediction time. We observe speedups of up 30 over density trees, and up to 200 over libagf.

Let us consider the runtimes for the checkerboard data set in detail. We vary the number of test data points to show their effect on the runtime of the classification methods. If we consider the total runtime (initialization and prediction time), our SGDE method is between 20 and 200 times faster than libagf, and about 30 times faster than density trees. Note that the accuracies corresponding to the checkerboard data sets are roughly equal for all three density estimation methods, cf. Table 3.

5.5 Photometric classification of stars/quasars Finally, let us consider the DR6 data set from the Sloan Digital Sky Survey (SDSS) repository which is available from [16]. It contains four photometric properties of astronomical objects that allow us to classify them either as stars or quasars. In particular, we learn from 505,290 data points and then classify another 200,000 objects. The results of the Bayesian classifiers with libagf, density trees, and our SGDE method are shown in Table 3. The sparse-grid-based classifier achieves a better accuracy result than density trees, but is slightly worse than libagf; however, our classifier is learned (including 10-fold CV) and evaluated about nine and 60 times faster than the libagf- and density-trees-based classifier, respectively.

6 Conclusions

We presented a sparse-grid-based density estimation method that is well-suited for large data sets because the density function is discretized on basis functions centered at grid points rather than on kernels centered at data points. We showed the consistency of the sparse grid estimators and discussed the computational procedure. Our numerical examples demonstrate that the

Table 3: Accuracy and runtime of Bayesian classifier with densities estimated by libagf, density trees, and SGDE. For "time*", data-independent matrices were pre-computed. Time measurements include cross validation. SGDE achieves speedups for large data sets of up to 100 and 80 compared to libagf and density trees, respectively.

				libagf		d. trees		SGDE			
data set	#train	#test	d	test[%]	time[s]	test[%]	time[s]	test[%]	time[s]	time*[s]	#gp
bupa liver	290	55	6	67.27	<1	61.82	<1	**70.91**	1189	32	10625
olives	348	88	8	**97.73**	<1	92.05	<1	**97.73**	3692	20	6401
spheres	607	68	3	**100.00**	<1	**100.00**	<1	98.53	5	1	351
oilflow	1318	687	12	**97.09**	1	86.03	1	85.74	13245	19	3249
svmguide	3089	4000	4	94.35	2	82.33	1	**94.50**	20	2	769
checker	100000	50000	2	99.84	2393	**99.95**	1893	99.47	23	20	769
DR6	505290	200000	4	**94.83**	14227	91.83	90619	93.36	1456	1025	7937

sparse grid method is competitive to modern kernel density estimation methods (adaptive bandwidths methods and tree-based methods) with respect to accuracy, and achieves speedups of up to several hundreds for large and moderately high-dimensional data sets.

Acknowledgments

This work was partially funded by the German Federal Ministry of Education and Research under grant 05M10WOA.

References

[1] H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.

[2] T. Duong. ks: Kernel density estimation and kernel discriminant analysis for multivariate data in R. *Journal of Statistical Software*, 21(7):1–16, 10 2007.

[3] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.

[4] J. Garcke. *Maschinelles Lernen durch Funktionsrekonstruktion mit verallgemeinerten dünnen Gittern*. PhD thesis, Universität Bonn, 2004.

[5] J. Garcke. Regression with the optimised combination technique. In *Proceedings of the 23rd ICML '06*, pages 321–328. ACM Press, 2006.

[6] A. G. Gray and A. W. Moore. Nonparametric density estimation: Toward computational tractability. In *SDM*, 2003.

[7] M. Hegland, G. Hooker, and S. Roberts. Finite element thin plate splines in density estimation. *ANZIAM Journal*, 42, 2000.

[8] A. Heinecke, S. Schraufstetter, and H.-J. Bungartz. A highly-parallel black-scholes solver based on adaptive sparse grids. *Int. J. Comp. Math.*, June 2012.

[9] A. Izenman. Recent developments in nonparametric density estimation. *J. Amer. Stat. Assoc.*, 86(413), 1991.

[10] M. C. Jones, J. S. Marron, and S. J. Sheather. A brief survey of bandwidth selection for density estimation. *J. of Amer. Stat. Assoc.*, 91(433), 1996.

[11] A. Klimke and B. Wohlmuth. Algorithm 847: spinterp: Piecewise multilinear hierarchical sparse grid interpolation in MATLAB. *ACM Transactions on Mathematical Software*, 31(4), 2005.

[12] H. Liu, J. Lafferty, and L. Wasserman. Sparse nonparametric density estimation in high dimensions using the rodeo. *JMLR Workshop and Conference Proceedings*, 2:283–290, 2007.

[13] P. Mills. Efficient statistical classification of satellite measurements. *International Journal of Remote Sensing*, 32(21):6109–6132, 2011.

[14] A. Murarasu, G. Buse, D. Püger, J. Weidendorfer, and A. Bode. Fastsg: A fast routines library for sparse grids. *Procedia Computer Science*, 9(0):354–363, 2012.

[15] K. Murphy. *The Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[16] F. Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[17] B. Peherstorfer. *Model Order Reduction Reduction of Parametrized Systems with Sparse Grid Learning Techniques*. PhD thesis, TU München, 2013.

[18] D. Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems*. Dr. Hut, 2010.

[19] P. Ram and A. G. Gray. Density estimation trees. In *KDD*, pages 627–635, 2011.

[20] S. G. Roberts and S. Bolt. A note on the convergence analysis of a sparse grid multivariate probability density estimator. In G. N. Mercer and A. J. Roberts, editors, *Proceedings of CTAC-2008*, volume 50 of *ANZIAM J.*, pages C858–C870, Mar. 2009.

[21] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Crc Pr Inc, 1986.

[22] P. Smyth and D. Wolpert. Linearly combining density estimators via stacking. *Machine Learning*, 36(1-2):59–83, 1999.