Classification with sparse grids using simplicial basis functions

Jochen Garcke and Michael Griebel
Institut für Angewandte Mathematik
Rheinische Friedrich-Wilhelms-Universität Bonn
Wegelerstraße 6
53115 Bonn, Germany
{garckej, griebel}@iam.uni-bonn.de

Abstract

Recently we presented a new approach [20] 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 a grid in the usually high-dimensional feature space for the minimization process. To cope with the curse of dimensionality, we employ sparse grids [52]. 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. We use the sparse grid combination technique [30] where the classification problem is discretized and solved on a sequence of conventional grids with uniform mesh sizes in each dimension. The sparse grid solution is then obtained by linear combination.

The method computes a nonlinear classifier but scales only linearly with the number of data points and is well suited for data mining applications where the amount of data is very large, but where the dimension of the feature space is moderately high. In contrast to our former work, where d-linear functions were used, we now apply linear basis functions based on a simplicial discretization. This allows to handle more dimensions and the algorithm needs less operations per data point. We further extend the method to so-called anisotropic sparse grids, where now different a-priori chosen mesh sizes can be used for the discretization of each attribute. This can improve the run time of the method and the approximation results in the case of data sets with different importance of the attributes.

We describe the sparse grid combination technique for the classification problem, give implementational details and discuss the complexity of the algorithm. It turns out that the method scales linearly with the number of given data points. Finally we report on the quality of the classifier built by our new method on data sets with up to 14 dimensions. We show that our new method achieves correctness rates which are competitive to those of the best existing methods.

Key words. data mining, classification, approximation, sparse grids, combination technique, simplicial discretization

1 Introduction

Data mining is the process of finding patterns, relations and trends in large data sets. Examples range from scientific applications like the post-processing of data in medicine or the evaluation of satellite pictures to financial and commercial applications, e.g. the assessment of credit risks or the selection of customers for advertising campaign letters. For an overview on data mining and its various tasks and approaches see [5, 12].

In this paper we consider the classification problem arising in data mining. Given is a set of data points in a d-dimensional feature space together with a class label. 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, besides others, decision tree induction, rule learning, adaptive multivariate regression splines, neural networks, and support vector machines. Interestingly, some of these techniques can be interpreted in the framework of regularization networks [23]. 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 [22]. Here, the classification of data is interpreted as a scattered data approximation problem with certain additional regularization terms in high-dimensional spaces.

In [20] we presented a new approach to the classification problem. 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 by finite element methods. Here, a uniform grid would result in $O(h_n^{-d})$ grid points, where d denotes the dimension of the feature space 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 the so-called sparse grids approach which allows to cope with the complexity of the problem to some extent. This method has been originally developed for the solution of partial differential equations [2, 8, 30, 52] and is now used successfully also for integral equations [15, 29], interpolation and approximation [3, 28, 42, 45], eigenvalue problems [18], and integration problems [21]. In the information based complexity community it is also known as 'hyperbolic cross points' and the idea can even be traced back to [44]. 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 of the approximation 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 conventional 'full' grid methods affects sparse grids much less.

In this paper, we apply the sparse grid combination technique [30] to the classification problem. Here, 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 the different grids by linear combination. Thus the classifier is build on sparse grid points and not on data points.

In contrast to [20], where d-linear functions stemming from a tensor-product approach were used, we now apply linear basis functions based on a simplicial discretization. In comparison, this approach allows the processing of more dimensions and needs less operations per data point. A further extension of the approach to so-called anisotropic sparse grids allows to treat attributes with a lot of variance differently than attributes with only a few different values.

A discussion of the complexity of the method gives that the 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 (e.g. 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 and problems with huge synthetical data sets in up to 14 dimensions. It turns out that our new method achieves correctness rates which are competitive to those of the best existing methods. Note that the combination method is simple to use and can be parallelized in a natural and straightforward way, see [19].

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 describe the sparse grid combination technique for the classification problem, discuss its properties and introduce the anisotropic sparse grid combination technique. Furthermore, we present our new approach based on a discretization by simplices and discuss complexity aspects. Section 3 presents the results of numerical experiments conducted with the sparse grid combination method, demonstrates the quality of the classifier build by our new method and compares the results with the ones from [20] and with the ones obtained with different forms of SVMs [11, 49]. 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 [23] 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 [14, 22].

Consider the given set of already classified data (the training set)

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

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 on f. To this end, regularization theory [46, 50] 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

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

with ∇ denoting the gradient and Δ the Laplace operator. The value of λ can be chosen according to cross-validation techniques [13, 24, 40, 47] or to some other principle, such as structural risk minimization [48]. Note that we find exactly this type of formulation in the case $d = 2, 3$ in scattered data approximation methods, see [1, 33], where the regularization term is usually physically motivated.

2.1 Discretization

We now 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

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

with f_N in the finite dimensional space V_N. We plug (2) into (4) and obtain after differentiation with respect to α_k, $k = 1, \ldots, N$

$$0 = \frac{\partial R(f_N)}{\partial \alpha_k} = \frac{2}{M}\sum_{i=1}^{M}\left(\sum_{j=1}^{N}\alpha_j\varphi_j(\boldsymbol{x}_i) - y_i\right)\cdot \varphi_k(\boldsymbol{x}_i) + 2\lambda\sum_{j=1}^{N}\alpha_j(P\varphi_j, P\varphi_k)_{L_2} \tag{5}$$

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

$$\sum_{j=1}^{N} \alpha_j \left[M\lambda(P\varphi_j, P\varphi_k)_{L_2} + \sum_{i=1}^{M} \varphi_j(\boldsymbol{x}_i) \cdot \varphi_k(\boldsymbol{x}_i) \right] = \sum_{i=1}^{M} y_i \varphi_k(\boldsymbol{x}_i). \tag{6}$$

In matrix notation we end up with the linear system

$$(\lambda C + B \cdot B^T)\alpha = By. \tag{7}$$

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 labels 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 V_N. For example if we use the gradient $\Phi(f_N) = ||\nabla f_N||_{L_2}^2$ 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 partial differential equation are Neumann conditions. The discretization (2) gives us then the linear system (7) 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 like radial basis approaches or support vector machines, 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 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 ourselves 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, where n is the refinement level. In the following we always use the gradient $P = \nabla$ in the regularization expression (3). Let $\mathbf{j}$ denote the multi-index $(j_1, ..., j_d) \in \mathbb{N}^d$. We now use piecewise d-linear, i.e. linear in each dimension, so-called hat functions as test- and trial-functions $\phi_{n,\mathbf{j}}(\boldsymbol{x})$ on grid Ω_n. Each basis function $\phi_{n,\mathbf{j}}(\boldsymbol{x})$ is thereby 1 at grid point $\mathbf{j}$ and 0 at all other points of grid Ω_n. A finite element method on grid Ω_n now would give

