Journal of Complexity 26 (2010) 508–522

Contents lists available at ScienceDirect

Journal of Complexity

journal homepage: www.elsevier.com/locate/jco

Spatially adaptive sparse grids for high-dimensional data-driven problems

Dirk Pflüger*, Benjamin Peherstorfer, Hans-Joachim Bungartz

Institut für Informatik, Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany

ARTICLE INFO

Article history:
Received 4 June 2009
Accepted 6 April 2010
Available online 18 April 2010

Keywords:
Spatially adaptive sparse grids
High-dimensional approximation
Classification
Regularization
Non-smooth functions

ABSTRACT

Sparse grids allow one to employ grid-based discretization methods in data-driven problems. We present an extension of the classical sparse grid approach that allows us to tackle high-dimensional problems by spatially adaptive refinement, modified ansatz functions, and efficient regularization techniques. The competitiveness of this method is shown for typical benchmark problems with up to 166 dimensions for classification in data mining, pointing out properties of sparse grids in this context. To gain insight into the adaptive refinement and to examine the scope for further improvements, the approximation of non-smooth indicator functions with adaptive sparse grids has been studied as a model problem. As an example for an improved adaptive grid refinement, we present results for an edge-detection strategy.

1. Introduction

In computational science and engineering, one can currently observe a shift of focus from merely compute-driven problems to data-driven problems. Data-driven problems are particularly challenging due to the increasing size, complexity, and dimensionality of the underlying data. Prominent examples for the latter are parameterized data in parameter identification or optimization, as well as classification. While most of the established approaches to classification, such as decision trees, nearest-neighbor techniques, support vector machines, and neural networks, directly address the given data, the spatial resolution of the feature space has not been considered seriously until recently. This is mainly due to the so-called curse of dimensionality, the fact that the cost of spatial numerical discretization typically grows exponentially in the number of dimensions.

* Corresponding author.
E-mail address: Dirk.Pflueger@in.tum.de (D. Pflüger).

0885-064X/$ – see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jco.2010.04.001

Actually, the curse of dimensionality is a roadblock for numerical algorithms in various fields where higher dimensionality occurs, ranging from numerical quadrature to partial differential equations (PDEs), and this topic has been intensively studied in the context of ε-complexity [30]. Since their introduction in 1990 for the solution of partial differential equations [33], sparse grids have focused on mitigating to circumventing the curse of dimensionality; for a discussion in the context of ε-complexity, see [4], and for a general survey on sparse grids, see [5]. Against this background, it was straightforward to explore sparse grids also for classification problems. Note that the underlying principle, a sparse tensor product decomposition, can be traced back to the Russian literature [2,28].

In this regard, the first approach [12] (and therefore the mainstream of the work so far) has been based on the so-called combination technique [16]. There, the solution is obtained by a superposition of solutions on a number of small standard full grids. This has the advantage that highly efficient and readily available solvers, for example multigrid methods, can be used, and that the individual solutions can be obtained in parallel. But even though the combination technique has been extended to allow for dimension-adaptive refinement [17], there is no straightforward way for a spatial and therefore arbitrary (with respect to the sparse grid structure) local refinement.

Recently, direct sparse grid discretization has been employed as well, enabling an algorithmically more complicated, but fully adaptive, discretization approach to classification. So far, the first promising results have been shown [6,25]. Here, we extend the direct sparse grid classification approach to high-dimensional settings and provide examples for different typical benchmark problems. We study criteria for adaptive refinement that take into account the non-smooth functions to be approximated, and we introduce a different error measure tailored for classification problems in contrast to the original PDE context of sparse grids.

The remainder of this article is organized as follows. In Section 2, we present the essentials of, first, the classification problem as we will consider it, i.e. as a scattered data approximation problem, and, second, the concept of sparse grid discretization that enables us to efficiently classify in high-dimensional feature spaces. The necessary modifications and extensions of sparse grids are discussed in Section 3. Section 4, then, presents three typical benchmark data sets for which the attractiveness of our approach is demonstrated. The numerical results obtained are analyzed in Section 5, based on the introduction of an appropriate error measure, the so-called data mining error. Finally, the concluding remarks of Section 6 close the discussion.

2. Problem formulation and sparse grids

We consider the problem of classification in the field of predictive modeling in data mining as a scattered data approximation problem. An unknown function $f : \mathbb{R}^d \to T$ in some function space V, depending on a set of d parameters (or features), is to be reconstructed—or at least approximated. All further knowledge we have about f is a set of so-called training data,

$$S = \left\{(\vec{x}_i, y_i) \in \mathbb{R}^d \times T\right\}_{i=1}^{m},$$

i.e., data points of which the target values $y_i \in T$ are known. We assume that S has been obtained by some given sampling of f, and that the sampling process was disturbed by an unknown amount of noise. The goal of the reconstruction of f is therefore not to interpolate the training data, but to generalize well enough to give good predictions on new, previously unseen data.

As we are dealing with finite sets of data points, the interesting part of the feature space can be normalized to fit in $[0, 1]^d$ without loss of generality. In the case of binary classification, which is our main interest, we restrict the function values to the two class labels ± 1, i.e., $T = \{-1, +1\}$, yielding training data $(\vec{x}_i, y_i) \in [0, 1]^d \times \{-1, +1\}$. (In the related task of regression we would set $T = \mathbb{R}$.)

As we restrict ourselves to reconstructions f_N of f in some subspace $V_N \subset V$ of finite dimensionality N, we introduce a basis $\Phi = \{\varphi_i(\vec{x})\}_{i=1}^{N}$ that spans V_N. The reconstruction f_N can then be written as a linear combination with coefficients α_i,

$$f_N(\vec{x}) = \sum_{i=1}^{N} \alpha_i \varphi_i(\vec{x}). \tag{1}$$

To deal with noise and to obtain a well-posed, uniquely solvable problem, we employ Tikhonov-style regularization [29], enforce a certain smoothness of f_N, and obtain as a regularization network ansatz [31] a variational problem. Following [12], we are solving the regularized least squares problem

$$f_N \stackrel{!}{=} \arg\min_{f_N \in V_N} \left(\frac{1}{m} \sum_{i=1}^{m} \left(y_i - f_N(\vec{x}_i)\right)^2 + \lambda \|\nabla f_N\|_{L_2}^2 \right). \tag{2}$$

The cost or error function $\frac{1}{m}\sum_{i=1}^{m}\left(y_i - f_N(\vec{x}_i)\right)^2$ guarantees closeness of f_N to the training data, whereas the regularization operator $\|\nabla f_N\|_{L_2}^2$ incorporates the smoothness assumption in classification: data points which are close to each other are very likely to have a similar function value. The trade-off between error and smoothness can be influenced by choosing appropriate values for the regularization parameter λ, which can be achieved for example with cross-validation techniques [1,14]. Note that other classification methods, such as neural networks or support vector machines, can be formulated as the same approach, choosing different error and smoothness functionals [8].

