Data mining with sparse grids

J. Garcke[+], M. Griebel[+] and M. Thess[*]
[+]Institut für Angewandte Mathematik
Abteilung für wissenschaftliches Rechnen und numerische Simulation
Universität Bonn, D-53115 Bonn
{garckej, griebel}@iam.uni-bonn.de
[*]Prudential Systems, Chemnitz
thess@prudsys.com

Abstract

We present a new approach to the classification problem arising in data mining. It is based on the regularization network approach but, in contrast to other methods which employ ansatz functions associated to data points, we use basis functions coming from a grid in the usually high-dimensional feature space for the minimization process. To cope with the curse of dimensionality, we employ sparse grids. Thus, only $O(h_n^{-1} n^{d-1})$ instead of $O(h_n^{-d})$ grid points and unknowns are involved. Here d denotes the dimension of the feature space and $h_n = 2^{-n}$ gives the mesh size. To be precise, we suggest to use the sparse grid combination technique where the classification problem is discretized and solved on a certain sequence of conventional grids with uniform mesh sizes in each coordinate direction. The sparse grid solution is then obtained from the solutions on these different grids by linear combination. In contrast to other sparse grid techniques, the combination method is simpler to use and can be parallelized in a natural and straightforward way.

We describe the sparse grid combination technique for the classification problem in terms of the regularization network approach. We then give implementational details and discuss the complexity of the algorithm. It turns out that the method scales linearly with the number of instances, i.e. the amount of data to be classified. Finally we report on the quality of the classifier build by our new method. Here we consider standard test problems from the UCI repository and problems with huge synthetical data sets in up to 9 dimensions. It turns out that our new method achieves correctness rates which are competitive to that of the best existing methods.

AMS subject classification. 62H30, 65D10, 68T10
Key words. data mining, classification, approximation, sparse grids, combination technique

1 Introduction

Data mining is the process of finding hidden patterns, relations and trends in large data sets. It plays an increasing role in commerce and science. Typical scientific applications are the post-processing of data in medicine (e.g. CT data), the evaluation of data in astrophysics (e.g. telescope and observatory data) the grouping of seismic data or the evaluation of satellite pictures (e.g. NASA earth observing system). Maybe even more important are financial and commercial applications. With the development of the internet and e-commerce there are huge data sets collected more or less automatically which can be used for business decisions and further strategic planning. Here, applications range from the termination and cancellation of contracts, the assessment of credit risks, the segmentation of customers for rate/tariff-planing and advertising campaign letters to fraud detection, stock analysis and turnover prediction.

Usually, the process of data mining (or knowledge discovery) can be separated in the planning step, the preparation phase, the mining phase (i.e. the machine learning) and the evaluation phase. To this end, association-analysis classification, clustering and prognostics are to be performed. For a thorough overview on the various tasks arising in the data mining process see [7, 18].

In this paper we consider the classification problem in detail. Here, a set of data points in d-dimensional feature space is given together with a class label in $\{-1, 1\}$ for example. From this data, a classifier must be constructed which allows to predict the class of any newly given data point for future decision making. Widely used approaches are nearest neighbor methods, decision tree induction, rule learning and memory-based reasoning. There are also classification algorithms based on adaptive multivariate regression splines, neural networks, support vector machines and regularization networks. Interestingly, these latter techniques can be interpreted in the framework of regularization networks [29]. This approach allows a direct description of the most important neural networks and it also allows for an equivalent description of support vector machines and n-term approximation schemes [27]. Here, the classification of data is interpreted as a scattered data approximation problem with certain additional regularization terms in high-dimensional spaces. With these techniques, it is possible to treat quite high-dimensional problems but the amount of data is limited due to complexity reasons. This situation is reversed in many practical applications. Here, after some preprocessing steps, the dimension of the resulting problem is moderate but the amount of data is usually huge. Thus, there is a strong need for methods which can be applied in this situation as well.

In this paper, we present a new approach to the classification problem arising in data mining. It is also based on the regularization network approach but, in contrast to the other methods which employ mostly global ansatz functions associated to data points, we use an independent grid with associated local ansatz functions in the minimization process. This is similar to the numerical treatment of partial differential equations. Here, a uniform grid would result in $O(h_n^{-d})$ grid points, where d denotes the dimension of the feature space, n is the refinement level and $h_n = 2^{-n}$ gives the mesh size. Therefore the complexity of the problem would grow exponentially with d and we encounter the curse of dimensionality. This is probably the reason why conventional grid-based techniques are not used in data mining up to now.

However, there is a special discretization technique using so-called sparse grids which allows to cope with the complexity of the problem, at least to some extent. This method has been originally developed for the solution of partial differential equations [4, 10, 40, 75] and is now used successfully also for integral equations [22, 39], interpolation and approximation [5, 37, 47, 60, 66], eigenvalue problems [25] and integration problems [26]. In the information based complexity community it is also known as 'hyperbolic cross points' and the idea can even be traced back to [63]. For a d-dimensional problem, the sparse grid approach employs only $\mathrm{O}(h_n^{-1}(\log(h_n{}^{-1}))^{d-1})$ grid points in the discretization. The accuracy however is nearly as good as for the conventional full grid methods, provided that certain additional smoothness requirements are fulfilled. Thus a sparse grid discretization method can be employed also for higher-dimensional problems. The curse of the dimensionality of full grid methods affects sparse grids much less. Note that there exist different variants of solvers working on sparse grids each one with its specific advantages and drawbacks. One variant is based on finite difference discretization [33, 59], another approach uses a Galerkin finite element discretization [4, 10, 13] and the so-called combination technique [40] makes use of multivariate extrapolation [16].

In this paper, we apply the sparse grid combination method to the classification problem. The regularization network problem is discretized and solved on a certain sequence of conventional grids with uniform mesh sizes in each coordinate direction. The sparse grid solution is then obtained from the solutions on these different grids by linear combination. Thus the classifier is build on sparse grid points and not on data points. A discussion of the complexity of the method shows that the new method scales linearly with the number of instances, i.e. the amount of data to be classified. Therefore, our method is well suited for realistic data mining applications where the dimension of the feature space is moderately high after some preprocessing steps but the amount of data is very large. Furthermore the quality of the classifier build by our new method seems to be very good. Here we consider standard test problems from the UCI repository and problems with huge synthetical data sets in up to 9 dimensions. It turns out that our new method achieves correctness rates which are competitive to that of the best existing methods. The combination method is simple to use and can be parallelized in a natural and straightforward way.

The remainder of this paper is organized as follows: In Section 2 we describe the classification problem in the framework of regularization networks as minimization of a (quadratic) functional. We then discretize the feature space and derive the associated linear problem. Here we focus on grid-based discretization techniques. Then, we introduce the concept of a sparse grid and its associated sparse grid space and discuss its properties. Furthermore, we introduce the sparse grid combination technique for the classification problem and present the overall algorithm. We then give implementational details and discuss the complexity of the algorithm. Section 3 presents the results of numerical experiments conducted with the new sparse grid combination method and demonstrates the quality of the classifier build by our new method. Some final remarks conclude the paper.

2 The problem

Classification of data can be interpreted as traditional scattered data approximation problem with certain additional regularization terms. In contrast to conventional scattered data approximation applications, we now encounter quite high-dimensional spaces. To this end, the approach of regularization networks [29] gives a good framework. This approach allows a direct description of the most important neural networks and it also allows for an equivalent description of support vector machines and n-term approximation

schemes [27].
Consider the given set of already classified data (the training set)

$$S = \{(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}\}_{i=1}^M,$$

with $\boldsymbol{x}_i$ representing the data points in the feature space and y_i their class labels. Assume now that these data have been obtained by sampling of an unknown function f which belongs to some function space V defined over $\mathbb{R}^d$. The sampling process was disturbed by noise. The aim is now to recover the function f from the given data as good as possible. This is clearly an ill-posed problem since there are infinitely many solutions possible. To get a well-posed, uniquely solvable problem we have to assume further knowledge of f. To this end, regularization theory [67, 72] imposes an additional smoothness constraint on the solution of the approximation problem and the regularization network approach considers the variational problem

$$\min_{f \in V} R(f)$$

with

$$R(f) = \frac{1}{M} \sum_{i=1}^{M} C(f(\boldsymbol{x}_i), y_i) + \lambda \Phi(f). \tag{1}$$

Here, $C(.,.)$ denotes an error cost function which measures the interpolation error and $\Phi(f)$ is a smoothness functional which must be well defined for $f \in V$. The first term enforces closeness of f to the data, the second term enforces smoothness of f and the regularization parameter λ balances these two terms. Typical examples are

$$C(x, y) = |x - y| \text{ or } C(x, y) = (x - y)^2,$$

and

$$\Phi(f) = ||Pf||_2^2 \quad \text{with} \quad Pf = \nabla f \text{ or } Pf = \Delta f.$$

The value of λ can be chosen according to cross-validation techniques [2, 30, 68, 71] or according to some other principle, such as structural risk minimization [69]. We find exactly this type of formulation in the case $d = 2, 3$ in many scattered data approximation methods, see [1, 44], where the regularization term is usually physically motivated.
Now assume that we have a basis of V given by $\{\varphi_j(\boldsymbol{x})\}_{j=1}^{\infty}$. Let also the constant function be in the span of the functions φ_j. We then can express a function $f \in V$ as

$$f(\boldsymbol{x}) = \sum_{j=1}^{\infty} \alpha_j \varphi_j(\boldsymbol{x})$$

with associated degrees of freedom α_j. In the case of a regularization term of the type

$$\Phi(f) = \sum_{j=1}^{\infty} \frac{\alpha_j^2}{\lambda_j}$$

where $\{\lambda_j\}_{j=1}^{\infty}$ is a decreasing positive sequence, it is easy to show [27], Appendix B, that independent of the function C the solution of the variational problem (1) has always the form

$$f(\boldsymbol{x}) = \sum_{j=1}^{M} \alpha_j K(\boldsymbol{x}, \boldsymbol{x}_j).$$