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

and the variational procedure (4) - (6) would result in the discrete linear system

$$(\lambda C_n + B_n \cdot B_n^T)\alpha_n = B_n y \tag{8}$$

of size $(2^n + 1)^d$ and matrix entries corresponding to (7). Note that f_n lives in the space

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

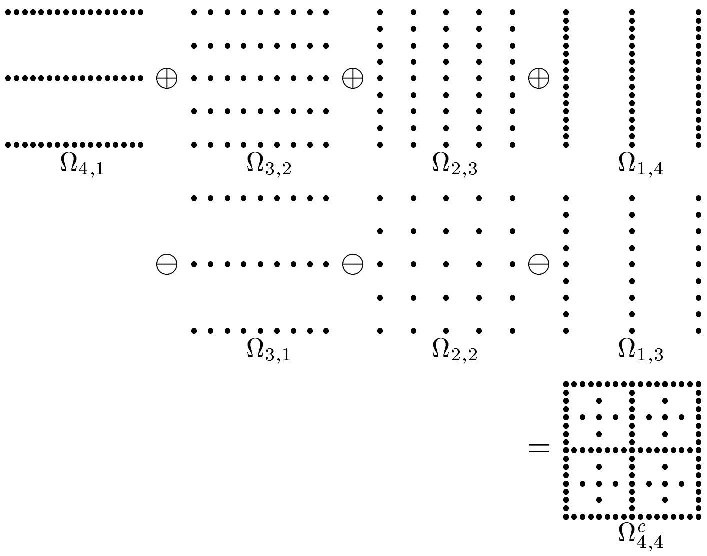

Figure 1: Combination technique with level $n = 4$ in two dimensions

The discrete problem (8) 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 the solution of the resulting linear system by an appropriate solver is clearly not possible for a d-dimensional problem if d is larger than four. The number of grid points is of the order $O(h_n^{-d}) = O(2^{nd})$ and, in the best case, 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 The sparse grid combination technique

Therefore we proceed as follows: We discretize and solve the problem on a certain sequence of grids $\Omega_{\mathbf{l}} = \Omega_{l_1,...,l_d}$ 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

$$l_1 + ... + l_d = n + (d-1) - q, \quad q = 0,..,d-1, \quad l_t > 0. \tag{9}$$

For the two-dimensional case, the grids needed in the combination formula of level 4 are shown in Figure 1. The finite element approach with piecewise d-linear test- and trial-functions

$$\phi_{\mathbf{l},\mathbf{j}}(\mathbf{x}) \;:=\; \prod_{t=1}^{d} \phi_{l_t,j_t}(x_t) \tag{10}$$

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) - (6) would result in the discrete system

$$(\lambda C_{\mathbf{l}} + B_{\mathbf{l}} \cdot B_{\mathbf{l}}^T)\alpha_{\mathbf{l}} = B_{\mathbf{l}} y \tag{11}$$

with the matrices

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

$j_t, k_t = 0, ..., 2^{l_t}, t = 1, ..., d, i = 1, ..., M$, 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 multigrid method with partial semi-coarsening can be applied. The discrete solutions $f_{\mathbf{l}}$ are contained in the spaces

$$V_{\mathbf{l}} \ := \ \text{span}\{\phi_{\mathbf{l},\mathbf{j}}, j_t = 0, ..., 2^{l_t}, t = 1, ..., d\}, \tag{12}$$

of piecewise d-linear functions on grid $\Omega_{\mathbf{l}}$.

Note that all these problems are substantially reduced in size in comparison to (8). 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 [25]. There is also a simple but effective static load balancing strategy available [27].

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

$$f_n^{(c)}(\boldsymbol{x}) := \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{|\mathbf{l}|_1 = n+(d-1)-q} f_{\mathbf{l}}(\boldsymbol{x}). \tag{13}$$

The resulting function $f_n^{(c)}$ lives in the sparse grid space

$$V_n^{(s)} := \bigcup_{\substack{l_1 + ... + l_d = n + (d-1) - q \\ q = 0, ..., d-1 \quad l_t > 0}} V_{\mathbf{l}}.$$

This space has $\dim(V_n^{(s)}) = O(h_n^{-1}(\log(h_n^{-1}))^{d-1})$. It is spanned by a piecewise d-linear hierarchical tensor product basis, see [8].

Note that the summation of the discrete functions from different spaces $V_{\mathbf{l}}$ in (13) involves d-linear interpolation which resembles just the transformation to a representation in the hierarchical basis. For details see [26, 30, 31]. 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 (13) acting directly on the functions $f_{\mathbf{l}}$, i.e.

$$\mathrm{F}(f_n^{(c)}) = \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{l_1+...+l_d = n+(d-1)-q} \mathrm{F}(f_{\mathbf{l}}). \tag{14}$$

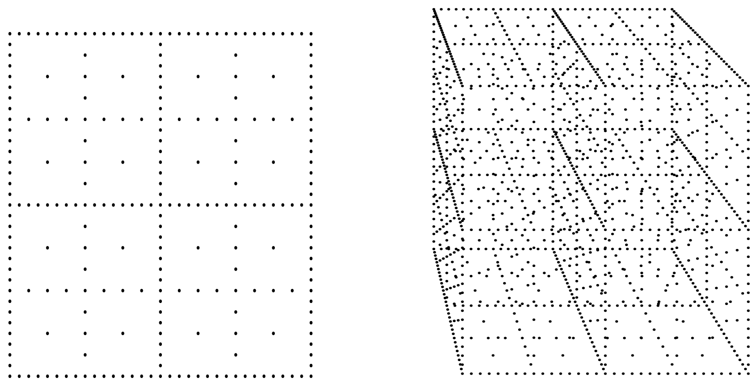

Figure 2: Two-dimensional sparse grid (left) and three-dimensional sparse grid (right), $n = 5$

Therefore, if we now want to evaluate a newly given set of data points $\{\tilde{\boldsymbol{x}}_i\}_{i=1}^{\tilde{M}}$ (the test or evaluation 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 (13). The evaluation of the different $f_{\mathbf{l}}$ in the test points can be done completely in parallel, their summation needs basically an all-reduce/gather operation.