Minimization then leads to a system of linear equations,

$$\left(\frac{1}{m} BB^T + \lambda C\right) \vec{\alpha} = \frac{1}{m} B\vec{y},$$

for the coefficient vector $\vec{\alpha}$ of f_N with respect to Φ. The matrix C, $(C)_{ij} = \int \nabla\varphi_i(\vec{x})\nabla\varphi_j(\vec{x})\mathrm{d}\vec{x}$, corresponds to the smoothness functional; the matrices B and B^T, $(B)_{ij} = \varphi_i(\vec{x}_j)$, correspond to the error function. The vector $\vec{y}$ is the vector of the class labels y_i. To set up the linear system, we have to select V_N (which should provide a good approximation to the original problem) and suitable basis functions φ_i (which should allow an efficient computation of $\vec{\alpha}$). Conventional classification algorithms mainly choose global basis functions associated to data points to reconstruct f, trying to adapt to the data with as few ansatz functions as possible. But due to the high degree of data dependence, they typically scale at least quadratically or even worse in the number of training data m.

To obtain an algorithm scaling only linearly in m, we discretize the feature space and employ basis functions associated to grid points. Unfortunately, a straightforward conventional finite element discretization with an equidistant mesh width $h_n = 2^{-n}$ suffers the curse of dimensionality: the number of grid points is of the order $\mathcal{O}(h_n^{-d}) = \mathcal{O}(2^{nd})$, depending exponentially on d. For reasonable mesh widths, only low-dimensional problems are feasible.

Here sparse grids come into play. They help to overcome the curse of dimensionality to some extent, reducing the number of grid points to $\mathcal{O}(h_n^{-1}(\log h_n^{-1})^{d-1})$ with only a slightly deteriorated accuracy if the underlying function f is sufficiently smooth (the mixed second derivatives have to be bounded; see also the discussion in Section 5). Therefore, they are well suited for the grid-point-oriented approach [12].

As the underlying principle is a hierarchical system of basis functions, we use for the space of piecewise d-linear functions the one-dimensional hierarchical hat functions

$$\varphi_{l,i}(x) = \varphi(2^l x - i)$$

(depending on a level l and an index i) which are based on the standard hat function

$$\varphi(x) = \max(1 - |x|, 0).$$

They are extended to the d-dimensional case via a tensor product approach,

$$\varphi_{\vec{l},\vec{i}}(\vec{x}) := \prod_{j=1}^{d} \varphi_{l_j,i_j}(x_j),$$

with $\vec{l}$ and $\vec{i}$ denoting multi-indices that indicate the level and index for each dimension. This leads to a set of subspaces $W_{\vec{l}}$, spanned by the basis

$$\Phi_{W_{\vec{l}}} := \left\{\varphi_{\vec{l},\vec{i}}(\vec{x}) : i_j = 1, \ldots, 2^{l_j} - 1,\ i_j \text{ odd},\ j = 1, \ldots, d\right\}.$$

Note that all basis functions in a $W_{\vec{l}}$ have supports of the same size and shape with pairwise disjoint interiors, and the union of the supports covers the whole domain.

The hierarchical representation now allows one to select only those subspaces that contribute most to the overall solution. This can be done by an a priori selection (see [5] for details). We then obtain a sparse grid space such as

$$V_n^{(1)} := \bigoplus_{|\vec{l}|_1 \leq n+d-1} W_{\vec{l}},$$

which in this case is optimized with respect to the L_2-norm.

So far, the sparse grid combination technique [16] has been used in the context of data mining not only for classification, but also for regression and semi-supervised learning; see, e.g., [12,19,11, 9,13]. A dimension-adaptive extension [17] allows one to enlarge the range of dimensionalities that can be handled even further, employing a scheme that allows different resolutions in the different coordinate directions. Furthermore, an optimized combination technique has been introduced [18], repairing instabilities of the classical combination technique.

We follow a rather different approach, solving (2) directly in the sparse grid space. There, we obtain one single, but larger, linear system of equations. Due to the hierarchical structure, one has to deal with challenging, multi-recursive algorithms. But this approach allows for a direct, spatial adaptivity, and allows one to reach to even-higher-dimensional problems. In previous work, we have already shown a few preliminary and promising results [6,25]. In the following section, we present which additional modifications enable us to compute a wide variety of classification problems.

3. Reaching to higher dimensionalities

Even though sparse grids allow one to overcome the curse of dimensionality to some extent—about 10–15 dimensions are not out of scope any more in many settings—problems in even-higher dimensionalities still need further considerations. Frequently, not all dimensions are equally important in real-world settings; there, dimensionally adaptive sparse grids have been successfully employed, spending fewer grid points in less important directions [10].

Following the direct sparse grid discretization approach, we can use spatial (local) adaptivity in a straightforward way. This can further reduce the number of unknowns that are needed to solve a problem up to some required accuracy. Using a suitable adaptivity criterion, more work can be spent in those regions of the feature space that are most important. In our machine learning settings, the advantage over pure dimension-adaptive (regular) refinement can become quite relevant, as the spatial adaptivity can avoid improving the accuracy in only some part of the feature space, but at the same time losing everything due to overfitting in other regions (see the first example in Section 4).

In classification, it is reasonable to employ spatial adaptivity in any case. For example, the underlying functions feature steep regions (where the class assignment changes) as well as flat ones (within parts of the feature space belonging to the same class). Furthermore, the problem-independent a priori refinement by restricting grid points to the sparse grid structure can be adapted by an a posteriori refinement to exploit the special characteristics of the problem at hand. However, to tackle even-higher-dimensional problems, further considerations have to be made. The number of grid points that is needed for even very coarse grids as well as the cost for the application of the standard regularization operator C depend exponentially on the dimensionality, even if the underlying problem's inherent dimensionality is low.