Here K is the symmetric kernel function

$$K(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^{\infty} \lambda_j \varphi_j(\boldsymbol{x}) \varphi_j(\boldsymbol{y})$$

which can be interpreted as the kernel of a Reproducing Kernel Hilbert Space (RKHS), see [3, 27]. In other words, if certain functions $K(\boldsymbol{x}, \boldsymbol{x}_j)$ are used in an approximation scheme which are centered in the location of the data points $\boldsymbol{x}_j$, then the approximation solution is a finite series and involves only M terms. For radially symmetric kernels we end up with radial basis function approximation schemes. But also many other approximation schemes like additive models, hyperbasis functions, ridge approximation models and several types of neural networks can be derived by a specific choice of the regularization operator [29]. For an overview of these different data-centered approximation schemes, see [19, 28, 29] and the references cited therein.
It is noteworthy that the support vector machine approach can be expressed equivalently in form of the minimization problem (1), see [19, 27, 62] for further details. The special choice of the cost function

$$C(x, y) = |y - x|_\varepsilon = \begin{cases} 0 & \text{if } |y - x| < \varepsilon \\ |y - x| - \varepsilon & \text{otherwise} \end{cases}$$

results in a regularization formulation which has been shown [27, 62] to be equivalent to the support vector machine approach. It has the form

$$f(\boldsymbol{x}, \alpha, \alpha^*) = \sum_{i=1}^{M} (\alpha_i^* - \alpha_i) K(\boldsymbol{x}, \boldsymbol{x}_i) + b$$

with constant b where α_i^* and α_i are positive coefficients which solve the following quadratic programming problem:

$$\min_{\alpha, \alpha^*} R(\alpha^*, \alpha) = \varepsilon \sum_{i=1}^{M} (\alpha_i^* + \alpha_i) - \sum_{i=1}^{M} y_i (\alpha_i^* - \alpha_i) + \frac{1}{2} \sum_{i,k=1}^{M} (\alpha_i^* - \alpha_i)(\alpha_k^* - \alpha_k) K(\boldsymbol{x}_i, \boldsymbol{x}_k)$$

subject to the constraints

$$0 \leq \alpha^*, \alpha \leq \frac{1}{2\lambda M} \quad \text{and} \quad \sum_{i=1}^{M} (\alpha_i^* - \alpha_i) = 0,$$

see also [19], Problem 5.2. Due to the nature of this quadratic programming problem, only a number of coefficients $\alpha_i^* - \alpha_i$ will be different from zero. The input data points $\boldsymbol{x}_i$ associated to them are called support vectors. This formulation is the starting point for some improved schemes like the support vector machine based on the $||.||_1$-norm, the smoothed support vector machine [48], the feature selection concave minimization algorithm [9] or the minimal support vector machine [23]. Also certain n-term approximation schemes, de-noising (basis pursuit) have been shown [27] to be equivalent to the regularization formulation.

2.1 Discretization

In the following we follow a slightly different approach: We explicitly restrict the problem to a finite dimensional subspace $V_N \in V$. The function f is then replaced by

$$f_N = \sum_{j=1}^{N} \alpha_j \varphi_j(\boldsymbol{x}). \tag{2}$$

Here the ansatz functions $\{\varphi_j\}_{j=1}^N$ should span V_N and preferably should form a basis for V_N. The coefficients $\{\alpha_j\}_{j=1}^N$ denote the degrees of freedom. Note that the restriction to a suitably chosen finite-dimensional subspace involves some additional regularization (regularization by discretization) which depends on the choice of V_N.
In the remainder of this paper, we restrict ourselves to the choice

$$C(f_N(\boldsymbol{x}_i), y_i) = (f_N(\boldsymbol{x}_i) - y_i)^2$$

and

$$\Phi(f_N) = ||Pf_N||_{L_2}^2 \tag{3}$$

for some given linear operator P. This way we obtain from the minimization problem a feasible linear system. We thus have to minimize

$$R(f_N) = \frac{1}{M}\sum_{i=1}^{M}(f_N(\boldsymbol{x}_i) - y_i)^2 + \lambda \|Pf_N\|_{L_2}^2, \quad f_N \in V_N \tag{4}$$

in the finite dimensional space V_N. We plug (2) into (4) and obtain

$$R(f_N) = \frac{1}{M}\sum_{i=1}^{M}\left(\sum_{j=1}^{N}\alpha_j\varphi_j(\boldsymbol{x}_i) - y_i\right)^2 + \lambda\|P\sum_{j=1}^{N}\alpha_j\varphi_j\|_{L_2}^2 \tag{5}$$

$$= \frac{1}{M}\sum_{i=1}^{M}\left(\sum_{j=1}^{N}\alpha_j\varphi_j(\boldsymbol{x}_i) - y_i\right)^2 + \lambda\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j(P\varphi_i, P\varphi_j)_{L_2}. \tag{6}$$

Differentiation with respect to α_k, $k = 1, \ldots, N$, gives

$$0 = \frac{\partial R(f_N)}{\partial \alpha_k} = \frac{2}{M}\sum_{i=1}^{M}\left(\sum_{j=1}^{N}\alpha_j\varphi_j(\boldsymbol{x}_i) - y_i\right)\cdot\varphi_k(\boldsymbol{x}_i) + 2\lambda\sum_{j=1}^{N}\alpha_j(P\varphi_j, P\varphi_k)_{L_2}. \tag{7}$$

This is equivalent to $(k = 1, \ldots, N)$

$$\lambda\sum_{j=1}^{N}\alpha_j(P\varphi_j, P\varphi_k)_{L_2} + \frac{1}{M}\sum_{j=1}^{N}\alpha_j\sum_{i=1}^{M}\varphi_j(\boldsymbol{x}_i)\cdot\varphi_k(\boldsymbol{x}_i) = \frac{1}{M}\sum_{i=1}^{M}y_i\varphi_k(\boldsymbol{x}_i) \tag{8}$$

and we obtain $(k = 1, \dots, N)$

$$\sum_{j=1}^{N} \alpha_j \left[M\lambda (P\varphi_j, P\varphi_k)_{L_2} + \sum_{i=1}^{M} \varphi_j(\boldsymbol{x}_i) \cdot \varphi_k(\boldsymbol{x}_i) \right] = \sum_{i=1}^{M} y_i \varphi_k(\boldsymbol{x}_i). \tag{9}$$

In matrix notation we end up with the linear system

$$(\lambda C + B \cdot B^T)\alpha = By. \tag{10}$$

Here C is a square $N \times N$ matrix with entries $C_{j,k} = M \cdot (P\varphi_j, P\varphi_k)_{L_2}$, $j, k = 1, \dots N$, and B is a rectangular $N \times M$ matrix with entries $B_{j,i} = \varphi_j(\boldsymbol{x}_i), i = 1, \dots M, j = 1, \dots N$ The vector y contains the data y_i and has length M. The unknown vector α contains the degrees of freedom α_j and has length N.

Depending on the regularization operator we obtain different minimization problems in d-dimensional space. For example if we use the gradient $P = \nabla$ in the regularization expression in (1) we obtain a Poisson problem with an additional term which resembles the interpolation problem. The natural boundary conditions for such a differential equation in e.g. $\Omega = [0, 1]^d$ are Neumann conditions. The discretization (2) gives us then the linear system (10) where C corresponds to a discrete Laplacian. To obtain the classifier f_N we now have to solve this system.

2.2 Grid based discrete approximation

Up to now we have not yet been specific what finite-dimensional subspace V_N and what type of basis functions $\{\varphi_j\}_{j=1}^N$ we want to use. In contrast to conventional data mining approaches which work with ansatz functions associated to data points we now use a certain grid in the attribute space to determine the classifier with the help of basis functions associated to these grid points. This is similar to the numerical treatment of partial differential equations.

For reasons of simplicity, here and in the remainder of this paper, we restrict ourself to the case $\boldsymbol{x}_i \in \Omega = [0, 1]^d$. This situation can always be reached by a proper rescaling of the data space. A conventional finite element discretization would now employ an equidistant grid Ω_n with mesh size $h_n = 2^{-n}$ for each coordinate direction. In the following we always use the gradient $P = \nabla$ in the regularization expression (3). Let $\mathbf{j}$ denote the multi-index $(j_1, ..., j_d) \in \mathbb{N}^d$. A finite element method with piecewise d-linear test- and trial-functions $\phi_{n,\mathbf{j}}(\boldsymbol{x})$ on grid Ω_n now would give the classifier f_N as

$$(f_N(\boldsymbol{x}) =) f_n(\boldsymbol{x}) = \sum_{j_1=0}^{2^n} \dots \sum_{j_d=0}^{2^n} \alpha_{n,\mathbf{j}} \phi_{n,\mathbf{j}}(\boldsymbol{x})$$

and the variational procedure (4) - (9) would result in the discrete system

$$(\lambda C_n + B_n \cdot B_n^T)\alpha_n = B_n y \tag{11}$$

with the discrete $(2^n + 1)^d \times (2^n + 1)^d$ Laplacian

$$(C_n)_{\mathbf{j},\mathbf{k}} = M \cdot (\nabla \phi_{n,\mathbf{j}}, \nabla \phi_{n,\mathbf{k}}),$$

$j_t, k_t = 0, ..., 2^n, t = 1, ..., d$, the $(2^n + 1)^d \times M$-matrix

$$(B_n)_{\mathbf{j},i} = \phi_{n,\mathbf{j}}(\boldsymbol{x}_i),$$

$j_t = 0, ..., 2^n, t = 1, ..., d$, $i = 0, ..., M$, and the unknown vector $(\alpha_n)_{\mathbf{j}}$, $j_t = 0, ..., 2^n, t = 1, ..., d$. Note that f_n lives in the space

$$V_n := \text{span}\{\phi_{n,\mathbf{j}}, j_t = 0, .., 2^n, t = 1, ..., d\}.$$

The discrete problem (11) might in principle be treated by an appropriate solver like the conjugate gradient method, a multigrid method or some other suitable efficient iterative method. However, this direct application of a finite element discretization and an appropriate linear solver for the arising system is clearly not possible for a d-dimensional problem if d is larger than four. The number of grid points would be of the order $O(h_n^{-d}) = O(2^{nd})$ and, in the best case, if the most effective technique like a multigrid method is used, the number of operations is of the same order. Here we encounter the so-called curse of dimensionality: The complexity of the problem grows exponentially with d. At least for $d > 4$ and a reasonable value of n, the arising system can not be stored and solved on even the largest parallel computers today.

2.3 Sparse grid space

However, there is a special discretization technique using so-called sparse grids which allows to cope with the complexity of the problem, at least to some extent. This method has been originally developed for the solution of partial differential equations [4, 10, 40, 75] and is now used successfully also for integral equations [22, 39], interpolation and approximation [5, 37, 47, 60, 66], eigenvalue problems [25] and integration problems [26]. In the information based complexity community it is also known as 'hyperbolic cross points' and the idea can even be traced back to [63]. For a d-dimensional problem, the sparse grid approach employs only $\mathrm{O}(h_n^{-1}(\log(h_n{}^{-1}))^{d-1})$ grid points in the discretization process. It can be shown that an accuracy of $O(h_n^2 \log(h_n^{-1})^{d-1})$ can be achieved pointwise or with respect to the L_2- or L_∞-norm provided that the solution is sufficiently smooth. Thus, in comparison to conventional full grid methods, which need $O(h_n^{-d})$ points for an accuracy of $O(h_n^2)$, the sparse grid method can be employed also for higher-dimensional problems. The curse of the dimensionality of full grid methods affects sparse grids much less.