For second order elliptic PDE model problems, it was proven that the combination solution $f_n^{(c)}$ is almost as accurate as the full grid solution f_n, i.e. the discretization error satisfies

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

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

$$|f|_\infty := \left\| \frac{\partial^{2d} f}{\prod_{j=1}^{d} \partial x_j^2} \right\|_\infty \tag{15}$$

to be bounded. Furthermore, a series expansion of the error is necessary for the combination technique. Its existence was shown for PDE model problems in [10].

The combination technique is only one of the various methods to solve problems on sparse grids. Note that there exist also finite difference [26, 41] and Galerkin finite element approaches [2, 8, 9] 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.

In [20] and in (9) we forced $l_t > 0$ which is necessary for the numerical solution of partial differential equations with Dirichlet boundary conditions. But since equation (6) uses Neumann boundary conditions this is not necessary and we can also use the grids $\Omega_{\mathbf{l}}$ with $l_t \geq 0$, i.e.

$$l_1 + ... + l_d = n - q, \quad q = 0, .., d-1, \quad l_t \geq 0 \tag{16}$$

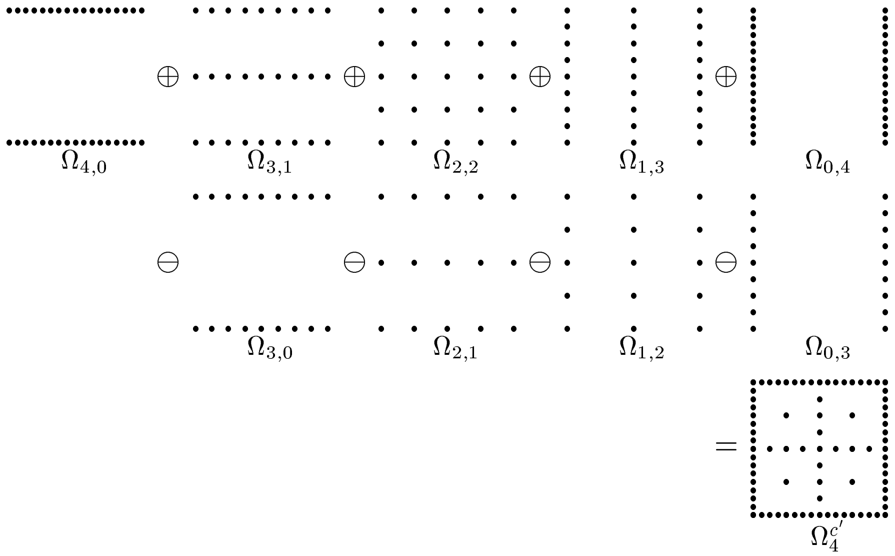

Figure 3: Combination technique based on (16) with level $n = 4$, $d = 2$

for the sequences of grids resulting in the sparse grid $\Omega_n^{c'}$ of level n. In Figure 3 we show the case of $n = 4$ and $d = 2$ of this version of the combination technique.

Now, for n fixed, the grids in the sequence possess less grids points than for (9) and, consequently, less memory is needed. Thus one can handle classification problems with some more attributes than before. Nevertheless a limit on the number of attributes is still imposed due to memory constraints.

2.4 Anisotropic sparse grids

Up to now we treated all attributes of the classification problem the same, i.e. we used the same mesh refinement level for all attributes. Obviously attributes have different properties, different number of distinct values, and different variances. For example to discretize the range of a binary attribute one does not need more than two grid points.

We generalized our approach to account for such situations as well. We use different mesh sizes for each dimension along the lines of [21]. This results in a so-called anisotropic sparse grid. Now different refinement level n_j for each dimension $j, j = 1, \ldots, d$ can be given instead of only the same refinement level n for the different dimensions. This extension of our approach can result in less computing time and better approximation results, depending on the actual data set.

To this end we define the so-called index set $\mathcal{I}_\mathbf{n}$ for given $\mathbf{n} = (n_1, \ldots, n_d)$ of an anisotropic sparse grid. An index set is valid, if for all $\mathbf{k} \in \mathcal{I}_\mathbf{n}$ the following holds

$$\mathbf{k} - \mathbf{e}_j \in \mathcal{I}_\mathbf{n} \quad \text{for} \quad 1 \leq j \leq d,\, k_j > 1, \tag{17}$$

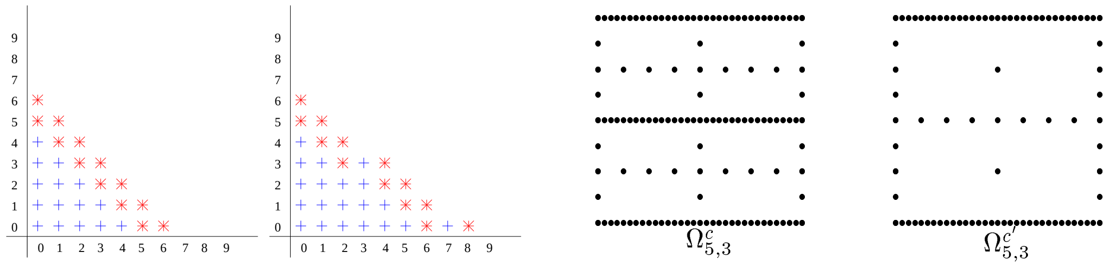

Figure 4: Index sets $\mathcal{I}_{\mathbf{n}}$ for $\mathbf{n} = (6, 6)$ and $\mathbf{n} = (6, 8)$ (left), and anisotropic sparse grid $\Omega^c_{\mathbf{n}}$ with $\mathbf{n} = (5, 3)$ based on (9) and (16), respectively (right)

with $\mathbf{e}_j$ denoting the j-th unit vector.