Regarding the number of grid points, as we usually have to allow non-zero boundary conditions, we have to spend two more degrees of freedom in the one-dimensional hierarchical scheme for the grid on level 1 (which up to now only used $\varphi_{1,1}$), namely the basis functions with level 0 and indices 0 and 1. This leads in the d-dimensional case to 3^d unknowns for the initial grid and introduces an exponential dependence on the dimensionality that is significant for practical computations. For a problem in 100 dimensions, we could not even start computing the solution, even if there is only one relevant dimension. Therefore, grid points on the boundary have to be omitted. Instead, the basis

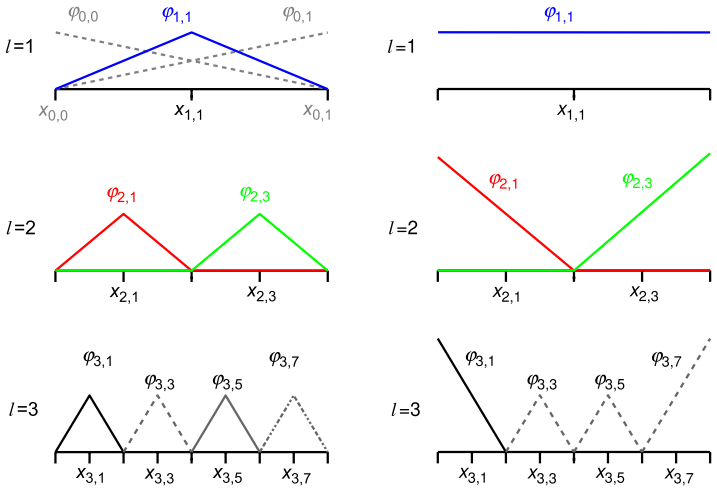

Fig. 1. The classical one-dimensional hierarchical hat basis functions with boundary basis functions on level 0, dashed (left) and the modified basis functions (right).

functions adjacent to the boundary have to be modified. A good choice is

$$\varphi_{l,i}(x) := \begin{cases} 1 & \text{if } l = 1 \wedge i = 1, \\ \left\{ \begin{cases} 2 - 2^l \cdot x & \text{if } x \in \left[0, \frac{1}{2^{l-1}}\right] \\ 0 & \text{else} \end{cases} \right\} & \text{if } l > 1 \wedge i = 1, \\ \left\{ \begin{cases} 2^l \cdot x + 1 - i & \text{if } x \in \left[1 - \frac{1}{2^{l-1}}, 1\right] \\ 0 & \text{else} \end{cases} \right\} & \text{if } l > 1 \wedge i = 2^l - 1, \\ \varphi\left(x \cdot 2^l - i\right) & \text{else} \end{cases}$$

for the one-dimensional basis functions, extrapolating linearly towards the boundary [26,6]; see Fig. 1. This allows one to start with only $\mathcal{O}(d)$ basis functions (one center point and one pair of points for probing in each coordinate direction); a suitable refinement will then create grid points where it is really necessary.

The application of the stiffness matrix C resulting from the regularization functional $\|\nabla f\|_{L_2}^2$ in (2) is complicated and multi-recursive (see, e.g., [26]), but it can be applied to a vector in $\mathcal{O}(2^d N)$ operations. This is clearly linear in the number of unknowns N; nevertheless, the algorithmic complexity depends exponentially on the dimensionality. Therefore, a simpler functional has to be used, exploiting the inherent smoothness of the hierarchical basis which is known to be spectrally close to the Laplacian [15]. Examining several regularization operators [3], we found out that choosing the Euclidean norm of the (hierarchical) coefficient vector yields very similar results in classification tasks.

Note that not even a scaling of the coefficients of different levels is necessary. This is due to the hierarchical structure of the basis (as functions on lower levels are generally non-zero for more data points and are therefore more often penalized by the error functional) and works only in the explicit sparse grid basis. For the combination technique, examples that do not work using this regularization term can be easily constructed.

Now the minimization problem reads

$$\arg\min_{f_N \in V_N} \left(\frac{1}{m} \sum_{i=1}^{m} \left(y_i - f_N(\vec{x}_i)\right)^2 + \lambda \sum_{i=1}^{N} \alpha_i^2 \right) \tag{3}$$

and results in the linear system

$$\left(\frac{1}{m}BB^T + \lambda I\right)\vec{\alpha} = \frac{1}{m}B\vec{y},$$

with I being the identity matrix. The system matrix is still symmetric and positive definite. The matrices B and B^T can be applied in $\mathcal{O}(m(\log N)^d)$ for regular sparse grids, λI obviously in $\mathcal{O}(N)$. Furthermore, the matrix–vector multiplications can be parallelized easily across the data points.

With these modifications, even high-dimensional classification tasks can be solved, as we demonstrate in the following section.

4. Classification examples

For three typical benchmark data sets used in data mining, we show numerical results, demonstrating distinct properties of sparse grids in classification. The first example is an artificial data set, exhibiting typical properties of classification problems, and visualizing adaptive refinement. The second is a 64-dimensional real-world task with the additional challenge of being a ten-class problem. The third data set, finally, has 166 dimensions and only few data points; both the high dimensionality and the risk of overfitting have to be dealt with. As sparse grid approximants are functions over the reals, we restrict ourselves here to examples with real-valued attributes only; in general, typical techniques to transform nominal attributes to numeric ones, e.g. substituting a nominal attribute by multiple binary ones, can, of course, be used.

The quality of the fit depends mainly upon two free parameters. Additionally to the regularization parameter λ, which has to be determined, we also have to specify the number of refinement steps (for a given refinement strategy). A good choice for both can be found using a validation set. The training data is split into two data sets. One of them is actually used to construct the classifier for different parameter combinations. The parameter set giving the best performance on the remaining validation set is then used to train on the whole training set, resulting in the classifier used on the test set. In the following examples, we selected one third of the training set as validation data randomly.

The adaptive refinement of the sparse grid can be done in different ways. Here, we picked the most straightforward refinement criterion, which already proved to give good results. Out of the grid points that can be refined, the one with the highest absolute value of the corresponding surplus (coefficient in the hierarchical basis) is refined first. In the context of PDEs it is usually more beneficial to refine more than one grid point at once, for example a certain percentage of the unrefined ones. Usually, the same holds for classification tasks with moderate dimensionalities. But in higher-dimensional settings or if the data set is relatively small, quite often only a few refinements can be spent anyway. In the first case, a broad refinement quickly leads to more grid points than can be handled, whereas in the latter case overfitting can already occur after very few refinements (as is the case in the third example).

The first example is an artificial classification task, taken from [27]. Being only two-dimensional, the sparse grid as well as the separation manifold can be visualized. The example consists of two data sets, one for training and one for testing, and has been constructed to comprise 8% of error. It shows typical characteristics of real-world data sets, as it is neither linearly separable nor very complicated. Fig. 2 shows the two data sets as well as the best separation manifold obtained by an adaptive sparse grid. Already eight refinement steps are enough to obtain an excellent accuracy on the test data of 91.5%, out of a maximum of 92%.

It can be seen that the separation boundary is resolved better in the central, critical region than in regions where very little information (data points) about the underlying function is given. This is due to the adaptive refinement. Interestingly, after only eight refinements, overfitting starts to take over and the accuracy deteriorates. For this data set a mere dimension-adaptive refinement leads to a lower accuracy: whereas more grid points towards the center of the feature space are necessary, using the same discretization level in the same direction towards the boundary leads to local overfitting there.