Now, with the multi-index $\mathbf{l} = (l_1, ..., l_d) \in \mathbb{N}^d$, we consider the family of standard regular grids

$$\{\Omega_{\mathbf{l}}, \mathbf{l} \in \mathbb{N}^d\} \tag{12}$$

on $\bar{\Omega}$ with mesh size $h_{\mathbf{l}} := (h_{l_1}, ..., h_{l_d}) := (2^{-l_1}, ..., 2^{-l_d})$. That is, $\Omega_{\mathbf{l}}$ is equidistant with respect to each coordinate direction, but, in general, has different mesh sizes in the different coordinate directions. The grid points contained in a grid $\Omega_{\mathbf{l}}$ are the points

$$x_{\mathbf{l},\mathbf{j}} \ := \ (x_{l_1,j_1}, ..., x_{l_d,j_d}) \tag{13}$$

with $x_{l_t,j_t} := j_t \cdot h_{l_t} = j_t \cdot 2^{-l_t}, j_t = 0, ..., 2^{l_t}$. On each grid $\Omega_{\mathbf{l}}$ we define the space $V_{\mathbf{l}}$ of piecewise d-linear functions,

$$V_{\mathbf{l}} \ := \ \text{span}\{\phi_{\mathbf{l},\mathbf{j}}, j_t = 0, ..., 2^{l_t}, t = 1, ..., d\}, \tag{14}$$

which is spanned by the usual d-dimensional piecewise d-linear hat functions

$$\phi_{\mathbf{l},\mathbf{j}}(\mathbf{x}) \;:=\; \prod_{t=1}^{d} \phi_{l_t,j_t}(x_t). \tag{15}$$

Here, the 1 D functions $\phi_{l_t,j_t}(x_j)$ with support $[x_{l_t,j_t}-h_{l_t}, x_{l_t,j_t}+h_{l_t}] = [(j_t-1)h_{l_t}, (j_t+1)h_{l_t}]$ (e.g. restricted to $[0,1]$) can be created from a unique 1 D mother function $\phi(x)$,

$$\phi(x) \;:=\; \begin{cases} 1-|x| & \text{if } x \;\in\;]-1,1[\,, \\ 0 & \text{otherwise,} \end{cases} \tag{16}$$

by dilatation and translation, i. e.

$$\phi_{l_t,j_t}(x_t) \;:=\; \phi\left(\frac{x_t - j_t \cdot h_{l_t}}{h_{l_t}}\right). \tag{17}$$

In the previous definitions and in the following, the multi-index $\mathbf{l} \in \mathbb{N}^d$ indicates the level of a grid or a space or a function, respectively, whereas the multi-index $\mathbf{j} \in \mathbb{N}^d$ denotes the location of a given grid point $x_{\mathbf{l},\mathbf{j}}$ or of the respective basis function $\phi_{\mathbf{l},\mathbf{j}}(\mathbf{x})$, respectively. Now, we can define the difference spaces

$$W_{\mathbf{l}} \;:=\; V_{\mathbf{l}} - \sum_{t=1}^{d} V_{\mathbf{l}-\mathbf{e}_t}, \tag{18}$$

where $\mathbf{e}_t$ denotes the t-th unit vector. To complete this definition, we formally set $V_{\mathbf{l}} := \emptyset$ if $l_t = -1$ for at least one $t \in \{1, ..., d\}$. These hierarchical difference spaces allow us the definition of a multilevel subspace splitting, i. e. the definition of the space V_n as a direct sum of subspaces,

$$V_n \;:=\; \sum_{l_1=0}^{n} \cdots \sum_{l_d=0}^{n} W_{(l_1,..l_d)} \;=\; \bigoplus_{|\mathbf{l}|_\infty \le n} W_{\mathbf{l}}, \tag{19}$$

Here and in the following, let $\leq$ denote the corresponding element-wise relation, and let $|\mathbf{l}|_\infty := \max_{1\le l_t \le d} l_t$ and $|\mathbf{l}|_1 := \sum_{t=1}^{d} l_t$ denote the discrete L_∞- and the discrete L_1-norm of $\mathbf{l}$, respectively. As it can be easily seen from (14) and (18), the introduction of index sets $\mathbf{I}_{\mathbf{l}}$,

$$\mathbf{I}_{\mathbf{l}} \;:=\; \left\{ (j_1, ..., j_d) \in \mathbb{N}^d, \begin{cases} j_t = 1, ..., 2^{l_t}-1, j_t \text{ odd}, t = 1, ..., d, & \text{if } l_t > 0, \\ j_t = 0, 1, t = 1, ..., d, & \text{if } l_t = 0, \end{cases} \right\}, \tag{20}$$

leads to

$$W_{\mathbf{l}} \;=\; \text{span}\{\phi_{\mathbf{l},\mathbf{j}}, \mathbf{j} \in \mathbf{I}_{\mathbf{l}}\}. \tag{21}$$

Therefore, the family of functions

$$\{\phi_{\mathbf{l},\mathbf{j}}, \mathbf{j} \in \mathbf{I}_{\mathbf{l}}\}_{\mathbf{0}}^{\mathbf{n}} \tag{22}$$

is just a hierarchical basis [20, 73, 74] of V_n that generalizes the one-dimensional hierarchical basis of [20] to the d-dimensional case by means of a tensor product approach. Note that the supports of all basis functions $\phi_{\mathbf{l},\mathbf{j}}(\mathbf{x})$ in (21) spanning $W_{\mathbf{l}}$ are mutually disjoint.

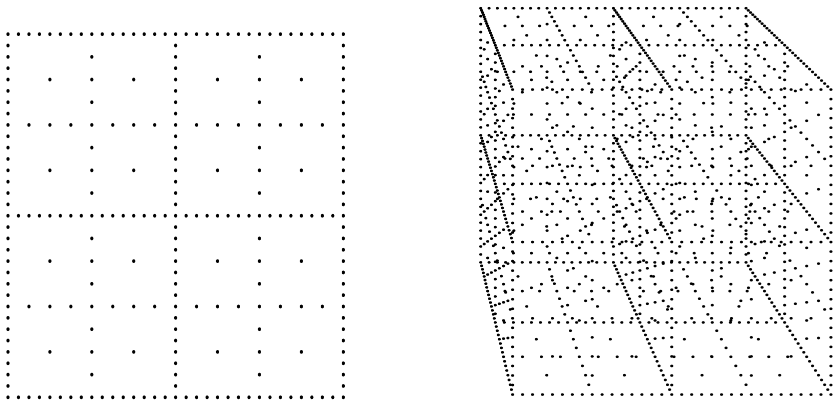

Figure 1: Two-dimensional sparse grid (left) and three-dimensional sparse grid (right), $n = 5$

Now, any function $f \in V_n$ can be splitted accordingly by

$$f(\mathbf{x}) \;=\; \sum_{\mathbf{l} \leq \mathbf{n}} f_{\mathbf{l}}(\mathbf{x}), \quad f_{\mathbf{l}} \in W_{\mathbf{l}}, \quad \text{and } f_{\mathbf{l}}(\mathbf{x}) \;=\; \sum_{\mathbf{j} \in \mathbf{I_l}} \alpha_{\mathbf{l},\mathbf{j}} \cdot \phi_{\mathbf{l},\mathbf{j}}(\mathbf{x}), \tag{23}$$

where $\alpha_{\mathbf{l},\mathbf{j}} \in \mathbb{R}$ are the coefficient values of the hierarchical product basis representation. It is the hierarchical representation which now allows to consider the following subspace $V_n^{(s)}$ of V_n which is obtained by replacing $|\mathbf{l}|_\infty$ by $|\mathbf{l}|_1$ in (19):

$$V_n^{(s)} \;:=\; \bigoplus_{|\mathbf{l}|_1 \leq n+d-1} W_{\mathbf{l}}. \tag{24}$$

Again, any function $f \in V_n^{(s)}$ can be splitted accordingly by

$$f_n^{(s)}(\mathbf{x}) \;=\; \sum_{|\mathbf{l}|_1 \leq n+d-1} \sum_{\mathbf{j} \in \mathbf{I_l}} \alpha_{\mathbf{l},\mathbf{j}} \phi_{\mathbf{l},\mathbf{j}}(\mathbf{x}). \tag{25}$$

The grids corresponding to the approximation spaces $V_n^{(s)}$ are called sparse grids and have been studied in detail in [10, 11, 12, 13, 31, 34, 40, 42, 75]. Examples of sparse grids for the two- and three-dimensional case are given in Figure 1.

Now, a straightforward calculation [14] shows that the dimension of the sparse grid space $V_n^{(s)}$ is of the order $O(n^{d-1}2^n)$. For the interpolation problem, as well as for the approximation problem stemming from second order elliptic PDEs, it was proven that the sparse grid solution $f_n^{(s)}$ is almost as accurate as the full grid function f_n, i.e. the discretization error satisfies

$$||f - f_n^{(s)}||_{L_p} = O(h_n^2 \log(h_n^{-1})^{d-1})$$

provided that a slightly stronger smoothness requirement on f holds than for the full grid approach. Here, we need the seminorm

$$|f|_\infty := \left|\left| \frac{\partial^{2d} f}{\prod_{t=1}^{d} \partial x_t^2} \right|\right|_\infty \tag{26}$$

to be bounded.
The idea is now to carry this discretization method and its advantages with respect to the degrees of freedom over to the minimization problem (1). The minimization procedure (4) - (9) with the discrete function $f_n^{(s)}$ in $V_n^{(s)}$

$$f_n^{(s)} = \sum_{|\mathbf{l}|_1 \le n+d-1} \sum_{\mathbf{I_l}} \alpha_{\mathbf{l},\mathbf{j}}^{(s)} \phi_{\mathbf{l},\mathbf{j}}(\boldsymbol{x})$$

would result in the discrete system

$$(\lambda C_n^{(s)} + B_n^{(s)} \cdot (B_n^{(s)})^T)\alpha_n^{(s)} = B_n^{(s)} y \tag{27}$$

with

$$(C_n^{(s)})_{(\mathbf{l},\mathbf{j}),(\mathbf{r},\mathbf{k})} = M \cdot (\nabla\phi_{n,(\mathbf{l},\mathbf{j})}, \nabla\phi_{n,(\mathbf{r},\mathbf{k})}), \quad (B_n^{(s)})_{(\mathbf{l},\mathbf{j}),i} = \phi_{n,(\mathbf{l},\mathbf{j})}(\boldsymbol{x}_i),$$

$|\mathbf{l}|_1 \le n+d-1, |\mathbf{r}|_1 \le n+d-1, \mathbf{j} \in \mathbf{I_l}, \mathbf{r} \in \mathbf{I_k}, i = 1, ..., M$, and the unknown vector $(\alpha_n^{(s)})_{(\mathbf{r},\mathbf{k})}$, $|\mathbf{r}|_1 \le n+d-1, \mathbf{r} \in \mathbf{I_k}$. The discrete problem (27) might in principle be treated by an appropriate iterative solver. Note that now the size of the problem is just of the order $O(2^n n^{d-1})$. Here, the explicit assembly of the matrix $C_n^{(s)}$ and $B_n^{(s)}$ should be avoided. These matrices are more densely populated than the corresponding full grid matrices and this would add further terms to the complexity. Instead only the action of these matrices onto vectors, i.e. a matrix-vector multiplication should be performed in an iterative method like the conjugate gradient method or a multigrid method. For example, for $C_n^{(s)}$ this is possible in a number of operations which is proportional to the unknowns only. For details, see [4, 10]. However the implementation of such a program is quite cumbersome and difficult. It also should be possible to avoid the assembly of $B_n^{(s)}$ and $(B_n^{(s)})^T$ and to program the respective matrix-vector multiplications in $O(2^n n^{d-1}, M)$ operations. But this is complicated as well.
There also exists another variant of a solver working on the sparse grid, the so called combination technique [40], which makes use of multivariate extrapolation [16]. In the following, we apply this method to the minimization problem (1). It is much simpler to use than the Galerkin-approach (27), it avoids the matrix assembly problem mentioned above and it can be parallelized in a natural and straightforward way, see [32, 36].
Note that an approach similar to ours was introduced recently in [43].