For a given $\mathbf{n} = (n_1, \ldots, n_d)$ we now define the index set $\mathcal{I}_{\mathbf{n}}$ as all the indices which are below or on the associated hyperplane defined by the indices $(n_1, 0, \ldots, 0), \ldots, (0, \ldots, 0, n_j, 0, \ldots, 0)$, $\ldots, (0, \ldots, 0, n_d)$. Thus, $\mathcal{I}_{\mathbf{n}}$ consists of all indices $\mathbf{k} = (k_1, \ldots, k_d)$ with

$$\sum_{j=1}^{d} \frac{k_j}{n_j} \leq 1, \quad \text{with} \quad k_j \geq 0, \tag{18}$$

where we set $k_j/n_j := 0$ for $k_j = n_j = 0$. The union of all grids associated to the indices in $\mathcal{I}_{\mathbf{n}}$ gives an anisotropic sparse grid of level $\mathbf{n}$ in the sense of (16). For the corresponding definition in the sense of (9) we have to replace $\mathbf{n}$ by $\mathbf{n} - \mathbf{1}$ and $\mathbf{k}$ by $\mathbf{k} - \mathbf{1}$, with $\mathbf{1} = (1, \ldots, 1)$, in (18). Figure 4 (right) shows the anisotropic sparse grid of level (5,3) for both variants. For example the index set of the sparse grid $\Omega^c_{5,3}$, in the sense of (9), is characterised by the indices $(5, 1), (3, 2), (1, 3)$, and consists of these 3 indices and all smaller ones.

We define the characteristic function $\chi^{\mathcal{I}_{\mathbf{n}}}$ of $\mathcal{I}_{\mathbf{n}}$ as

$$\chi^{\mathcal{I}_{\mathbf{n}}}(\mathbf{k}) = \begin{cases} 1 & \text{if } \mathbf{k} \in \mathcal{I}_{\mathbf{n}}, \\ 0 & \text{else.} \end{cases}$$

The generalized combination technique is now defined as follows:

$$f^{(c)}_{\mathcal{I}_{\mathbf{n}}}(\boldsymbol{x}) := \sum_{\mathbf{k} \in \mathcal{I}_{\mathbf{n}}} \left(\sum_{\mathbf{z}=(0,\ldots,0)}^{(1,\ldots,1)} (-1)^{|\mathbf{z}|_1} \cdot \chi^{\mathcal{I}_{\mathbf{n}}}(\mathbf{k} + \mathbf{z}) \right) f_{\mathbf{k}}(\boldsymbol{x}). \tag{19}$$

Of course, only if the sum over the $\chi^{\mathcal{I}_{\mathbf{n}}}$ of the neighbors $\mathbf{k} + \mathbf{z}$ is non-zero for a $\mathbf{k} \in \mathcal{I}_{\mathbf{n}}$, the discrete problem (11) on the corresponding grid $\Omega_{\mathbf{k}}$ has to be dealt with to compute the classifier (19) on the anisotropic sparse grid $\Omega^c_{\mathbf{n}}$. In Figure 4 (left) these non-zero indices are marked by stars, zero indices by crosses.

The difficulty now is to know a-priori for which dimensions higher refinement levels n_j are worthwhile. Sometimes further knowledge on the data is available which determines such dimensions but, in general, strategies like cross-validation are necessary to find the best anisotropic sparse grid.

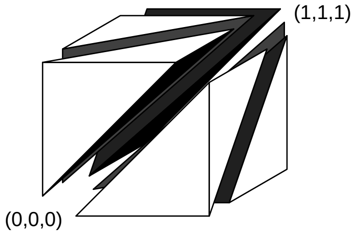

Figure 5: Kuhn's triangulation of a three-dimensional unit cube

2.5 Simplicial basis functions

So far we only mentioned d-linear basis functions based on a tensor-product approach, this case was presented in detail in [20]. But, on the grids of the combination technique, linear basis functions based on a simplicial discretization are also possible. Here, we use the so-called Kuhn's triangulation [16, 34] for each rectangular block, see Figure 5. Now, the summation of the discrete functions for the different spaces V_l in (13) only involves linear interpolation.

	d-linear basis functions		linear basis functions	
	$C_{\mathbf{l}}$	$G_{\mathbf{l}} := B_{\mathbf{l}} \cdot B_{\mathbf{l}}^T$	$C_{\mathbf{l}}$	$G_{\mathbf{l}} := B_{\mathbf{l}} \cdot B_{\mathbf{l}}^T$
storage	$O(3^d \cdot N)$	$O(3^d \cdot N)$	$O((2 \cdot d + 1) \cdot N)$	$O(2^d \cdot N)$
assembly	$O(3^d \cdot N)$	$O(d \cdot 2^{2d} \cdot M)$	$O((2 \cdot d + 1) \cdot N)$	$O((d+1)^2 \cdot M)$
mv-multiplication	$O(3^d \cdot N)$	$O(3^d \cdot N)$	$O((2 \cdot d + 1) \cdot N)$	$O(2^d \cdot N)$

Table 1: Complexities of the storage, the assembly and the matrix-vector multiplication for the different matrices arising in the combination method on one grid $\Omega_{\mathbf{l}}$ for both discretization approaches. Note that $C_{\mathbf{l}}$ and $G_{\mathbf{l}}$ can be stored together in one matrix structure.

The theoretical properties of this variant of the sparse grid technique still has to be investigated in more detail. However the results which are presented in section 3 warrant its use. We see, if at all, just slightly worse results with linear basis functions than with d-linear basis functions and we believe that our new approach results in the same approximation order.

Since in our new variant of the combination technique the overlap of supports, i.e. the regions where two basis functions are both non-zero, is greatly reduced due to the use of a simplicial discretization, the complexities scale significantly better. This concerns both the costs of the assembly and the storage of the non-zero entries of the sparsely populated matrices from (8), see Table 1. Note that for general operators P the complexities for $C_{\mathbf{l}}$ scale with $O(2^d \cdot N)$. But for our choice of $P = \nabla$ structural zero-entries arise, which need not to be considered and which further reduce the complexities, see Table 1 (right), column $C_{\mathbf{l}}$. The actual iterative solution process (by a diagonally preconditioned conjugate gradient method) scales independent of the number of data points for both approaches.

Note however that both the storage and the run time complexities still 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 $d \leq 10$ for d-linear basis

level	λ	training correctness	testing correctness
5	0.0003	94.87 %	82.99 %
6	0.0006	97.42 %	84.02 %
7	0.00075	100.00 %	88.66 %
8	0.0006	100.00 %	89.18 %
9	0.0006	100.00 %	88.14 %

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

functions and $d \leq 15$ for linear basis functions. A decomposition of the matrix entries over several computers in a parallel environment would permit more dimensions.