The second data set has been taken from the UCI repository [21], consisting of 3823 data points for training and 1797 to test on. The aim is to optically recognize handwritten digits, discriminating the digits 0–9. Originally, the digits have been written on a 32×32 checkerboard pattern. For each

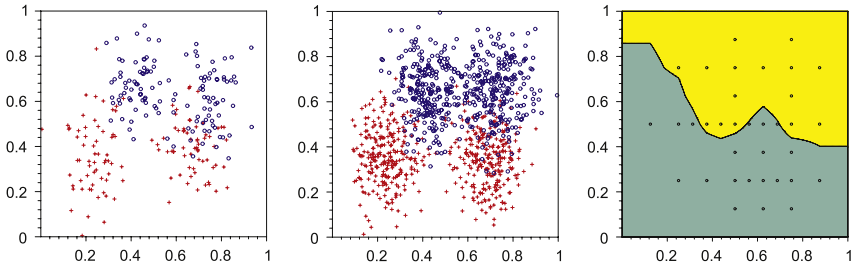

Fig. 2. Ripley data set: 250 data points for training (left), 1000 to test on (middle), and the classification areas together with the corresponding sparse grid (right).

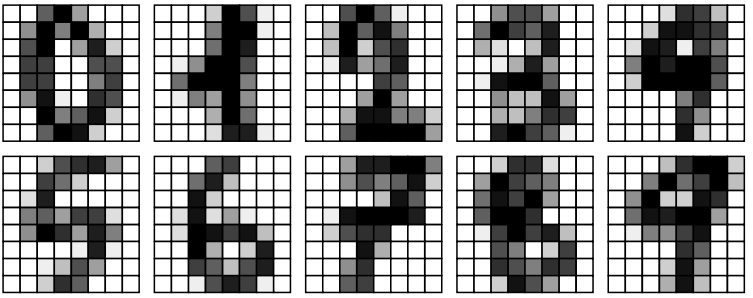

Fig. 3. Optical recognition of handwritten digits [21]; some example data points.

of the fields it was measured whether the pen touched it or not, leading to 1024 binary features. In a preprocessing step, 4×4 patterns have been compressed to a single "gray value" by counting the number of touched fields. A data point is therefore a 64-dimensional vector $\vec{x}, x_i \in \{0, 1, \ldots, 16\}$, $i = 1, \ldots, 64$; see Fig. 3 for some examples.

To tackle the multi-class classification problem, we could introduce ten distinct function values as class target labels, for example the values of the digits, solve the minimization problem (3) and compute a single function. Evaluating the classification function, we would then cut off just in between the target values to assign class predictions to new data points. This approach yields low accuracies as the quality depends a lot on the ordering of the target labels. Artificial examples that introduce a lot of noise along the classification boundary can be constructed easily, where two regions next to each other with different class assignments have class values that are not neighboring on the target axis.

A better approach is to compute ten different sparse grid classification functions $f_i(\vec{x}), i = 0, \ldots, 9$, discriminating digit i from the others. To obtain a class prediction, we can make use of the fact that—in contrast to some other types of classification algorithms such as decision trees—each f_i is real-valued, and consider $f_i(\vec{x})$ as a measure of confidence of f_i in the class prediction for digit i. This makes sense, as the confidence in a correct prediction is low close to the separation manifold and high in regions where a neighborhood belongs to the same class.

If for each f_i the target label of the corresponding class i is $+1$, we can then predict the target j for new data points $\vec{x}$ straightforwardly as

$$\arg\max_{j \in \{0,\ldots,9\}} f_j(\vec{x}).$$

An optimized combination of the functions (which takes into account the frequency of data points belonging to a certain class, for example) could further improve the accuracy. Even without more sophisticated techniques, the accuracy on the data set is already competitive. Table 1 shows the performance of sparse grids in comparison to results taken from two benchmark studies [22,7]. Only k-nearest-neighbor (k-NN) techniques provided better results than our approach.

Table 1
Comparison with two benchmark studies [22,7] for the optical digit recognition.

Classification method	Accuracy (%)
k-NN	98.00
Adaptive sparse grids	*97.74*
RBF-DDA networks	97.45
Support vector machines	97.27
AdaBoost	95.33
MLP	89.05

Again, using adaptive refinement was crucial. For example, to be able to distinguish the digit 2 well enough from the others, the sparse grid had to be refined until the first grid points on level 5 were created. Whereas a 64-dimensional regular sparse grid up to level 5 has 12,638,213 degrees of freedom, clearly exceeding the size of the training data, the adaptively refined grid consisted only of 1760 grid points and was therefore able to generalize from the given data, i.e., to provide good predictions on new data points.

The third classification task deals with separating musk molecules from non-musk ones [21]. The rather small musk-version-1 data set (476 data points) consists of 166 attributes that describe mainly distance features of conformations of the molecules. For this data set we conducted a benchmark study [20], comparing 45 classification and meta-classification algorithms provided by the WEKA toolkit [32], 38 of which were able to tackle the musk data set. For all algorithms the two most critical parameters (if available) have been optimized by a search algorithm to determine a good choice of values within reasonable ranges. As this had neither been tailored for the sparse grid technique nor for any other one, the heuristics did not guarantee to find "the" optimal parameter set, but for all tests conducted on a lot of benchmark data sets they usually did find a reasonably good one.

The results for the musk classification task were obtained by performing 10×10-fold cross-validation. Fig. 4 shows the mean accuracy and the standard deviation over all 100 subtasks for the nine best algorithms. The sparse grid classification shows a competitive performance ($88.83\% \pm 4.72\%$ accuracy), even though the setting is very challenging. Starting with 333 grid points on level two, training on only 283 data points each, we already have more unknowns than function values. A higher degree of smoothness is needed to avoid overfitting, and only very few refinements can be spent; we therefore refined at most twice. Comparing the standard deviation, sparse grids have the second lowest one within the top ten algorithms, indicating a good robustness of the classification. If we had selected a smaller parameter range around $\lambda = 0.01$, we would have even obtained accuracies of about $89.77\% \pm 4.51\%$, which indicates that there could still be room for improvement.

Considering the run-time for the classification task, reaching with 166 dimensions the limit in terms of dimensionality so far, the sparse grid classifier took more time than the other classification algorithms, of course. Including 10×10-fold cross-validation, adaptivity and parameter search with a validation set, it took us about 1220 h (serial execution time) altogether. This comprised roughly 296,000 iterations of a conjugated gradient solver (un-preconditioned). In contrast, the fastNN neural network classifier took only 11 h, the support vector machine implementation (libSVM) about 5 h, and most of the other classification algorithms even less. Note that there is clearly room for improvements, as our code is rather general and not optimized just for classification purposes; applying a better, preconditioned solver should reduce the run-times, for example. Note further that we have been dealing with a rather small data set. If the amount of training data is big enough, sparse grids can outperform most other methods, as our algorithms scale only linearly in the number of data points, in contrast to quadratically or even worse, as is the case for most conventional classification algorithms.

If we preprocess the data set and reduce the dimensionality by principal component analysis (PCA), keeping 95% of the variance, we obtain a problem which is only 35-dimensional. Sparse grids yield even better results in this setting; see Fig. 5. We can refine more often before the number of grid points exceeds the number of data points. Therefore adaptivity can better adapt to the structure of the underlying function. Again, the standard deviation is lower than for most of the other classification algorithms.