2.4 The sparse grid combination technique

For the sparse grid combination technique we proceed as follows: We discretize and solve the problem on a certain sequence of grids $\Omega_\mathbf{l}$ with uniform mesh sizes $h_t = 2^{-l_t}$ in the t-th coordinate direction. These grids may possess different mesh sizes for different coordinate directions. To this end, we consider all grids $\Omega_\mathbf{l}$ with

$$l_1 + ... + l_d = n + (d-1) - q, \quad q = 0, .., d-1, \quad l_t > 0. \tag{28}$$

In contrast to the definition (24), for reasons of efficiency, we now restrict the level indices to $l_t > 0$. The finite element approach with piecewise d-linear test- and trial-functions

$\phi_{\mathbf{l},\mathbf{j}}(\boldsymbol{x})$ on grid $\Omega_{\mathbf{l}}$ now would give

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

and the variational procedure (4) - (9) would result in the discrete system

$$(\lambda C_{\mathbf{l}} + B_{\mathbf{l}} \cdot B_{\mathbf{l}}^T)\alpha_{\mathbf{l}} = B_{\mathbf{l}} y \tag{29}$$

with the matrices

$$(C_{\mathbf{l}})_{\mathbf{j},\mathbf{k}} = M \cdot (\nabla \phi_{\mathbf{l},\mathbf{j}}, \nabla \phi_{\mathbf{l},\mathbf{k}}) \text{ and } \quad (B_{\mathbf{l}})_{\mathbf{j},i} = \phi_{\mathbf{l},\mathbf{j}}(\boldsymbol{x}_i),$$

$j_t, k_t = 0, ..., 2^{l_t}, t = 1, ..., d, i = 1, ..., M$, and the unknown vector $(\alpha_{\mathbf{l}})_{\mathbf{j}}$, $j_t = 0, ..., 2^{l_t}, t = 1, ..., d$. We then solve these problems by a feasible method. To this end we use here a diagonally preconditioned conjugate gradient algorithm. But also an appropriate multi-grid method with partial semi-coarsening can be applied. The discrete solutions $f_{\mathbf{l}}$ are contained in the spaces $V_{\mathbf{l}}$, see (14), of piecewise d-linear functions on grid $\Omega_{\mathbf{l}}$.

Note that all these problems are substantially reduced in size in comparison to (11). Instead of one problem with size $\dim(V_n) = O(h_n^{-d}) = O(2^{nd})$, we now have to deal with $O(dn^{d-1})$ problems of size $\dim(V_{\mathbf{l}}) = O(h_n^{-1}) = O(2^n)$. Moreover, all these problems can be solved independently which allows for a straightforward parallelization on a coarse grain level, see [32]. Also there is a simple but effective static load balancing strategy available [36].

Finally we linearly combine the results $f_{\mathbf{l}}(\boldsymbol{x}) = \sum_{\mathbf{j}} \alpha_{\mathbf{l},\mathbf{j}} \phi_{\mathbf{l},\mathbf{j}}(\boldsymbol{x}) \in V_{\mathbf{l}}$ from the different grids $\Omega_{\mathbf{l}}$ as follows:

$$f_n^{(c)}(\boldsymbol{x}) := \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{l_1+\dots+l_d=n+(d-1)-q} f_{\mathbf{l}}(\boldsymbol{x}). \tag{30}$$

The resulting function $f_n^{(c)}$ lives in the above-defined sparse grid space $V_n^{(s)}$(but now with $l_t > 0$ in (24)). For the two-dimensional case, the grids needed in the combination formula of level 4 are shown in Figure 2.

The combination technique can be interpreted as a certain multivariate extrapolation method which works on a sparse grid, for details see [16, 40, 56]. The combination solution $f_n^{(c)}$ is in general not equal to the Galerkin solution $f_n^{(s)}$ but its accuracy is usually of the same order, see [40]. To this end, a series expansion of the error is necessary. Its existence was shown for PDE-model problems in [15].

Note that the summation of the discrete functions from different spaces $V_{\mathbf{l}}$ in (30) involves d-linear interpolation which resembles just the transformation to a representation in the hierarchical basis (22). For details see [33, 41]. However we never explicitly assemble the function $f_n^{(c)}$ but keep instead the solutions $f_{\mathbf{l}}$ on the different grids $\Omega_{\mathbf{l}}$ which arise in the combination formula. Now, any linear operation F on $f_n^{(c)}$ can easily be expressed by means of the combination formula (30) acting directly on the functions $f_{\mathbf{l}}$, i.e.

$$\mathrm{F}(f_n^{(c)}) = \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{l_1+\dots+l_d=n+(d-1)-q} \mathrm{F}(f_{\mathbf{l}}). \tag{31}$$

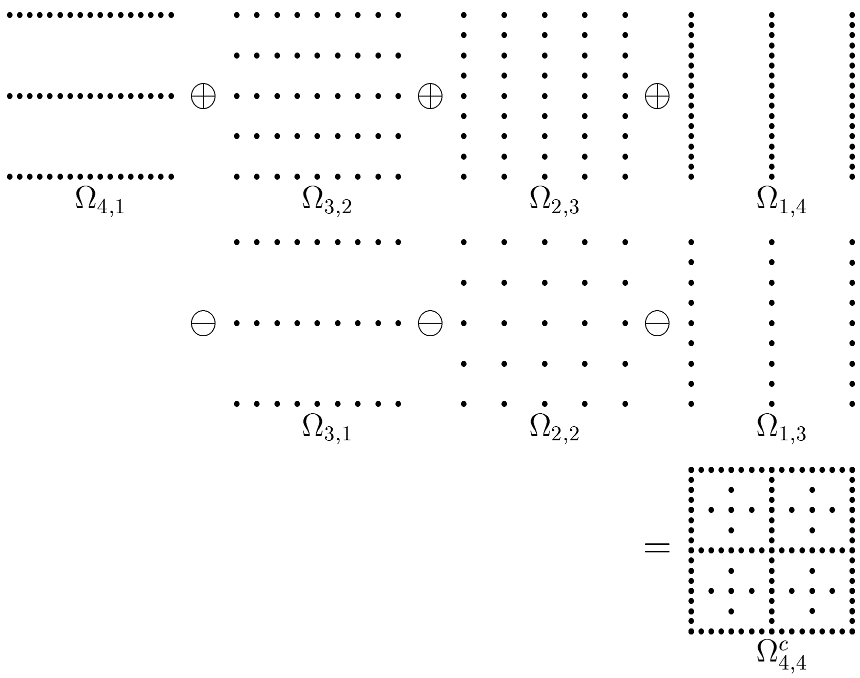

Figure 2: Combination technique for level 4 in two dimensions

Therefore, if we now want to evaluate a newly given set of data points $\{\tilde{\boldsymbol{x}}_i\}_{i=1}^{\tilde{M}}$ (the test set) by

$$\tilde{y}_i := f_n^{(c)}(\tilde{\boldsymbol{x}}_i), \quad i = 1, ..., \tilde{M}$$

we just form the combination of the associated values for $f_{\mathbf{l}}$ according to (30). Altogether we obtain the algorithm of Figure 3.
The combination technique is only one of the various methods to solve problems on sparse grids. Note that there exist also finite difference [33, 59] and Galerkin finite element approaches [4, 10, 13] which work directly in the hierarchical product basis on the sparse grid. But the combination technique is conceptually much simpler and easier to implement. Moreover it allows to reuse standard solvers for its different subproblems and is straightforwardly parallelizable.

2.5 Implementational details and complexity discussion

Up to now we did not adress the question how to assemble and treat the linear systems (29)

$$(\lambda C_{\mathbf{l}} + B_{\mathbf{l}} \cdot B_{\mathbf{l}}^T)\alpha_{\mathbf{l}} = B_{\mathbf{l}} y \ (=: r_{\mathbf{l}}), \tag{32}$$

$|\mathbf{l}|_1 = n + (d-1) - q, \quad q = 0, .., d-1$, which arise in the combination method. Here, different approaches are possible: First we can explicitly assemble and store the matrices

$$C_{\mathbf{l}} \quad \text{and} \quad G_{\mathbf{l}} := B_{\mathbf{l}} \cdot B_{\mathbf{l}}^T.$$

Second, we can assemble and store $C_{\mathbf{l}}$ and $B_{\mathbf{l}}$. Of course, all these matrices are sparsely populated. We only assemble the non-zero entries. The resulting complexities are given

Given are data points $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^M$ (the training set)
Computation of the sparse grid classsificator:
$q = 0 \;\; to \;\; d-1$
  $l_1 = 1 \;\; to \;\; n-q$
    $l_2 = 1 \;\; to \;\; n-q-(l_1-1)$
      $l_3 = 1 \;\; to \;\; n-q-(l_1-1)-(l_2-1)$
        .....
          $l_{d-1} = 1 \;\; to \;\; n-q-(l_1-1)-...-(l_{d-2}-1)$
            $l_d = n-q-(l_1-1)-(l_2-1)-...-(l_{d-2}-1)-(l_{d-1}-1)$
              solve the linear equation system $(\lambda C_{\mathbf{l}} + B_{\mathbf{l}} \cdot B_{\mathbf{l}}^T)\alpha_{\mathbf{l}} = B_{\mathbf{l}} y$
Evaluation in newly given data points $\{\tilde{\boldsymbol{x}}_i\}_{i=1}^{\tilde{M}}$ (the test set):
$\tilde{y}_i = 0, i = 1, ..., \tilde{M}$
$q = 0 \;\; to \;\; d-1$
  $l_1 = 1 \;\; to \;\; n-q$
    $l_2 = 1 \;\; to \;\; n-q-(l_1-1)$
      $l_3 = 1 \;\; to \;\; n-q-(l_1-1)-(l_2-1)$
        .....
          $l_{d-1} = 1 \;\; to \;\; n-q-(l_1-1)-...-(l_{d-2}-1)$
            $l_d = n-q-(l_1-1)-(l_2-1)-...-(l_{d-2}-1)-(l_{d-1}-1)$
              evaluate the solution $f_{\mathbf{l}}$ in $\tilde{\boldsymbol{x}}_i$ by d-linear interpolation on grid $\Omega_{\mathbf{l}}$
              set $\tilde{y}_i := \tilde{y}_i + (-1)^q \binom{d-1}{q} f_{\mathbf{l}}(\tilde{\boldsymbol{x}}_i), \quad i = 1, ..., \tilde{M}$