3 Numerical results

We now apply our approach to different test data sets. Here we mainly consider the properties of the method with simplicial basis functions, the main result of this paper, and give only a few results with anisotropic sparse grids. Both synthetical data and real data from practical data mining applications are used. All the data sets are rescaled to $[0, 1]^d$ for ease of computation. For data sets with a small number of data points we map the median of each attribute to 0.5 and scale the other values accordingly. The bigger data sets are just normalized to $[0, 1]^d$. To evaluate our method we give the correctness rates on testing data sets, if available, or the ten-fold cross-validation results otherwise. For further details and a critical discussion on the evaluation of the quality of classification algorithms see [13, 40]. The run times are measured on a Pentium III 700 MHz machine.

3.1 Two-dimensional problems

We first consider synthetic two-dimensional problems with small sets of data which correspond to certain structures.

3.1.1 Spiral

The first example is the spiral data set proposed by Alexis Wieland of MITRE Corp [51]. Here, 194 data points describe two intertwined spirals, see Figure 6. This is surely an artificial problem which does not appear in practical applications. However it serves as a hard test case for new 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 [43].

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 8 with 89.18% in comparison to 77.20% in [43].

In Figure 6 we show the corresponding results obtained with our sparse grid combination method for the levels 5 to 8. With level 7 the two spirals are clearly detected and resolved. Note that here 1281 grid points are contained in the sparse grid. For level 8 (2817 sparse grid

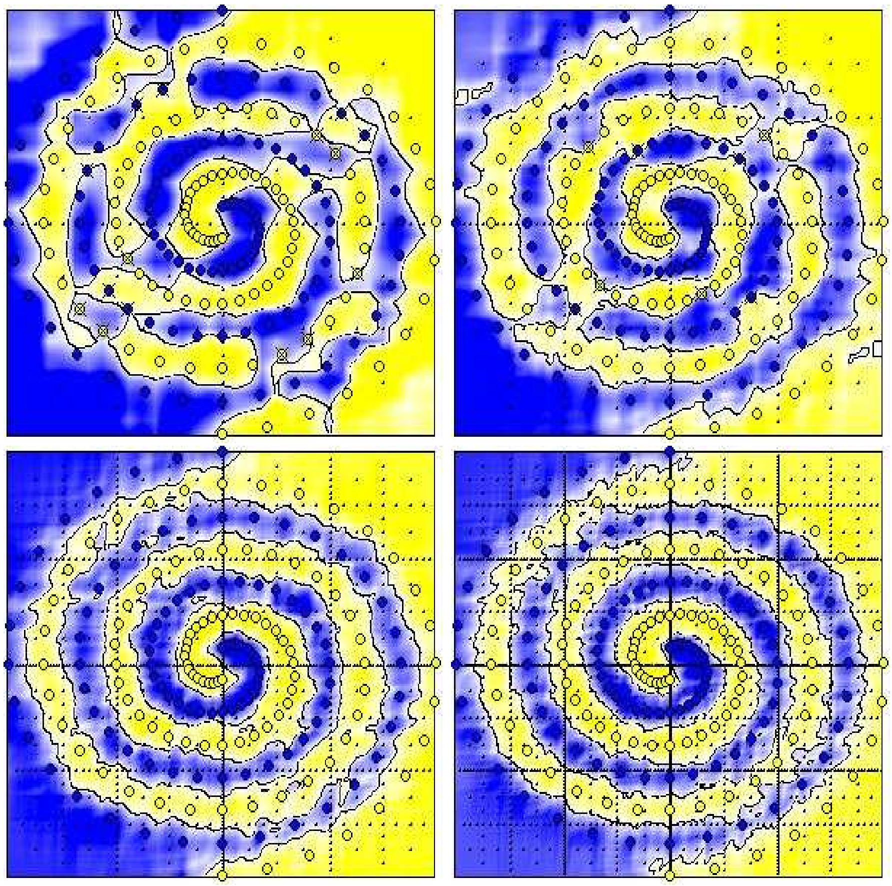

Figure 6: Spiral data set, sparse grid with level 5 (top left) to 8 (bottom right)

points) the shape of the two reconstructed spirals gets smoother and the reconstruction gets more precise.

3.1.2 Ripley

This data set, taken from [39], 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 the training set to determine the best regularization parameter λ per ten-fold cross-validation. The best test correctness rate and the corresponding λ are given for different levels n in the first two columns of Table 3. With this λ we then compute the sparse grid classifier from the 250 training data. Column three of Table 3 gives the result of this classifier on the (previously unknown) test data set. We see that our method works well. Already level 4 is sufficient to obtain a test

	linear basis			d-linear basis	best possible %	
level	ten-fold test %	λ	test data %	test data %	linear	d-linear
1	85.2	0.0020	89.9	89.8	90.6	90.3
2	85.2	0.0065	90.3	90.4	90.4	90.9
3	88.4	0.0020	90.9	90.6	91.0	91.2
4	87.2	0.0035	91.4	90.6	91.4	91.2
5	88.0	0.0055	91.3	90.9	91.5	91.1
6	86.8	0.0045	90.7	90.8	90.7	90.8
7	86.8	0.0008	89.0	88.8	91.1	91.0
8	87.2	0.0037	91.0	89.7	91.2	91.0
9	87.7	0.0015	90.1	90.9	91.1	91.0
10	89.2	0.0020	91.0	90.6	91.2	91.1

Table 3: Results for the Ripley data set

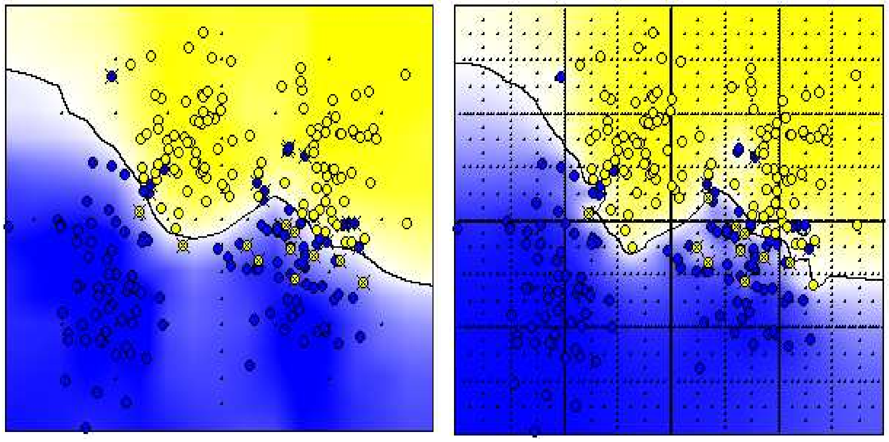

Figure 7: Ripley data set, combination technique with linear basis functions. Left: level 4, λ = 0.0035. Right: level 8, λ = 0.0037

correctness rate of 91.4 %. The reason is surely the relative simplicity of the data, see Figure 7. Just a few hyperplanes should be enough to separate the classes quite properly. We also see that there is not much need to use any higher levels, on the contrary there is even an overfitting effect visible in Figure 7. In column 4 we show the results from [20]. We see that we achieve almost the same results with d-linear functions. To see what kind of results could be possible with a more sophisticated strategy for determining λ we give in the last two columns of 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 the approach to determine the value of λ from the training set by cross-validation works well. Again we have almost the same results with linear and d-linear