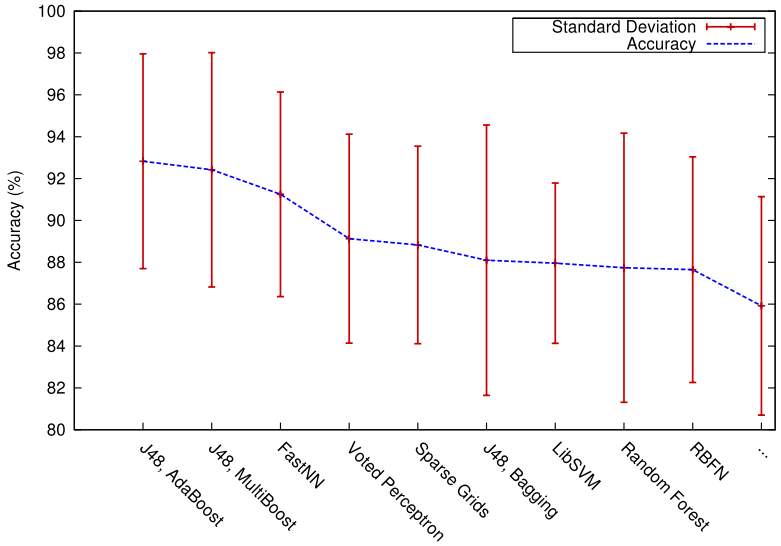

Fig. 4. Benchmark study (best nine out of 38 algorithms) for the 166-dimensional musk1 data set, UCI repository, 10×10-fold cross-validation results.

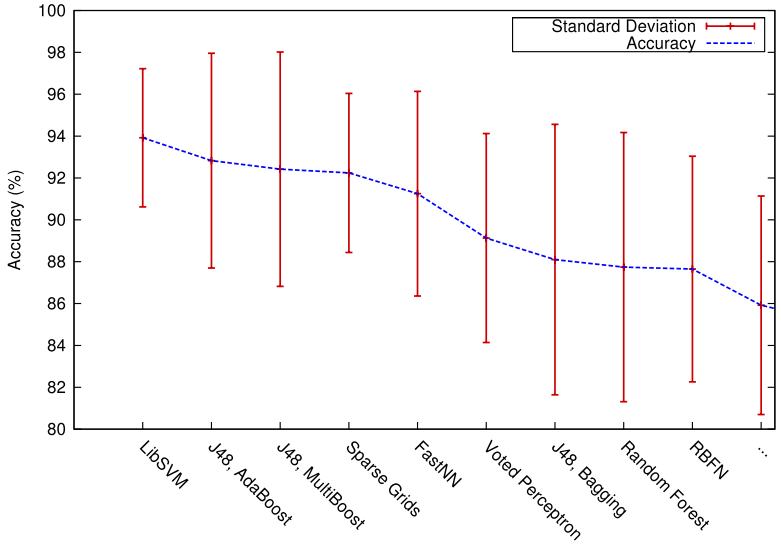

Fig. 5. Benchmark study (best nine out of 38 algorithms) for the musk1 data set, UCI repository, after reduction to 35 dimensions with PCA, keeping 95% of the variance; 10×10-fold cross-validation results.

5. Error measure

As demonstrated, benchmark examples show that classification tasks can be successfully tackled with adaptively refined sparse grids, even though only a direct, surplus-based refinement strategy has been used so far. To gain deeper insight in the mechanisms leading to the competitive performance, indicator functions that have a simple structure, but still exhibit typical characteristics of classification functions, have been studied. Even though the following examples are of rather academic nature, the results illustrate the potential for further improvements.

As described in Section 2, the error estimates for piecewise d-linear interpolation and quadrature for regular sparse grids hold only for sufficiently smooth functions. Although the decision functions in real-life applications are unknown, we know that they are obviously non-smooth functions: they are even discontinuous, jumping from the class label -1 to $+1$ at the manifolds separating the two classes. Therefore, the classical estimates do not hold. Adaptivity has to be used to cope with the discontinuities and to maintain good convergence rates.

Let us assume that we sample our function f at an equidistributed sequence of points S. If we spend more and more evaluations, then the minimization of the pointwise error term $1/m \sum_{i=1}^{m} (y_i - f_N(\vec{x}_i))^2$ in (3) approaches (for $m \to \infty$) the minimization of $\|f - f_N\|_{L_2}^2$. Transferring the setting of classification to the continuous case as well and restricting ourselves to non-noisy data, we can examine indicator functions that represent simple classification tasks. To keep things as simple as possible, we restrict ourselves to indicator functions

$$f : [0, 1]^d \to \{0, 1\}$$

with Dirichlet zero boundary conditions and a single compact and convex non-zero region. (We deviate from the classical definition with class labels $\{-1, 1\}$ to avoid any influence by a certain choice of boundary treatment in the sparse grids basis.) The separation manifold is then

$$\left\{\vec{x} \mid f(\vec{x}) = 0.5\right\}$$

and not, as in the original formulation, at the zero-crossing.

Solving the continuous counterpart of (3), we are looking for the so-called best approximation $f_N = \sum_{i=1}^{N} \alpha_i \varphi_i$ with respect to the L_2-norm plus a regularization term in some sparse grid space V_N,

$$\arg\min_{f_N \in V_N} \left(\|f - f_N\|_{L_2}^2 + \lambda \frac{1}{2} \sum_{i=1}^{N} \alpha_i^2 \right),$$

and we have to solve the linear system

$$(B + \lambda I)\, \vec{\alpha} = \vec{b}$$

with $(B)_{ij} = \int \varphi_i(\vec{x}) \varphi_j(\vec{x})\, \mathrm{d}\vec{x}$ and $b_i = \int f(\vec{x}) \varphi_i(\vec{x})\, \mathrm{d}\vec{x}$.

The pure surplus-based refinement strategy is known to tend to minimize norms such as the L_2-norm of the error; therefore it is clearly a good choice for numerical interpolation and quadrature. But for binary classification, the aim is rather to minimize what we call the data mining error, i.e. the misclassified volume within the feature space, and not the L_2-norm.

To measure the data mining error, we define the best approximation indicator function as in [23] to be

$$f_N^{\mathrm{Ind}}(\vec{x}) := \begin{cases} 0, & f_N(\vec{x}) < 0.5, \\ 1, & f_N(\vec{x}) \geq 0.5. \end{cases}$$

The data mining error is the set

$$X^{\mathrm{err}} := \left\{\vec{x} \mid f_N^{\mathrm{Ind}}(\vec{x}) \neq f(\vec{x})\right\},$$

the misclassified volume of the feature space.