Figure 3: The overall algorithm

in Table 1. Here, $N = \prod_{t=1}^d (2^{l_t} + 1)$ denotes the number of unknowns on grid $\Omega_{\mathbf{l}}$. Note

	$C_{\mathbf{l}}$	$G_{\mathbf{l}}$	$B_{\mathbf{l}}$	$r_{\mathbf{l}}$	$\alpha_{\mathbf{l}}$
storage	$O(3^d \cdot N)$	$O(3^d \cdot N)$	$O(2^d \cdot M)$	$O(N)$	$O(N)$
assembly	$O(3^d \cdot N)$	$O(d \cdot 2^{2d} \cdot M)$	$O(d \cdot 2^d \cdot M)$	$O(d \cdot 2^d \cdot M)$	-
mv-multiplication	$O(3^d \cdot N)$	$O(3^d \cdot N)$	$O(2^d \cdot M)$	-	-

Table 1: Complexities for the storage, the assembly and the matrix-vector multiplication for different matrices and vectors arising in the combination method on one grid $\Omega_{\mathbf{l}}$

that $G_{\mathbf{l}}$ is in principle a $N \times N$ matrix. But of course, we take into account the non-zero entries only.
Interestingly there is a difference in the complexities of the two approaches: The storage cost and the cost of the matrix-vector multiplication for the first approach scale with N, whereas they scale with M for the second approach. Especially for the case $M >> N$, i.e. for a moderate number of levels n in the sparse grid but a very large amount of data to be handled for the training phase in the classification problem, there is a strong difference in the resulting storage cost and run time.

Since we have $O(dn^{d-1})$ different problems to store we obtain an overall storage complexity of $O(dn^{d-1} \cdot 3^d N)$ for the first approach and $O(dn^{d-1} \cdot 2^d \cdot M)$ for the second approach. Clearly, for n and thus N fixed, and large amount of data, i.e. $M >> N$ the storage of the matrices $G_{\mathbf{l}}$ is more advantageous than the storage of the matrices B_l. With respect to the assembly cost ($O(dn^{d-1} \cdot (3^d N + d2^{2d} M))$ versus $O(dn^{d-1} \cdot (3^d N + d2^d M))$) there is not such a difference. However when it finally comes to the overall cost of the matrix-vector multiplications for all $|\mathbf{l}|_1 = n + d - q, q = 0, ..., d-1$, then we obtain only a complexity of the order $O((3^d + 3^d) \cdot N \cdot dn^{d-1})$ for the first approach, but a complexity of the order $O((3^d N + 2^d M) dn^{d-1})$ for the second method. Thus, analogously to the storage complexities, also the matrix-vector multiplication for the first approach behaves advantageous in the practically relevant case $M >> N$ of large data sets.
Note however that both the storage and the run time complexities depend exponentially on the dimension d. Presently, due to the limitations of the memory of modern workstations (512 MByte - 2 GByte), we therefore can only deal with the case $d \leq 8$ if we assemble the matrices and keep them in the computer memory. Alternatively we can avoid to build these matrices explicitly but compute the necessary entries on the fly when needed in the CG-iteration. This way we trade main memory storage versus computing time. Then, depending on the chosen value of n, problems up to $d = 15$ fit into the memory of modern workstations. Now, since the cost of the assembly of the matrices arises in every matrix-vector multiplication arising in the CG-iteration, there is no more an advantage for the first approach. On the contrary, due to the difference in the d-dependency (2^{2d} versus 2^d), the second approach performs better.
Note that the solution of the various problems arising in the combination technique by an iterative method like conjugate gradients is surely not optimal. Here, the number of iterations necessary to obtain a prescribed accuracy depends on the condition numbers of the respective system matrices. Here it is possible to use an appropriate multigrid method to iteratively treat the different linear systems with a convergence rate which is independent of n. Also robustness, i.e. a convergence rate independent of the discrete anisotropy caused by the possible difference in the mesh sizes in different coordinate directions can be achieved by applying semi-coarsening techniques where necessary [38]. Furthermore independence of the convergence rate from the regularization parameter is another issue. Here, since $B_{\mathbf{l}} B_{\mathbf{l}}^T$ resembles a somewhat smeared out local mixture between the identity and the zero matrix, there is hope to obtain an overall robust method along the lines of [53, 54, 65]. All these techniques are well understood in the two- and three-dimensional case. They should work analogously in the higher-dimensional case as well. However, there is no implementation of a high-dimensional robust multigrid method yet. This involves quite some work and future effort. Therefore we stick to the diagonally preconditioned conjugate gradient method for the time being.
Finally let us consider the evaluation of the computed sparse grid combination classifier for a newly given set of points $\{\tilde{\boldsymbol{x}}_i\}_{i=1}^{\tilde{M}}$. From (28) we directly see that this involves $O(dn^{d-1}\tilde{M})$ operations. Presently it is not clear, if this complexity can be reduced somewhat by switching to the explicit assembly of the combination solution $f_n^{(c)}$ in terms of the hierarchical basis (cost $O(2^n n^{d-1})$) and evaluating the $\tilde{M}$ points $\tilde{\boldsymbol{x}}_i$ in the hierarchical basis representation. There is a slight chance that certain common intermediate results might be exploited in a way analogously to other multiscale calculations on sparse grids,

compare [4, 10], but this is unclear up to now. In any case, the evaluation of the different f_1 in the test points can be done completely in parallel, their summation according to (30) needs basically an all-reduce/gather operation.
Note finally that efficient data structures for sparse matrices are crucial to allow for an efficient matrix-vector-multiplication. Since we are dealing with higher dimensional problems we have to use data structures which are still fast for rows with many, i.e. 3^d entries. In the case of the on-the-fly matrix-vector-multiplication it is also very important to optimize this part of the algorithm in a similar way since it consumes the main part of the computing time.

3 Numerical results

We now apply our approach to different test data sets. Here we use both synthetical data generated by DatGen [51] and real data from practical data mining applications. All the data sets are rescaled to $[0,1]^d$. To evaluate our method we give the correctness rates on testing data sets if available and the ten-fold cross-validation results on training and testing data sets. For the ten-fold cross validation we proceed as follows: We divide the training data into 10 equally sized disjoint subsets. For $i = 1$ to 10, we pick the i-th of these subsets as further testing set and build with the data from the remaining 9 subsets the sparse grid combination classifier. We then evaluate the correctness rates of the current training and testing set. This way we obtain ten different training and testing correctness rates. The ten-fold cross validation result is then just the average of these ten correctness rates. For further details see [64]. For a critical discussion on the evaluation of the quality of classifier algorithms see [58].

3.1 Two-dimensional problems

We first consider two-dimensional problems with small sets of data which correspond to certain structures. Then we treat problems with huge sets of synthetic data with up to 5 million points. In the following Figures we show the border between the two classes, i.e. a class label of zero, with the black line, values between 0 and 1 are presented bright and values between 0 and -1 are shown from bright to dark to show the change in the classifier.

3.1.1 Checkerboard

The first example is taken from [45, 46]. Here, 1000 training data points were given which are more or less uniformly distributed in $\Omega = [0,1]^2$. The associated class labels are plus one or minus one depending on their location in Ω such that a 4×4 checkerboard structure appears, see Figure 4 (left). We computed the 10-fold cross-validated training and testing correctness with the sparse grid combination method for different values of the regularization parameter λ and different levels n. The results are shown in Figure 4 (right).
We see that the 10-fold testing correctness is well around 95 % for values of λ between $3 \cdot 10^{-5}$ and $5 \cdot 10^{-3}$. Our best 10-fold-testing correctness was 96.20% on level 10 with

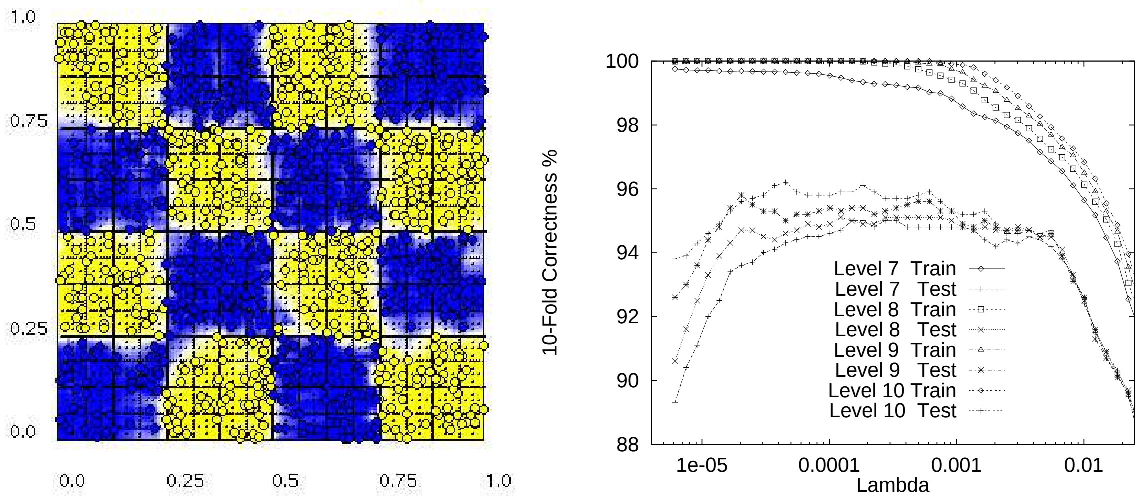

Figure 4: Left: Checkerboard data set [45], combination technique with level 10, $\lambda = 4.53999 \cdot 10^{-5}$, Right: plot of the dependence on λ (in logscale) and level

$\lambda = 4.54 \cdot 10^{-5}$. The checkerboard structure is thus reconstructed with less than 4 % error.