		Sparse Grid Combination Technique						Results with SVM[17]	
		d-linear		linear		lin. dim 3 refined			
Level	10-fold		time (sec)		time (sec)		time (sec)	linear	non-linear
1	train	77.7 %	2.2	82.3 %	0.1	78.7 %	0.2	70.2 %	75.8 %
	test	71.8 %		72.4 %		73.9 %		70.0 %	73.7 %
2	train	84.3 %	27.0	80.0 %	1.0	84.1 %	1.4	time (sec.)	
	test	70.4 %		72.5 %		71.6 %		0.8	20.6
3	train	91.4 %	194.1	86.6 %	9.6	88.0 %	10.8		
	test	70.8 %		69.9 %		70.5 %			
4	train	92.6 %	1217.6	94.2 %	68.3	93.1 %	75.4		
	test	68.8 %		71.4 %		70.5 %			

Table 4: Results for the BUPA liver disorders data set

basis functions. Note that a testing correctness of 90.6 % and 91.1 % was achieved with neural networks in [39] and [38], respectively, for this data set.

3.2 Small data sets

3.2.1 BUPA Liver

The BUPA Liver Disorders data set from Irvine Machine Learning Database Repository [6] consists of 345 data points with 6 features and a selector field used to split the data into 2 sets with 145 instances and 200 instances respectively. Here we have no test data and therefore can only report our ten-fold cross-validation results.

We compare with our d-linear results from [20] and with results using linear and non-linear support vector machines. The results are given in Table 4. Our sparse grid combination approach with linear basis functions performs slightly better than the d-linear approach, the best result was 72.5% testing correctness on level 2, and needs significantly less computing time. It also performs better than a linear SVM (70.0 %), but worse than a non-linear SVM (73.7 %), see [17].

Refining dimension 3 once, i.e. using the anisotropic grid $\Omega^c_{1,1,2,1,1,1}$, we achieve the best result of 73.9 % using linear basis functions. We picked this attribute for refinement through cross-validation.

3.2.2 Galaxy Dim

The Galaxy Dim data set is a commonly used subset of the data presented in [37]. It consists of 4192 data points with 14 attributes, the two classes have almost the same number of instances. Again no test data is present and therefore we can only report our ten-fold cross-validation results. Since this data set is in 14 dimensions we use the combination technique of equation (16) with simplicial basis functions to be able to fit the matrixes into main memory.

With isotropic sparse grids we achieve a testing correctness of 95.6 % for level 0 and on level 1 a slightly better result with 95.9 %, see Table 5. If we refine dimension 5 once we already achieve with level 0 the slightly better results of 96.0 %. We need for this result with 66 seconds

		Sparse Grid Combination Technique				Results with
		normal		dim 5 refined		linear
Level	10-fold		time (sec)		time (sec)	SVM[17]
0	train	97.2 %	58	97.9 %	66	95.0 %
	test	95.6 %		96.0 %		95.0 %
1	train	97.6 %	965	98.1 %	1185	time (sec.)
	test	95.9 %		96.2 %		5.2

Table 5: Results for the Galaxy Dim data set

far less time than for the original level 1 result (965 seconds). If we use level 1 with dimension 5 refined once, the result improves slightly to 96.2 %. Note that with linear SVMs a 95.0 % correctness rate in 5 seconds was achieved in [17].

3.3 Big and massive data sets

3.3.1 Synthetic massive data set in 6D

To measure the performance on a massive data set we produced with DatGen [36] a 6-dimensional test case 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:0/200,R,O:0/200,R,O -R2 -C2/4 -D2/5 -T10/60 -O5020000 -p -e0.15. The results are given in Table 6. Note that already on level 1 a testing correctness of over 90 % was achieved with just $\lambda = 0.01$. The main observation on this test case concerns the execution time. Besides the total run time, we also give the CPU time which is needed for the computation of the matrices $G_{\mathbf{l}} = B_{\mathbf{l}} \cdot B_{\mathbf{l}}^T$.

We see that with linear basis functions really huge data sets of 5 million points can be processed in reasonable time. Note that more than 50 % of the computation time is spent for the data matrix assembly only and, more importantly, that the execution time scales linearly with the number of data points. The latter is also the case for the d-linear functions, but, as mentioned, this approach needs more operations per data point and results in a much longer execution time, compare also Table 6. Especially the assembly of the data matrix needs more than 96 % of the total run time for this variant. For our present example the linear basis approach is about 40 times faster than the d-linear approach on the same refinement level, e.g. for level 2 we need 17 minutes in the linear case and 11 hours in the d-linear case. For higher dimensions this factor will be even larger.

3.3.2 Forest cover type

The forest cover type data set comes from the UCI KDD Archive [4], it was also used in [32], where an approach similar to ours was followed. It consists of cartographic variables for 30 x 30 meter cells. Here, a forest cover type is to be predicted. The 12 originally measured attributes resulted in 54 attributes in the data set, besides 10 quantitative variables there are 4 binary wilderness areas and 40 binary soil type variables. We only use the 10 quantitative variables. The class label has 7 values, Spruce/Fir, Lodgepole Pine, Ponderosa Pine, Cottonwood/Willow, Aspen, Douglas-fir and Krummholz. Like [32] we only report results for the classification of

	# of points	training correctness	testing correctness	total time (sec)	data matrix time (sec)	# of iterations
		linear basis functions				
level 1	50 000	90.4	90.5	3	1	23
	500 000	90.5	90.5	25	8	25
	5 million	90.5	90.6	242	77	28
level 2	50 000	91.4	91.0	12	5	184
	500 000	91.2	91.1	110	55	204
	5 million	91.1	91.2	1086	546	223
level 3	50 000	92.2	91.4	48	23	869
	500 000	91.7	91.7	417	226	966
	5 million	91.6	91.7	4087	2239	1057
		d-linear basis functions				
level 1	500 000	90.7	90.8	597	572	91
	5 million	90.7	90.7	5897	5658	102
level 2	500 000	91.5	91.6	4285	4168	656
	5 million	91.4	91.5	42690	41596	742

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

Ponderosa Pine, which has 35754 instances out of the total 581012.