For a given indicator function f and a regularization parameter λ, we can then compute the best approximation f_N by solving the system of linear equations and inspect the error. Having an infinite non-noisy training data set, the quality of the solution is far less sensitive to the choice of λ than in the non-continuous classification setting. In the following, we can even set $\lambda = 0$ and neglect the regularization term. Finding the best approximation with respect to the L_2-norm is a well-posed problem.

In low dimensionalities, we can compute the error terms numerically. For example, we can use Marching Squares (or Marching Cubes, respectively) or octrees to determine the misclassified volume. In higher dimensionalities we can still approximate the error stochastically with Monte Carlo or quasi-Monte Carlo methods.

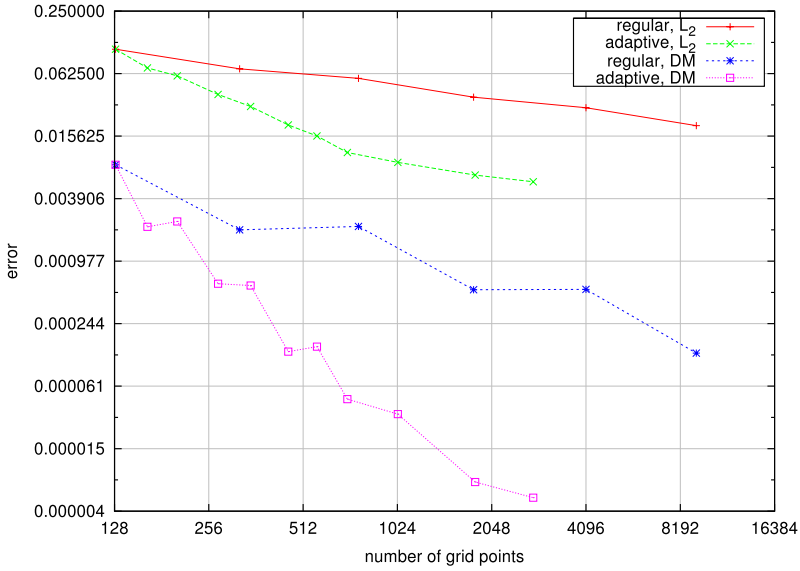

Fig. 6. Convergence of the L_2 error and the data mining error for the indicator function $f_\square$ and both regular and adaptive sparse grids.

Looking at the two-dimensional paraxial indicator function

$$f_\square(\vec{x}) := \begin{cases} 1, & \vec{x} \in [0.15, 0.85]^2 \\ 0, & \text{else} \end{cases}$$

we can observe a few properties; see Fig. 6. Due to the violation of the smoothness requirements, the L_2 error converges slightly worse than $\mathcal{O}(\sqrt{h})$, and much worse than $\mathcal{O}(h^2(\log 1/h)^{d-1})$ as in the smooth case. Fortunately, the convergence of the data mining error is much better, approximately $\mathcal{O}(h)$. But for this measure (which is not the norm from the minimization problem), using more grid points does not necessarily lead to a better accuracy: the error can increase, depending on how the separation manifold is located relative to the sparse grid structure.

If we start with a regular grid (here for level 5) and employ the straightforward, surplus-based refinement strategy, refining 10% of the grid points in every step, then we can even cope with the discontinuities to a large extent: the convergence of the data mining error is about in $\mathcal{O}(h^2)$; the L_2 error improves to (roughly) about $\mathcal{O}(h)$.

Still, the refinement criterion does not target the data mining error, and so there is plenty of room for improvement. It enforces the reduction of the L_2-norm in regions belonging to the same class, guaranteeing a close fit in those regions as well. In a context where the priority is to represent a discontinuity as well as possible, this should be avoided. Refining primarily along the separation manifold requires refinement along the jumps; the adaptivity has to somehow detect them.

Investigating the individual surpluses α_i as well as the accumulated contributions of the subspaces $W_{\vec{l}}$ for several indicator functions [24], we examined different strategies for the refinement criteria. First, we have implemented a very greedy refinement strategy, where we pick the refinement candidate which reduces the error measure most. Even though we spend a lot of effort—we have to refine our grid for every possible grid point, compute how much this reduces the overall data mining error for all those grids, and pick the best one—we usually run into local minima until a very high grid level is reached locally before refining in other, more critical regions.

To demonstrate the potential of criteria for adaptive refinement, we have implemented an edge-detection strategy, as further observations reconfirmed that the orientation of the separation manifold plays an important role. Due to the tensor product structure, basis functions with a long, narrow

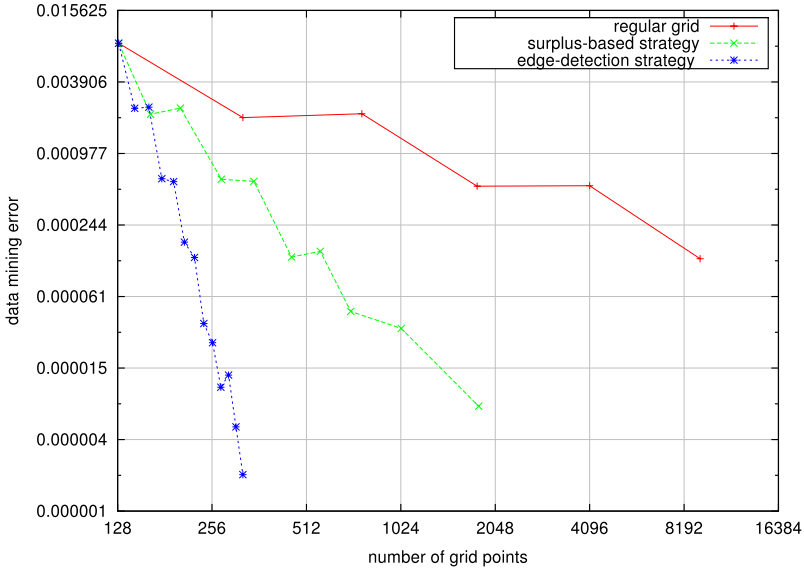

Fig. 7. Convergence of the data mining error for the indicator function $f_\square$ for regular sparse grids and both the surplus-based and the edge-detection strategy refinements.

support ($\vec{l} \approx (1, \ldots, 1, n, 1, \ldots, 1)^T$) are better suited to approximate paraxial edges, but basis functions with a more quadratic support ($\vec{l} \approx (n, \ldots, n)^T$) are needed for non-paraxial edges.

For each refinement step we proceed as follows. First, we examine all grid points for a given level n with a nearly quadratic support. We then determine (using our knowledge about f) whether it is crossed by a non-paraxial edge and create it if this is the case. Then we proceed likewise with the basis functions with long support and approximately paraxial edges. (As we are dealing with simple indicator functions, we do not have to consider so far to what extent jumps are "approximately" paraxial or not.) Note that for the grid points in question the hierarchical ancestors (that are needed to determine the hierarchical surplus in the new grid point) in the sparse grid structure do not have to exist yet and that they may have to be created if the corresponding grid point is selected for refinement.