3.1.2 Spiral

Another two-dimensional example with structure is the spiral data set, first proposed by Alexis Wieland of MITRE Corp, see also [21]. Here, 194 data points describe two intertwined spirals, see Figure 5. This is surely an artificial problem which does not appear in many practical applications. However it serves as a hard test case for data mining algorithms. It is known that neural networks can have severe problems with this data set and some neural networks can not separate the two spirals at all [61]. In Table 2 we give the correctness rates achieved with the leave-one-out cross-validation method, i.e. a 194-fold cross-validation. The best testing correctness was achieved on level 6 with 89.69% in comparison to 77.20% in [61].

level	λ	training correctness	testing correctness
4	0.00001	95.31 %	87.63 %
5	0.001	94.36 %	87.11 %
6	0.00075	100.00 %	89.69 %
7	0.00075	100.00 %	88.14 %
8	0.0005	100.00 %	87.63 %

Table 2: Leave-one-out cross-validation results for the spiral data set

In Figure 5 we show the corresponding results obtained with our sparse grid combination method for the levels 4 to 7. Already for level 6 the two spirals are clearly detected and resolved. Note that here only 447 grid points are contained in the sparse grid. For level

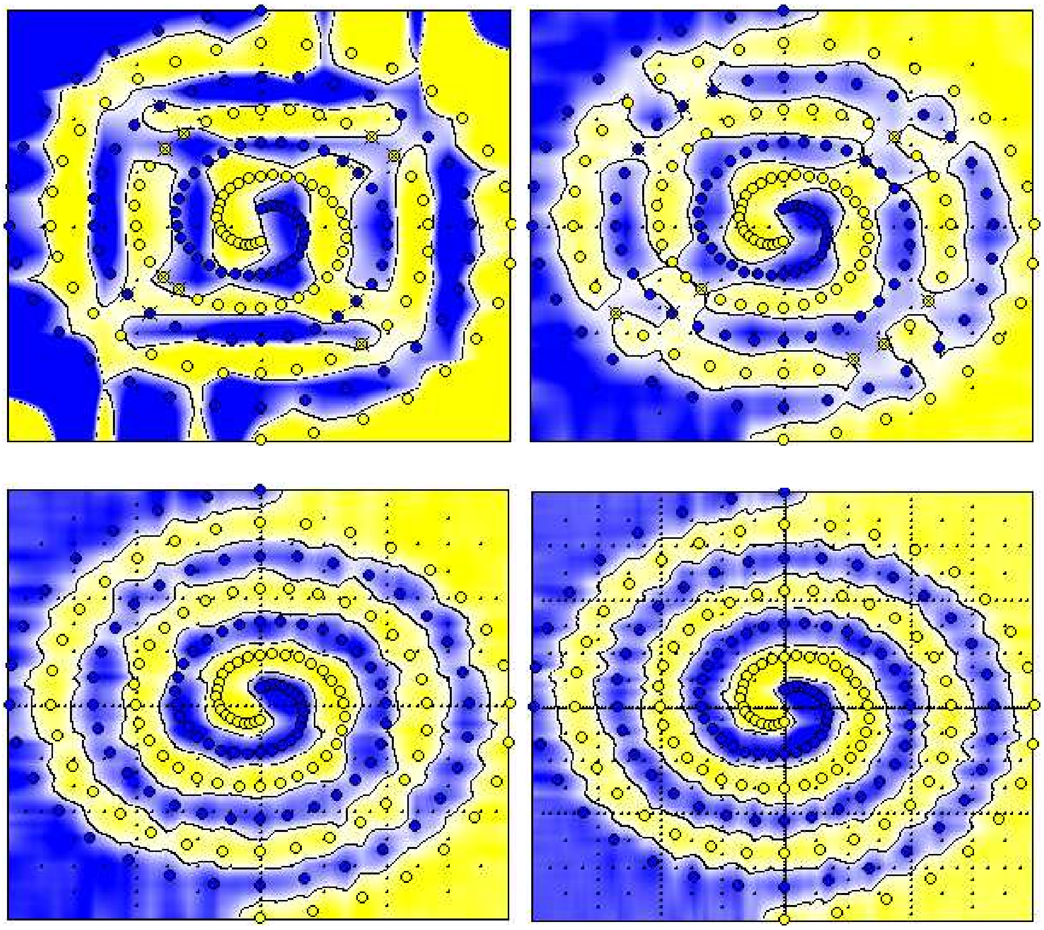

Figure 5: Spiral data set, sparse grid with level 4 (top left) to 7 (bottom right)

7 (1023 sparse grid points) the shape of the two reconstructed spirals gets smoother and the reconstruction gets more precise.

3.1.3 Ripley

This data set, taken from [57], consists of 250 training data and 1000 test points. The data set was generated synthetically and is known to exhibit 8 % error. Thus no better testing correctness than 92 % can be expected.
Since we now have training and testing data, we proceed as follows: First we use ten-fold cross-validation on the training set to determine the best regularization parameter λ. This value is given for different levels n in the first column in Table 3. With this λ we then compute the sparse grid classifier from the 250 training data. The columns two and three of Table 3 give the performance of this classifier on the test data set. Here, we show the 10-fold cross-validation testing correctness using the training data set and the testing correctness using the (previously unknown) test data. We see that our method works well. Already level 5 is sufficient to obtain results of 90.9 %. We also see that there is not much need to use any higher levels. The reason is surely the relative simplicity of the data, see Figure 6. Just a few hyperplanes should be enough to separate the classes quite

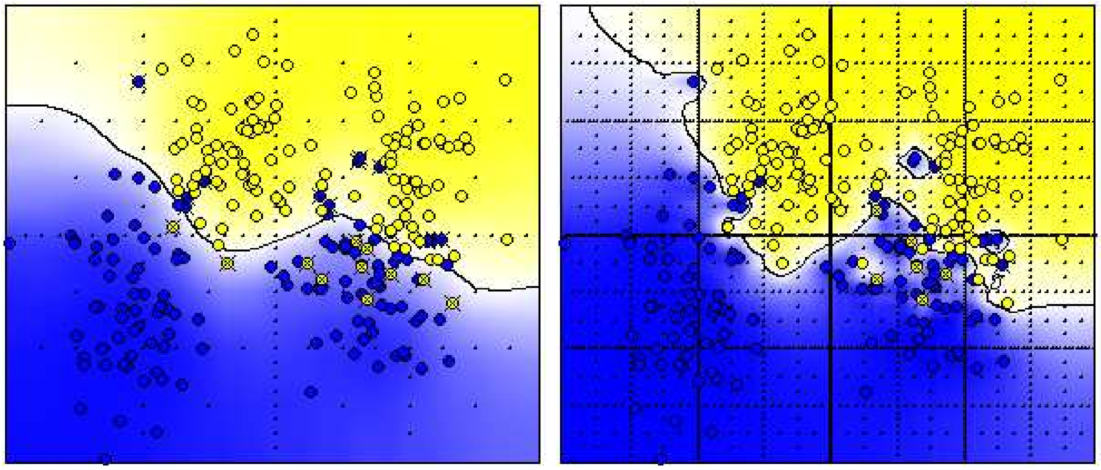

Figure 6: Ripley data set, combination technique with level 5, $\lambda = 0.01005$ (left) level 8, $\lambda = 0.00166$ (right)

properly. This is achieved with the sparse grid already for a small number n. Additionally

	ten-fold			best	
level	ten-fold testing	λ	on test data	λ	testing
1	84.8 %	0.01005	89.8 %	0.00370	90.3 %
2	85.2 %	$9.166 \cdot 10^{-6}$	90.4 %	0.00041	90.9 %
3	88.4 %	0.00166	90.6 %	0.00370	91.2 %
4	87.6 %	0.00248	90.6 %	0.01500	91.2 %
5	87.6 %	0.01005	90.9 %	0.00673	91.1 %
6	86.4 %	0.00673	90.8 %	0.00673	90.8 %
7	86.4 %	0.00075	88.5 %	0.00673	91.0 %
8	88.0 %	0.00166	89.7 %	0.00673	91.0 %
9	88.4 %	0.00203	90.9 %	0.00823	91.0 %
10	88.4 %	0.00166	90.6 %	0.00452	91.1 %
[57]					90.6
[55]					91.1

Table 3: Results for the Ripley data set [57]

we give in Table 3 the testing correctness which is achieved for the best possible λ. To this end we compute for *all* (discrete) values of λ the sparse grid classifiers from the 250 data points and evaluate them on the test set. We then pick the best result. We clearly see that there is not much of a difference. This indicates that our approach to determine the value of λ from the training set by cross-validation works well.

3.1.4 Synthetic huge data set in 2D

To test our method with very large data sets we produced with DatGen [51] (datgen -r2 -X0/100,R,O:0/100,R,O -R2 -C2/4 -D0/1 -T10/70 -O2000000 -e0.15 -p) a data set with 5 million training points and 20 000 points for testing. The results obtained with our sparse grid combination technique are given in Table 4. Here we consider different levels n and different amounts of data points and use the value 0.01 for the regularization parameter λ. We also show the actual total run time of the combination technique on one processor (MIPS R10000 (195 MHz)) of an SGI O200. Then, the sum over the overall number of iterations is given, which was needed for the CG-treatment of the different problems arising in the combination method. Here the accuracy in the termination criterion of the CG-solver was set to 10^{-10}. We see that our method gives testing correctness results up to 95 %. Furthermore we see that really large data sets can be handled without any problems. Here, the execution time scales linearly with the number of data as predicted by the complexity analysis of the previous section. Finally we see that the run times are reasonable also for huge data sets.

	# of points	training correctness	testing correctness	time (sec)	# of iterations
level 1	100 000	94.7 %	94.7 %	2.9	9
	1 million	94.7 %	94.7 %	26	9
	2 million	94.7 %	94.7 %	54	9
	5 million	94.7 %	94.7 %	132	9
level 2	100 000	94.9 %	94.9 %	5.3	40
	1 million	94.9 %	94.9 %	50	41
	2 million	94.9 %	94.9 %	103	41
	5 million	94.9 %	94.9 %	253	41
level 3	100 000	94.9 %	94.9 %	7.8	113
	1 million	94.9 %	94.9 %	75	115
	2 million	94.9 %	94.8 %	150	117
	5 million	94.9 %	94.9 %	373	117
level 4	100 000	95.0 %	94.9 %	10	277
	1 million	95.0 %	94.9 %	99	283
	2 million	95.0 %	94.9 %	199	285
	5 million	95.0 %	94.9 %	496	287
level 5	100 000	95.0 %	95.0 %	12	650
	1 million	95.0 %	94.9 %	124	666
	2 million	95.0 %	94.9 %	247	669
	5 million	95.0 %	95.0 %	620	676

Table 4: Results for a 2D huge synthetic data set, $\lambda = 0.01$

3.2 6-dimensional problems

3.2.1 BUPA Liver

The BUPA Liver Disorders data set from Irvine Machine Learning Database Repository [8] consists of 345 data points with 6 features plus a selector field used to split the data into 2 sets with 145 instances and 200 instances respectively. Here we only have training data and therefore can only report our ten-fold cross-validation results. No comparison with unused test data is possible.
We compare with the two best results from [48], the therein introduced smoothed support vector machine (SSVM) and the feature selection concave minimization (FSV) algorithm [9]. The results are given in Table 5.

			sparse grid combination method			
	SSVM	FSV	Level 1	Level 2	Level 3	Level 4
	[48]	[48]	$\lambda = 0.02$	$\lambda = 0.1$	$\lambda = 0.007$	$\lambda = 0.0004$
10-fold train. corr., %	70.37	68.18	76.00	77.49	84.28	90.27
10-fold test. corr., %	70.33	65.20	67.87	67.84	70.34	70.92

Table 5: Results for the BUPA liver disorders data set

Our sparse grid combination approach performs on level 4 with $\lambda = 0.0004$ at 70.92 % 10-fold testing correctness. But also our other results were in this range. Our method performs here slightly better than the SSVM but clearly better than FSV. Note that the results for the robust linear program (RLP) algorithm [6], the support vector machine using the 1-norm approach ($\mathrm{SVM}_{||.||_1}$) and the classical support vector machine ($\mathrm{SVM}_{||.||_2^2}$) [9, 17, 70] were reported to be somewhat worse in [48].