Since far less than 10 % of the instances belong to Ponderosa Pine we enforce this class with a factor of 5, i.e. Ponderosa Pine has a class value of 5, all others of -1 and the threshold value for separating the classes is 0. The data set was randomly separated into a training set, a test set, and a evaluation set, all similar in size.

In [32] only results up to 6 dimensions could be reported. In Table 7 we present our results for the 6 dimensions chosen there, i.e. the dimensions 1,4,5,6,7, and 10, and for all 10 dimensions as well. To give an overview of the behavior over several λ's we present for each level n the overall correctness results, the correctness results for Ponderosa Pine and the correctness result for the other class for three values of λ. We then give results on the evaluation set for a chosen λ.

We see in Table 7 that already with level 1 we have a testing correctness of 93.95 % for the Ponderosa Pine in the 6 dimensional version. Higher refinement levels do not give better results. The result of 93.52% on the evaluation set is almost the same as the corresponding testing correctness. Note that in [32] a correctness rate of 86.97 % was achieved on the evaluation set.

The usage of all 10 dimensions improves the results slightly, we get 93.81 % as our evaluation result on level 1. As before, higher refinement levels do not improve the results for this data set.

In [32] it was remarked that the elevation is the dominant attribute of this data set. We therefore choose this dimension for refinement. In Table 8 we present the results. For a conventional isotropic sparse grid a finer resolution for all dimensions does not improve the result, but if we just refine the sparse grid discretization for the elevation attribute only, the results improve slightly for one refinement step. It improves further for two and three refinement

	λ	testing correctness overall	Ponderosa Pine	other class	time (sec.)
		6 dimensions			
level 1	0.0005	92.7 %	93.9 %	92.6 %	3.4
	0.0050	92.5 %	94.0 %	92.4 %	3.4
	0.0500	92.5 %	93.4 %	92.4 %	3.4
on evaluation set	0.0050	92.5 %	93.5 %	92.4 %	3.4
level 2	0.0001	93.4 %	92.1 %	93.4 %	23.1
	0.0010	93.2 %	92.3 %	93.3 %	23.1
	0.0100	92.3 %	89.0 %	92.5 %	23.1
on evaluation set	0.0010	93.2 %	91.7 %	93.3 %	23.1
level 3	0.0010	92.8 %	90.9 %	92.9 %	93.9
	0.0100	93.1 %	91.7 %	93.2 %	93.8
	0.1000	93.5 %	88.0 %	93.9 %	93.8
on evaluation set	0.0100	93.0 %	91.4 %	93.1 %	93.8
		10 dimensions			
level 1	0.0025	93.6 %	94.0 %	93.6 %	20.1
	0.0250	93.6 %	94.2 %	93.5 %	18.2
	0.2500	93.6 %	92.3 %	93.7 %	17.8
on evaluation set	0.0250	93.5 %	93.8 %	93.5 %	18.2
level 2	0.0050	93.0 %	92.4 %	93.0 %	316.6
	0.0500	93.7 %	93.0 %	93.7 %	282.2
	0.5000	93.1 %	91.8 %	93.2 %	273.6
on evaluation set	0.0500	93.7 %	92.9 %	93.8 %	282.2

Table 7: Results for forest cover type data set using 6 and 10 attributes

steps to the best result of 95.5 % evaluation correctness for Ponderosa Pine. Note that the results did not improve for the anisotropic grids $\Omega^c_{k,2,\ldots,2}$, $k = 3, 4, 5$.

Note that the forest cover example is sound enough to serve as an example of classification, but it might strike forest scientists as being amusingly superficial. It has been known for 30 years that the dynamics of forest growth can have a dominant effect on which species is present at a given location [7], yet there are no dynamic variables in the classifier. This can be seen as a warning never to assume that the available data contains all the relevant information.

3.3.3 ndcHard data set

This 10-dimensional data set of two million instances was synthetically generated and first used in [35]. Like in the synthetical 6-dimensional example of section 3.3.1 the main observations concern the run time. Besides the total run time, we also give the CPU time which is needed for the computation of the matrices $G_\mathbf{l} = B_\mathbf{l} \cdot B_\mathbf{l}^T$. Note that the highest amount of memory needed (for level 2 in the case of 2 million data points) was 350 MBytes, about 250 MBytes for the matrix and about 100 MBytes for keeping the data points in memory.

In Table 9 we give the results using the combination technique of equation (9). More than

	λ	testing correctness overall	Ponderosa Pine	other class	time (sec.)
$\Omega^c_{2,1,\ldots,1}$	0.0050	93.9 %	94.4 %	93.9 %	29.6
on evaluation set	0.0050	93.9 %	94.0 %	93.9 %	29.6
$\Omega^c_{3,1,\ldots,1}$	0.0200	94.2 %	95.2 %	94.1 %	47.7
on evaluation set	0.0200	94.2 %	95.2 %	94.2 %	47.7
$\Omega^c_{4,1,\ldots,1}$	0.00150	94.6 %	95.5 %	94.5 %	115.1
on evaluation set	0.00150	94.6 %	95.5 %	94.6 %	115.4

Table 8: Results for forest cover type data set with anisotropic grids

50 % of the run time is spent for the assembly of the data matrix. The time needed for the data matrix scales linearly with the number of data points. The total run time seems to scale even better than linear. Already for level 1 we get a 84.9 % testing correctness, no improvement with level 2 is achieved.

Since all dimensions are treated equally during the generation of the data set one should expect no improvement through the use of anisotropic sparse grids. Numerical tests with refined dimensions confirm this expectation. Note that with support vector machines correctness rates of 69.5 % were reported in [17, 35].

	# of points	training correctness	testing correctness	total time (sec)	data matrix time (sec)	# of iterations
level 1	20 000	86.2 %	84.2 %	6.3	0.9	45
	200 000	85.1 %	84.8 %	16.2	8.7	51
	2 million	84.9 %	84.9 %	114.9	84.9	53
level 2	20 000	85.1 %	83.8 %	134.6	10.3	566
	200 000	84.5 %	84.2 %	252.3	98.2	625
	2 million	84.3 %	84.2 %	1332.2	966.6	668

Table 9: Results for the ndcHard data set

4 Conclusions

We presented the anisotropic sparse grid combination technique with linear basis functions based on simplices for the classification of data in moderate-dimensional spaces. Our new method gave good results for a wide range of problems. It is capable to handle huge data sets with millions of points. The run time scales only linearly 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.