Here, we use our knowledge about the indicator functions to easily determine whether and how the separation manifold intersects a basis function. However, if we have no knowledge about the underlying function, we can still compute the function values of the current sparse grid function f_N at the corners of the support to determine it. Furthermore, this can be done hierarchically recursive in an octree-like manner.

Fig. 7 shows the classical surplus-based refinement criterion in comparison to the edge-detection strategy for the same indicator function $f_\square$. As $f_\square$ is paraxial (apart from the corners), the long basis functions are preferably selected. The resulting sparse grid can quickly adapt to the separation manifold, providing an outstanding convergence of the data mining error.

Of course, the contour line $f_\square(\vec{x}) = 0.5$ is neatly aligned to the main axes of the sparse grid and we cannot expect the convergence to behave as good in the general case. If the underlying functions have some kind of tensor product structure, they are better suited for sparse grids. The (again two-dimensional) indicator function

$$f_\diamond(\vec{x}) := \begin{cases} 1, & |x_1 - 0.4| + |x_2 - 0.4| < 0.2 \\ 0, & \text{else} \end{cases}$$

is completely non-paraxial and not centered. It can be obtained by rotating the non-zero square of $f_\square$ by 45 degrees, scaling and translation.

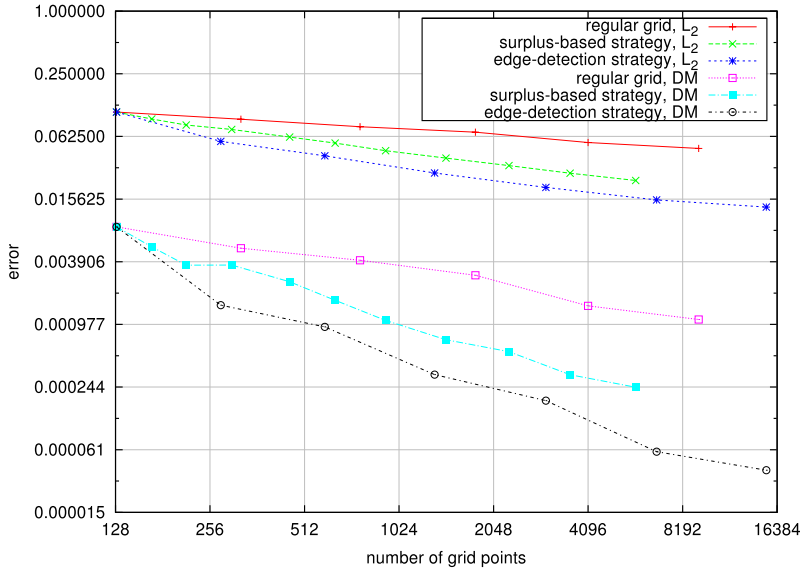

Fig. 8. Convergence of the L_2 error and the data mining error for the indicator function $f_\diamond$ for regular sparse grids and both the surplus-based and the edge-strategy refinements.

As expected, neither the L_2 errors nor the data mining errors converge as well as in the previous example; see Fig. 8. Still, there is some gain in both error measures by choosing the edge-detection strategy rather than the surplus-based one. In high-dimensional settings, the jump detection as described above has to be substituted by a simpler algorithm: looking at all corners of the support would introduce a factor of at least $\mathcal{O}(2^d)$.

Finally, and to give an indication that a similar behavior can be expected in higher-dimensional settings, we show the data mining error for the indicator function

$$f_{\square}^{d}(\vec{x}) := \begin{cases} 1, & \vec{x} \in [0.15, 0.85]^d \\ 0, & \text{else;} \end{cases}$$

see Fig. 9 for $d = 2, 3, 4$. The introduction of additional corners, edges and hyperplanes does not harm the overall behavior, as the setting is paraxial and symmetric in every dimension. Again, for non-paraxial settings, lower convergence rates are encountered.

6. Conclusions

In this article, we have shown that in the setting of classification in data mining even up to 166-dimensional problems can be successfully solved, working directly in the sparse grid space, at least if the inherent dimensionality of the underlying problem is not too high, which is the case for many real-world problems in data-driven settings. Whereas sparse grids already overcome the curse of dimensionality to some extent, spatial adaptivity has to be employed and the classical sparse grid basis and regularization operator have to be modified to be able to extend the field of application to high-dimensional problems. To this end, the boundary of the feature space has to be treated in a special way and a simpler regularization operator can be chosen, exploiting the inherent smoothness of the hierarchical basis.

For three common classification tasks, we showed some properties of spatially adaptive sparse grids and demonstrated the competitiveness of the results. Multi-class problems can be solved as well as high-dimensional problems with very few data, demonstrating the robustness of our approach.

To investigate the potential of better refinement criteria and to examine the effect that classification functions violate the sparse grids' smoothness requirements, non-continuous indicator

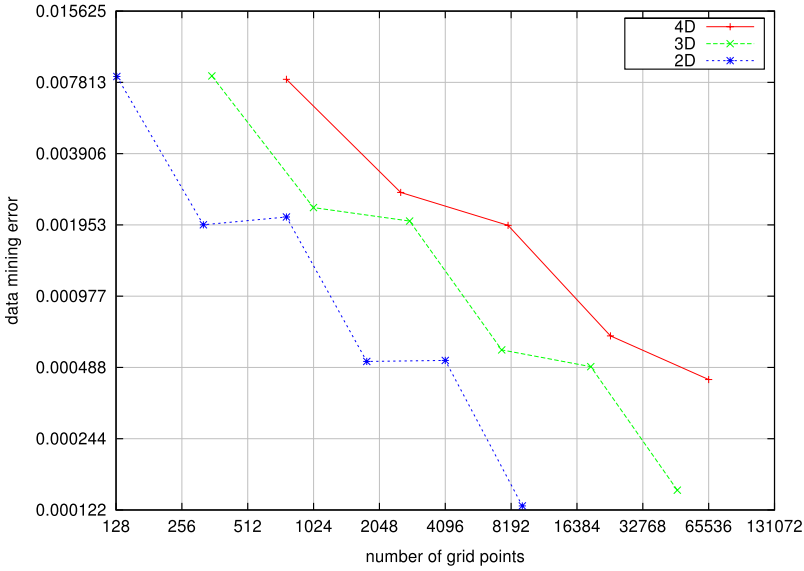

Fig. 9. Convergence of the data mining error for the indicator function $f_{\square}^{d}$ for regular grids in up to four dimensions.

functions have been studied. Whereas the classical surplus-based refinement criterion does not target the classification error, but still provides good results, a refinement strategy detecting edges can further improve the convergence of the error quite significantly, for both paraxial and non-paraxial jumps, but especially in regions where the problem has a tensor-product structure. For use in higher-dimensional applications, a more efficient criterion to detect edges has to be found. Additionally, the minimization criterion itself could be modified to better target the data mining error rather than the least squares error which could help to extend the scope to even-higher dimensionalities.