3.2.2 Synthetic massive data set in 6D

Now we produced with DatGen [51] a 6-dimensional data set with 5 million training points and 20 000 points for testing. We used the call datgen -r1 -X0/100,R,O:0/100,R,O:0/100, R,O:0/100,R,O:0/100,R,O:0/200,R,O:0/200,R,O -R2 -C2/4 -D2/5 -T10/60 -O5020000 -p -e0.15.
The results are given in Table 6. Already on level 1 a testing correctness of 88 % was achieved which is quite satisfying for this data. We see that really huge data sets of 5 million points could be handled. We also give the CPU time which is needed for the computation of the matrices $G_{\mathbf{l}} = B_{\mathbf{l}} \cdot B_{\mathbf{l}}^T$. Here, more than 96 % of the computation time is spent for the matrix assembly. Again, the execution times scale linearly with the number of data points.

3.3 8-dimensional problem

The Pima Indians Diabetes data set from Irvine Machine Learning Database Repository consists of 768 instances with 8 features plus a class label which splits the data into 2 sets with 500 instances and 268 instances respectively, see [8].

	# of points	training correctness	testing correctness	total time (sec)	data matrix time (sec)	# of iterations
	50 000	90.8 %	90.8 %	158	152	41
level 1	500 000	90.7 %	90.8 %	1570	1528	44
	5 million	90.7 %	90.7 %	15933	15514	46
	50 000	91.9 %	91.5 %	1155	1126	438
level 2	500 000	91.5 %	91.6 %	11219	11022	466
	5 million	91.4 %	91.5 %	112656	110772	490

Table 6: Results for a 6D synthetic massive data set, $\lambda = 0.01$

Again we compare with the two best results from [48], the smoothed support vector machine (SSVM) and the robust linear program (RLP) algorithm [6]. The results are given in Table 7.

			sparse grid combination method		
	SSVM [48]	RLP [48]	Level 1 $\lambda = 0.075$	Level 2 $\lambda = 0.25$	Level 3 $\lambda = 1.0$
10-fold train. corr., %	78.11	76.48	83.94	88.51	93.29
10-fold test. corr., %	78.12	76.16	77.47	75.01	72.93

Table 7: Results for the Pima indian diabetes data set

Now our sparse grid combination approach performs already on level 1 with $\lambda = 0.075$ at 77.47 % 10-fold testing correctness. Here the data set seems to be simple enough that one level gives enough resolution for building the necessary class separators. The results on higher levels do not improve but even deteriorate. Our method performs here only slightly worse than the SSVM but clearly better than RLP. Note that the results for the feature selection concave minimization (FSV) algorithm [9], the support vector machine using the 1-norm approach ($\mathrm{SVM}_{||.||_1}$) and the classical support vector machine ($\mathrm{SVM}_{||.||_2^2}$) [9, 17, 70] were reported to be slightly worse in [48].

3.4 9-dimensional problems

We finally consider two problems in 9-dimensional space. Here, due to the limitation of the main memory of our computer we are not able to fully store the system matrices any longer. Instead we only assemble and store the matrices $B_{\mathbf{l}}$ which is possible for the considered problems since M and thus the storage needed for $B_{\mathbf{l}}$ is sufficiently limited. The operations of the matrices $C_{\mathbf{l}}$ and $G_{\mathbf{l}}$ on the vectors are then computed on the fly when needed in the conjugate gradient iteration.

3.4.1 Shuttle Data

The shuttle data set comes from the StatLog Project [52]. It consists of 43 500 observations in the training set and 14 500 data in the testing set and has 9 attributes and 7 classes in the original version. To simplify the problem we use the original class 1, approximately 80 % of the data, as one class and the other 6 classes as class -1. The aim is to get correctness rates on the testing data of more than 99 %. The resulting correctness rates are given in Table 8. We achieve extremely good correctness rates. But note that the full 58 000 data set can be classified with only 19 errors by a rather simple linear decision tree using nine terminal nodes [52]. This shows that the problem is very simple. Finally, we see that the number of iterations needed in the CG-solvers grows with $\lambda \to 0$. This indicates that the condition number of the respective linear systems gets worse with declining values of λ.

# of points	λ	training correctness	testing correctness	total time (sec)	# of iterations
435	0.1	93.8 %	92.8 %	1573	269
	0.01	96.1 %	96.2 %	3719	645
	0.001	99.5 %	99.1 %	11221	1966
	0.0001	99.8 %	99.6 %	34010	5891
	0.00001	100 %	99.6 %	83400	14719
4350	0.1	94.0 %	93.8 %	1849	298
	0.01	97.6 %	97.4 %	4008	653
	0.001	99.6 %	99.4 %	12574	2059
	0.0001	99.7 %	99.8 %	38897	6382
	0.00001	99.7 %	99.7 %	107698	17572
43500	0.1	94.4 %	94.8 %	4019	324
	0.01	98.0 %	98.0 %	7491	692
	0.001	99.5 %	99.5 %	32587	2137
	0.0001	99.7 %	99.6 %	74642	6776
	0.00001	99.8 %	99.8 %	203227	19930

Table 8: Results for the Shuttle data set, only Level 1

3.4.2 Tic-Tac-Toe

This data set comes from the UCI Machine Learning Repository [8]. The 958 instances encode the complete set of possible board configurations at the end of tic-tac-toe games. We did 10-fold cross-validation and got a training correctness of 100 % and a testing correctness of 98.33 % already on level 1 for $\lambda = 0.1, 0, 01, 0, 001$ and 0.0001. In [49], testing correctness rates around 70 % are reported with ASVM. With LSVM [50] and a quadratic kernel, 100% training correctness and 93.33 % testing correctness were achieved.

4 Conclusions

We presented the sparse grid combination technique for the classification of data in moderate-dimensional spaces. Our new method gave good results for a wide range of problems. It works nicely for problems with a small amount of data like that of the UCI repository, but it is also capable to handle huge data sets with 5 million points and more.

The run time scales only linear with the number of data. This is an important property for many practical applications where often the dimension of the problem can substantially be reduced by certain preprocessing steps but the number of data can be extremely huge. We believe that our sparse grid combination method possesses a great potential in such practical application problems. In principle, by computing the matrix entries on the fly in the CG-iteration, classification problems up to 15 dimensions can be dealt with.
We demonstrated for the Ripley data set how the best value of the regularization parameter λ can be determined. This is also of practical relevance.
It is surely necessary to improve on the implementation of the method. Here, instead of grids of d-dimensional brick-type elements with d-linear finite element basis functions, grids of d-simplices might be used with linear finite elements. Thus the storage and run time complexities would only depend linearly on d ($2d+1$ versus 3^d). This would speed up things substantially. A parallel version of the sparse grid combination technique would also reduce the run time of the method significantly. Note that our new method is easily parallelizable on a coarse grain level.
Note furthermore that our approach delivers a continuous classifier function which approximates the data. It therefore can be used without modification for regression problems as well. This is in contrast to many other methods like e.g. decision trees. Also more than two classes can be handled by using isolines with just different values.
Finally, for reasons of simplicity, we used the operator $P = \nabla$. But other differential (e.g. $P = \Delta$) or pseudo-differential operators can here be employed with their associated regular finite element ansatz functions.

References

[1] E. Arge, M. Dæ hlen, and A. Tveito. *Approximation of scattered data using smooth grid functions.* J. Comput. Appl. Math, 59:191–205, 1995.

[2] A. Allen. *Regression and the Moore-Penrose Pseudoinverse.* Academic Press, New York, 1972.

[3] N. Aronszajn. *Theory of reproducing kernels.* Trans. Amer. Math. Soc. 68 1950:337–404, 1950.

[4] R. Balder. *Adaptive Verfahren für elliptische und parabolische Differentialgleichungen auf dünnen Gittern.* PhD thesis, Technische Universität München, 1994.

[5] G. Baszenski. *N–th order polynomial spline blending.* In W. Schempp and K. Zeller, editors, *Multivariate Approximation Theory III*, ISNM 75, pages 35–46. Birkhäuser, Basel, 1985.

[6] K. P. Bennett and O. L. Mangasarian. *Robust linear programming discrimination of two linearly inseparable sets.* Optimization Methods and Software, 1:23–34, 1992.

[7] M. Berry and G. Linoff, *Mastering Data Mining*, Wiley, 2000.

[8] C.L. Blake and C.J. Merz, *UCI Repository of machine learning databases*, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html

[9] P. S. Bradley and O. L. Mangasarian. *Feature selection via concave minimization and support vector machines.* In J. Shavlik, editor, Machine Learning Proceedings of the Fifteenth International Conference(ICML '98), pages 82–90. Morgan Kaufmann, 1998.

[10] H.-J. Bungartz, *Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimensionalen Poisson-Gleichung*, Dissertation, Inst. für Informatik, TU München, 1992.

[11] H.-J. Bungartz, *An adaptive Poisson solver using hierarchical bases and sparse grids*, in Iterative Methods in Linear Algebra, P. de Groen and R. Beauwens, eds., Elsevier, Amsterdam, 1992, pp. 293–310.

[12] H.-J. Bungartz, T. Dornseifer, *Sparse grids: Recent developments for elliptic partial differential equations.* In W. Hackbusch et al, editor, *Multigrid Methods V*, volume 3 of *Lecture Notes in Computational Science and Engineering*, pages 45–70. Springer, 1998.

[13] H.-J. Bungartz, T. Dornseifer, C. Zenger, *Tensor product approximation spaces for the efficient numerical solution of partial differential equations*, to appear in Proc. Int. Workshop on Scientific Computations, Konya, 1996, Nova Science Publishers, 1997.

[14] H.-J. Bungartz, M. Griebel, *A note on the complexity of solving Poisson's equation for spaces of bounded mixed derivatives*, J. Complexity 15 (1999), 167-199.

[15] H.-J. Bungartz, M. Griebel, D. Röschke, C. Zenger, *Pointwise convergence of the combination technique for Laplace's equation*, East-West J. of Numerical Mathematics 1(2) (1994), 21-45.