For the used data sets the full effect of an anisotropic sparse grid did not take place since the best results were often achieved already for level 1 or 2.

Right now the refinement levels for each attribute have to be given a-priori. To find good values for these one has to use either trial-and-error approaches with cross-validation or knowledge about the attributes. For numerical integration [21] or the numerical solution of partial differential equations self-adaptive strategies are well-known and widely used. These ideas should be transferable to the classification problem. Furthermore, such dimension adaptive refinement strategies could be extended to a tool for dimension reduction.

A parallel version of the sparse grid combination technique reduces the run time significantly, see [19]. Note that our method is easily parallelizable already on a coarse grain level. A second level of parallelization is possible on each grid of the combination technique with the standard techniques known from the numerical treatment of partial differential equations.

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.

Finally, for reasons of simplicity, we used the operator $P = \nabla$. But other differential (e.g. $P = \Delta$) or pseudo-differential operators can be employed here with their associated regular finite element ansatz functions.

5 Acknowledgements

Part of the work was supported by the German Bundesministerium für Bildung und Forschung (BMB+F) within the project 03GRM6BN. This work was carried out in cooperation with Prudential Systems Software GmbH, Chemnitz. The authors thank one of the original referees for his remarks on the forest cover data set.

References

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

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

[3] 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.

[4] S. D. Bay. The UCI KDD archive. http://kdd.ics.uci.edu, 1999.

[5] M. J. A. Berry and G. S. Linoff. *Mastering Data Mining.* Wiley, 2000.

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

[7] D. Botkin, J. Janak, and J. Wallis. Some ecological consequences of a computer model of forest growth. *J. Ecology*, 60:849–872, 1972.

[8] H.-J. Bungartz. *Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimensionalen Poisson-Gleichung.* Dissertation, Institut für Informatik, Technische Universität München, 1992.

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

[10] H.-J. Bungartz, M. Griebel, D. Röschke, and C. Zenger. Pointwise convergence of the combination technique for the Laplace equation. *East-West J. Numer. Math.*, 2:21–45, 1994.

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

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

[13] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924, 1998.

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

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

[16] H. Freudenthal. Simplizialzerlegungen von beschränkter Flachheit. *Annals of Mathematics*, 43:580–582, 1942.

[17] G. Fung and O. Mangasarian. Proximal support vector machine classifiers. In F. Provost and R. Srikant, editors, *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 77–86, 2001.

[18] J. Garcke and 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*, 165(2):694–716, 2000.

[19] J. Garcke and M. Griebel. On the parallelization of the sparse grid approach for data mining. In *Large-Scale Scientific Computations, Third International Conference, LSSC 2001, Sozopol, Bulgaria*, volume 2179 of *Lecture Notes in Computer Science*, pages 22-32, 2001.

[20] J. Garcke, M. Griebel, and M. Thess. Data mining with sparse grids. *Computing*, 67(3):225–253, 2001.

[21] T. Gerstner and M. Griebel. Numerical Integration using Sparse Grids. *Numer. Algorithms*, 18:209–232, 1998.

[22] F. Girosi. An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10(6):1455–1480, 1998.

[23] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7:219–265, 1995.

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

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

[26] M. Griebel. Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences. *Computing*, 61(2):151–179, 1998.

[27] 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 A. Bode and M. D. Cin, editors, *Parallel Computer Architectures: Theory, Hardware, Software, Applications*, volume 732 of *Lecture Notes in Computer Science*, pages 276–291. Springer Verlag, 1993.

[28] M. Griebel and S. Knapek. Optimized tensor-product approximation spaces. *Constructive Approximation*, 16(4):525–540, 2000.

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

[30] M. Griebel, M. Schneider, and C. Zenger. A combination technique for the solution of sparse grid problems. In P. de Groen and R. Beauwens, editors, *Iterative Methods in Linear Algebra*, pages 263–281. IMACS, Elsevier, North Holland, 1992.

[31] 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):251–269, 1995.

[32] M. Hegland, O. M. Nielsen, and Z. Shen. High dimensional smoothing based on multilevel analysis. Technical report, Data Mining Group, The Australian National University, Canberra, November 2000. Submitted to SIAM J. Scientific Computing.

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

[34] H. W. Kuhn. Some combinatorial lemmas in topology. *IBM J. Res. Develop.*, 4:518–524, 1960.

[35] O. L. Mangasarian and D. R. Musicant. Lagrangian support vector machines. *Journal of Machine Learning Research*, 1:161–177, 2001.

[36] G. Melli. Datgen: A program that creates structured data. Website. http://www.datasetgenerator.com.

[37] S. Odewahn, E. Stockwell, R. Pennington, R. Humphreys, and W. Zumach. Automated star/galaxy discrimination with neural networks. *Astronomical Journal*, 103(1):318–331, 1992.

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

[39] B. D. Ripley. Neural networks and related methods for classification. *Journal of the Royal Statistical Society B*, 56(3):409–456, 1994.

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

[41] T. Schiekofer. *Die Methode der Finiten Differenzen auf dünnen Gittern zur Lösung elliptischer und parabolischer partieller Differentialgleichungen.* Dissertation, Institut für Angewandte Mathematik, Universität Bonn, 1999.

[42] 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.

[43] S. Singh. 2d spiral pattern recognition with possibilistic measures. *Pattern Recognition Letters*, 19(2):141–147, 1998.

[44] S. A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Dokl. Akad. Nauk SSSR*, 148:1042–1043, 1963. Russian, Engl. Transl.: Soviet Math. Dokl. 4:240–243, 1963.

[45] V. N. Temlyakov. Approximation of functions with bounded mixed derivative. *Proc. Steklov Inst. Math.*, 1, 1989.

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

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

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

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

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

[51] A. Wieland. Spiral data set. http://www.cs.cmu.edu/afs/cs.cmu.edu/project/ai-repository/ai/areas/neural/bench/cmu/0.html.

[52] C. Zenger. Sparse grids. In W. Hackbusch, editor, *Parallel Algorithms for Partial Differential Equations, Proceedings of the Sixth GAMM-Seminar, Kiel, 1990*, volume 31 of *Notes on Num. Fluid Mech.* Vieweg-Verlag, 1991.