References

[1] D.M. Allen, The relationship between variable selection and data augmentation and a method for prediction, Technometrics 16 (1) (1974) 125–127.
[2] K. Babenko, Approximation by trigonometric polynomials in a certain class of periodic functions of several variables, Soviet Math. Dokl. 1 (1960) 672–675; Russian original in Dokl. Akad. Nauk SSSR 132 (1960) 982–985.
[3] J. Blank, R. Röttger, Data Mining mit Dünnen Gittern, Student project (SEP), Fakultät für Informatik, Technische Universität München, Jan. 2008.
[4] H.-J. Bungartz, M. Griebel, A note on the complexity of solving Poisson's equation for spaces of bounded mixed derivatives, J. Complexity 15 (2) (1999) 167–199.
[5] H.-J. Bungartz, M. Griebel, Sparse grids, Acta Numer. 13 (2004) 147–269.
[6] H.-J. Bungartz, D. Pflüger, S. Zimmer, Adaptive sparse grid techniques for data mining, in: H. Bock, E. Kostina, X. Hoang, R. Rannacher (Eds.), Modelling, Simulation and Optimization of Complex Processes, Proceedings of the High Performance Scientific Computing 2006, Springer, Hanoi, Vietnam, 2008, pp. 121–130.
[7] V.S. Devi, M.N. Murty, An incremental prototype set building technique, Pattern Recognit. 35 (2) (2002) 505–513.
[8] T. Evgeniou, M. Pontil, T. Poggio, Regularization networks and support vector machines, in: Advances in Computational Mathematics, MIT Press, 2000, pp. 1–50.
[9] J. Garcke, Regression with the optimised combination technique, in: ICML'06: Proceedings of the 23rd International Conference on Machine Learning, ACM Press, New York, NY, USA, 2006, pp. 321–328.
[10] J. Garcke, A dimension adaptive sparse grid combination technique for machine learning, in: W. Read, J.W. Larson, A.J. Roberts (Eds.), Proceedings of the 13th Biennial Computational Techniques and Applications Conference, CTAC-2006, ANZIAM J. 48 (2007), C725–C740.
[11] J. Garcke, M. Griebel, Semi-supervised learning with sparse grids, in: M.-R. Amini, O. Chapelle, R. Ghani (Eds.), Proceedings of ICML, Workshop on Learning with Partially Classified Training Data, 2005, pp. 19–28.
[12] J. Garcke, M. Griebel, M. Thess, Data mining with sparse grids, Computing 67 (3) (2001) 225–253.
[13] J. Garcke, M. Hegland, Fitting multidimensional data using gradient penalties and the sparse grid combination technique, Computing 84 (1–2) (2009) 1–25.
[14] G.H. Golub, M. Heath, G. Wahba, Generalized cross-validation as a method for choosing a good ridge parameter, Technometrics 21 (2) (1979) 215–223.

[15] M. Griebel, P. Oswald, On additive Schwarz preconditioners for sparse grid discretizations, Numer. Math. 66 (1994) 449–464, also as Bericht Math/92/7, Institut für Angewandte Mathematik, Friedrich-Schiller-Universität Jena, 1992.
[16] M. Griebel, M. Schneider, C. Zenger, A combination technique for the solution of sparse grid problems, in: Proceedings of the IMACS International Symposium on Iterative Methods in Linear Algebra, Elsevier, Amsterdam, 1992, pp. 263–281.
[17] M. Hegland, Adaptive sparse grids, in: K. Burrage, R.B. Sidje (Eds.), Proc. of 10th Computational Techniques and Applications Conference, CTAC-2001, vol. 44, 2003, pp. C335–C353.
[18] M. Hegland, J. Garcke, V. Challis, The combination technique and some generalisations, Linear Algebr. Appl. 420 (2–3) (2007) 249–275.
[19] M. Hegland, O.M. Nielsen, Z. Shen, Multidimensional smoothing using hyperbolic interpolatory wavelets, Electron. Trans. Numer. Anal. 17 (2004) 168–180.
[20] L. Johansson, Einsatz der Dünngitterklassifikation für Benchmark-probleme, Diplomarbeit, Fakultät für Informatik, Technische Universität München, Apr. 2008.
[21] D. Newman, S. Hettich, C. Blake, C. Merz, UCI repository of machine learning databases, University of California, Irvine, Dept. of Information and Computer Sciences, 1998.
[22] A.L.I. Oliveira, F.B.L. Neto, S.R.L. Meira, Improving rbf-dda performance on optical character recognition through parameter selection, in: Pattern Recognition, International Conference on, vol. 4, 2004, pp. 625–628.
[23] B. Peherstorfer, Adaptive sparse grid representation for indicator functions, Bachelor's thesis, Fakultät für Informatik, Technische Universität München, Nov. 2008.
[24] B. Peherstorfer, More on Adaptive Sparse Grid Representation for Indicator Functions, Student project (IDP), Fakultät für Informatik, Technische Universität München, Mar. 2009.
[25] D. Pflüger, I.L. Muntean, H.-J. Bungartz, Adaptive sparse grid classification using grid environments, in: Y. Shi, D. van Albada, J. Dongarra, P. Sloot (Eds.), ICCS 2007, International Conference on Computational Science, in: LNCS, vol. 4487, Springer, Berlin, Heidelberg, 2007, pp. 708–715.
[26] D. Pflüger, Data Mining mit Dünnen Gittern, Diplomarbeit, IPVS, Universität Stuttgart, Mar. 2005.
[27] B.D. Ripley, N.L. Hjort, Pattern Recognition and Neural Networks, Cambridge University Press, New York, NY, USA, 1995.
[28] S. Smolyak, Quadrature and interpolation formulas for tensor products of certain classes of functions, Soviet Math. Dokl. 4 (1963) 240–243; Russian original in Dokl. Akad. Nauk SSR, 148 (1963), 1042–1045.
[29] A. Tikhonov, V. Arsenin, Solutions of Ill-Posed Problems, W.H. Winston, Washington, DC, 1977.
[30] J. Traub, G. Wasilkowski, H. Wozniakowski, Information-Based Complexity, Academic Press, London, 1988.
[31] G. Wahba, Spline Models for Observational Data, in: CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 59, SIAM, Philadelphia, 1990.
[32] I.H. Witten, E. Frank, Data mining: practical machine learning tools and techniques with Java implementations, ACM SIGMOD Rec. 31 (1) (2002) 76–77.
[33] C. Zenger, Sparse grids, in: W. Hackbusch (Ed.), Parallel Algorithms for Partial Differential Equations, in: Notes on Numerical Fluid Mechanics, vol. 31, Vieweg, 1991, pp. 241–251.