[16] H.-J. Bungartz, M. Griebel, and U. Rüde. *Extrapolation, combination, and sparse grid techniques for elliptic boundary value problems.* Comput. Methods Appl. Mech. Eng., 116:243–252, 1994.

[17] V. Cherkassky and F. Mulier. *Learning from Data - Concepts, Theory and Methods.* John Wiley & Sons, 1998.

[18] K. Cios, W. Pedrycz, and R. Swiniarski, *Data Mining Methods for Knowledge Discovery*, Kluwer, 1998.

[19] T. Evgeniou, M. Pontil and T. Poggio, *Regularization networks and support vector machines*, Advances in Computational Mathematics, 13 (2000), pp. 1–50.

[20] G. Faber, *Über stetige Funktionen*, Mathematische Annalen, 66 (1909), pp. 81–94.

[21] S.E. Fahlmann and C. Lebiere. The cascade-correlation learning architecture. In Touretzky, editor, *Advances in Neural Information Processing Systems 2.* Morgan-Kaufmann, 1990.

[22] K. Frank, S. Heinrich, and S. Pereverzev. *Information Complexity of Multivariate Fredholm Integral Equations in Sobolev Classes.* J. of Complexity, 12:17–34, 1996.

[23] G. Fung and O.L. Mangasarian. *Data selection for support vector machine classifiers.* Technical report 00-02, Data Mining Institute, Computer Sciences Department, University of Wisconsin, 2000. KDD-2000, Boston august 20-23, 2000.

[24] J. Garcke. Classification with sparse grids, 2000. http://wissrech.iam.uni-bonn.de/people/garcke/SparseMining.html.

[25] J. Garcke, M. Griebel *On the computation of the eigenproblems of hydrogen and helium in strong magnetic and electric fields with the sparse grid combination technique*, Journal of Computational Physics, submitted, 2000, also as SFB 256 preprint 670, Universität Bonn.

[26] T. Gerstner and M. Griebel. *Numerical integration using sparse grids.* Numer. Algorithms, 18:209–232, 1998.

[27] F. Girosi, *An Equivalence Between Sparse Approximation and Support Vector Machines*, Neural Computation 10(6) (1998), 1455-1480.

[28] F. Girosi, M. Jones, T. Poggio, *Priors, Stabilizers and Basis Functions: from regularization to radial, tensor and additive splines*, A.I. Memo No. 1430, Artificial Intelligence Laboratory, MIT, 1993.

[29] F. Girosi, M. Jones, T. Poggio *Regularization Theory and Neural Networks Architectures*, Neural Computation, 7:219-265, 1995.

[30] G. Golub, M. Heath, G. Wahba. *Generalized cross validation as a method for choosing a good ridge parameter*, Technometrics, 21:215-224, 1979.

[31] M. Griebel, *A parallelizable and vectorizable multi-level algorithm on sparse grids*, in Parallel Algorithms for Partial Differential Equations, Notes on Numerical Fluid Mechanics 31, W. Hackbusch, ed., Vieweg, Braunschweig/Wiesbaden, 1991, pp. 94–100.

[32] M. Griebel, *The combination technique for the sparse grid solution of PDEs on multiprocessor machines*, Parallel Processing Letters 2(1) (1992), 61-70.

[33] M. Griebel, *Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences.* Computing, 61:151-179, 1998.

[34] M. Griebel, *Multilevelmethoden als Iterationsverfahren über Erzeugendensystemen*, Teubner Skripten zur Numerik, Teubner, Stuttgart, 1994.

[35] M. Griebel, W. Huber, U. Rüde, T. Störtkuhl, *The combination technique for parallel sparse-grid-preconditioning and -solution of PDEs on multiprocessor machines and workstation networks*, in L. Bouge, M. Cosnard, Y. Robert and D. Trystram (eds.), Lecture Notes in Computer Science 634, Parallel Processing: CONPAR92-VAPP V, pp. 217-228, Springer Verlag, 1992.

[36] M. Griebel, W. Huber, T. Störtkuhl, and C. Zenger, *On the parallel solution of 3D PDEs on a network of workstations and on vector computers*, in Parallel Computer Architectures: Theory, Hardware, Software, Applications, A. Bode and M. D. Cin, eds., vol. 732 of Lecture Notes in Computer Science, Springer Verlag, 1993, pp. 276–291.

[37] M. Griebel, S. Knapek, *Optimized tensor-product approximation spaces*, Constructive Approximation, to appear.

[38] M. Griebel, P. Oswald, *Tensor product type subspace splittings and multilevel iterative methods for anisotropic problems*, Advances in Computational Mathematics 4 (1995), 171-206.

[39] M. Griebel, P. Oswald, and T. Schiekofer. *Sparse grids for boundary integral equations.* Numer. Mathematik, 83(2):279–312, 1999.

[40] M. Griebel, M. Schneider, C. Zenger, *A combination technique for the solution of sparse grid problems*, in Iterative Methods in Linear Algebra, P. de Groen and R. Beauwens (eds.), Elsevier, Amsterdam, 1992, 263–281.

[41] M. Griebel and V. Thurner, *The efficient solution of fluid dynamics problems by the combination technique*, Int. J. Num. Meth. for Heat and Fluid Flow 5(3) (1995), 251-269.

[42] M. Griebel, C. Zenger, S. Zimmer, *Multilevel Gauß -Seidel-algorithms for full and sparse grid problems*, Computing, 50 (1995), pp. 127–148.

[43] M. Hegland, O. M. Nielsen, and Z. Shen, *High dimensional smoothing based on multilevel analysis*, tech. rep., Data Mining Group, The Australian National University, Canberra, November 2000.

[44] J. Hoschek and D. Lasser, *Grundlagen der goemetrischen Datenverarbeitung.* Kapitel 9, Teubner, 1992

[45] Tin Kam Ho and Eugene M. Kleinberg. Checkerboard dataset, 1996. http://www.cs.wisc.edu/math-prog/mpml.html.

[46] L. Kaufman, *Solving the quadratic programming problem arising in support vector classification*, in Advances in Kernel Methods - Support Vector Learning, B. Schölkopf, C. J. C. Burges, and A. J. Smola, eds., MIT Press, 1999, pp. 146–167.

[47] S. Knapek. *Approximation und Kompression mit Tensorprodukt–Multiskalen– Approximationsräumen.* PhD thesis, Universität Bonn, Institut für Angewandte Mathematik, 2000.

[48] Y.J. Lee, O. L. Mangasarian, *SSVM: A Smooth Support Vector Machine for Classification*, Technical Report 99-03, Sep 1999, Data Mining Institute, University of Wisconsin, Computational Optimization and Applications, to appear.

[49] O. L. Mangasarian and David R. Musicant. Active set support vector machine classification. Technical Report 00-04, Data Mining Institute, University of Wisconsin, April 2000.

[50] O. L. Mangasarian and David R. Musicant. Lagrangian support vector machines. Technical Report 00-06, Data Mining Institute, University of Wisconsin, June 2000.

[51] G. Melli. Datgen: A program that creates structured data. www.datasetgenerator.com.

[52] D. Michie, D.J. Spiegelhalter, and C.C. Taylor, editors. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, 1994.

[53] M. A. Olshanskii and A. Reusken. *On the convergence of a multigrid method for linear reaction-diffusion problems.* Bericht Nr. 192, Institut für Geometrie und Praktische Mathematik, RWTH Aachen, 2000.

[54] P. Oswald. *Two remarks on multilevel preconditioners.* Forschungsergebnisse Math-1-91, FSU Jena, 1991.

[55] W.D. Penny and S.J. Roberts. *Bayesian neural networks for classification: how useful is the evidence framework ?* Neural Networks, 12:877–892, 1999.

[56] C. Pflaum and A. Zhou. *Error analysis of the combination technique.* Numer. Math., 1999. to appear.

[57] B. Ripley, *Neural networks and related methods for classification*, Journal of the Royal Statistical Society B, 56 (1994), pp. 409–456.

[58] S. L. Salzberg, *On comparing classifiers: Pitfalls to avoid and a recommended approach*, Data Mining and Knowledge Discovery, 1 (1997), pp. 317–327.

[59] T. Schiekofer, *Die Methode der Finite Differenzen auf dünnen Gittern zur Lösung elliptischer und parabolischer partieller Differentialgleichungen*, Dissertation, Inst. für angewandte Mathematik, Universität Bonn, 1998.

[60] W. Sickel and F. Sprengel. *Interpolation on sparse grids and Nikol'skij–Besov spaces of dominating mixed smoothness. J. Comput. Anal. Appl.*, 1:263–288, 1999.

[61] S. Singh. *2D Spiral Pattern Recognition with Possibilistic Measures* Pattern Recognition Letters, 19(2):141–147, 1998.

[62] A. Smola, B. Schölkopf, K. Müller, *The connection between regularization operators and support vector kernels*, Neural Networks, 11:637-649, 1998.

[63] S. A. Smolyak, *Quadrature and interpolation formulas for tensor products of certain classes of functions*, Dokl. Akad. Nauk SSSR 4 (1963), 240–243.

[64] M. Stone. *Cross-validatory choice and assessment of statistical predictions.* Journal of the Royal Statistical Society, 36:111–147, 1974.

[65] K. Stüben. *Algebraic Multigrid (AMG): An introduction with applications.* GMD Report 53, GMD, 1999.

[66] V. N. Temlyakov, *Approximation of functions with bounded mixed derivative*, Proceedings of the Steklov Institute of Mathematics 1 (1989).

[67] A.N. Tikhonov, V.A. Arsenin, *Solutions of ill-posed problems.* W.H. Winston, Washington D.C., 1977.

[68] F. Utreras, *Cross-validation techniques for smoothing spline functions in one or two dimensions.* In T. Gasser, M. Rosenblatt, editors, *smoothing techniques for curve estimation*, pp. 196-231.Springer-Verlag, Heidelberg, 1979.

[69] V.N. Vapnik, *Estimation of dependences based on empirical data*, Springer-Verlag, Berlin, 1982.

[70] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, 1995.

[71] G. Wahba, *A comparison of GCV and GML for choosing the smoothing parameter in the generalized splines smoothing problem.* The annals of statistics, 13:1378-1402, 1985.

[72] G. Wahba, *Spline models for observational data*, Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990.

[73] H. Yserentant, *On the multi-level splitting of finite element spaces*, Numer. Math., 49 (1986), pp. 379–412.

[74] H. Yserentant, *Hierarchical bases*, in Proc. ICIAM'91, Washington, 1991, R. E. O'Malley, Jr. et al., eds., SIAM, Philadelphia, 1992.

[75] C. Zenger, *Sparse grids*, Parallel Algorithms for Partial Differential Equations, *Notes on Num. Fluid Mech. 31*, W. Hackbusch (ed.), Vieweg, Braunschweig, 1991.