ANZIAM J. 48(2006), 11–22

PARALLELISATION OF SPARSE GRIDS FOR LARGE SCALE DATA ANALYSIS

JOCHEN GARCKE✉ [1] [2], MARKUS HEGLAND[2] and OLE NIELSEN[2]

(Received 28 November, 2005; revised 27 February, 2006)

Abstract

Sparse grids are the basis for efficient high dimensional approximation and have recently been applied successfully to predictive modelling. They are spanned by a collection of simpler function spaces represented by regular grids. The sparse grid combination technique prescribes how approximations on a collection of anisotropic grids can be combined to approximate high dimensional functions.

In this paper we study the parallelisation of fitting data onto a sparse grid. The computation can be done entirely by fitting partial models on a collection of regular grids. This allows parallelism over the collection of grids. In addition, each of the partial grid fits can be parallelised as well, both in the assembly phase, where parallelism is done over the data, and in the solution stage using traditional parallel solvers for the resulting PDEs. Using a simple timing model we confirm that the most effective methods are obtained when both types of parallelism are used.

2000 *Mathematics subject classification*: primary 65Y05, 65D15; secondary 62G08.

Keywords and phrases: predictive modelling, sparse grids, parallelism, numerical linear algebra.

1. Introduction

Data mining algorithms have to address two major computational challenges. First, they have to be able to handle large and growing datasets and secondly, they need to be able to process complex data. Datasets used in data mining studies have been doubling in size every year and many are now in the terabyte range. The second challenge is sometimes referred to as the *curse of dimensionality* as the algorithmic

[1] Institut für Numerische Simulation, Rheinische Friedrich-Wilhelms-Universität Bonn, Wegelerstr. 6, 53115 Bonn, Germany; e-mail: garcke@ins.uni-bonn.de.

[2] Centre for Mathematics and its Applications, Mathematical Sciences Institute, Australian National University, Canberra ACT 0200, Australia; e-mail: jochen.garcke@anu.edu.au, markus.hegland@anu.edu.au, ole.nielsen@ga.gov.au.

https://doi.org/10.1017/S1446181100003382 Published online by Cambridge University Press

complexity grows exponentially in the number of features or dimension of the data. Parallel processing is a major tool in addressing the large computational requirements of data mining algorithms.

Data mining aims to find patterns or structure in the data. One important type of pattern discovered in data mining algorithms is represented by functions between selected dataset features. In data mining, the discovery of such functions is referred to as *predictive modelling* and includes both classification and regression. Here we will consider regression but the same algorithms are used in classification as well. Different types of models are obtained from different function classes. The classical methods of least squares use linear and nonlinear functions with relatively few parameters. Modern nonparametric methods are characterised by large numbers of parameters and can flexibly approximate general function sets. They include artificial neural nets [2], Bayesian nets [19], classification and regression trees (CART) [5], Multivariate Adaptive Regression Splines (MARS) [7], Support Vector Machines [23], ANOVA splines [24], and additive models [18, 17].

All approaches are able to characterise a large class of behaviours and involve training or fitting the model to the dataset. For example, one may wish to predict the vegetation cover of a particular region based on cartographic measurements such as elevation, slope, distance to water, *etc.* [3, 10, 22]. Other examples are prediction of the likelihood of a car insurance customer making a claim, a business customer to purchase a product, or a resident to commit taxation fraud [1].

For a given response variable y and predictor variables $\underline{x} = x_1, \dots, x_d$ a predictive model is described by a function $y = f(x_1, \dots, x_d) = f(\underline{x})$. We will only consider the case where the function f is an element of a linear space and, in the following, we will discuss methods to compute its representation from data.

In the next section we describe the sparse grid approach for predictive modelling using a penalised least squares approach. In Section 3 two solution strategies are presented. The first uses the combination of the partial solutions in the collection of grids—this is the so-called combination technique. The other method uses the combination technique as a preconditioner for the solution of the sparse grid problem in the hierarchical basis. We then discuss the parallelisation of these solution strategies using both coarse and fine grain approaches and their combination to further reduce computation time.

2. Sparse grids for predictive modelling

Recently a technique called sparse grids [12, 25], based on a hierarchical basis approach, has generated considerable interest as a vehicle for reducing dimensionality problems where approximations of high dimensional functions are sought. Sparse

https://doi.org/10.1017/S1446181100003382 Published online by Cambridge University Press

grid functions $f(\underline{x}) \in V$ are approximations which can be seen, in a generalised formulation, as additive models of the form

$$f(\underline{x}) = \sum_{\alpha} c_\alpha f_\alpha(\underline{x}), \tag{2.1}$$

where the partial functions $f_\alpha \in V_\alpha$ are simpler than f in some sense, that is, they are from smaller function spaces $V_\alpha \subset V$. Typically, the partial functions only depend on a subset of the variables or the dependence has a coarser scale as discussed below.

Sparse grids for the solution of partial differential equations, numerical integration and approximation problems have been studied for more than a decade by Zenger, Griebel *et al.* (see [6] for an overview article). They also developed an algorithm known as the combination technique [16] prescribing how the collection of standard grids can be combined to approximate the high dimensional function. More recently, Garcke, Griebel and Thess [10, 11] demonstrated the feasibility of sparse grids in data mining by using the combination technique in predictive modelling.

Additive models of the form of Equation (2.1) generalise linear models and thus form a core technique in nonparametric regression. They include the Multivariate Adaptive Regression Splines (MARS) [7], and the Additive Models by Hastie and Tibshirani [17, 18].

Challenges include the selection of function spaces and the determination of the subset of variables that a partial function depends on. Here we will discuss algorithms for the determination of the function f and hence the partial functions f_α given observed function values when the function spaces are known. For observed data points $\underline{x}^1, \dots, \underline{x}^n$ and function values $y^1, \dots, y^n$ we define the function f from some finite dimensional function space V to be the solution of a penalised least squares problem of the form

$$J(f) = \frac{1}{n} \sum_{i=1}^{n} \left(f(\underline{x}^{(i)}) - y^i\right)^2 + \beta \, \|Lf\|^2 \tag{2.2}$$

for some (differential) operator L. Typical examples are $L = \nabla$ or $L = \Delta$. The parameter β can be chosen according to cross-validation techniques, see for example [24]. If the partial functions f_α of the additive model (2.1) are known to be orthogonal with respect to the corresponding norm (here the standard 2-norm), they can be computed independently as minima of J. In a slightly more general case, which is considered for the combination technique, the projections into the spaces of the partial functions are assumed to commute. In this case the partial functions f_α in (2.1) are known to be multiples of the projections g_α into the partial spaces, that is, the solutions of the minimisation problem (2.2) for the space V_α. The factors are known (integer) coefficients, the combination coefficients c_α [11, 16], and thus $f = \sum_\alpha c_\alpha g_\alpha$.

https://doi.org/10.1017/S1446181100003382 Published online by Cambridge University Press

Note that these approximations can break down when the projections do not commute. As a generalisation of this approach, an approximation has been suggested in [21] where the partial functions are again multiples of the projections g_α but the coefficients are now determined by minimising the functional J. In this case we obtain

$$f = \sum_\alpha \gamma_\alpha g_\alpha.$$

This also generalises the approximations obtained from the additive Schwarz method which is

$$f = \gamma \sum_\alpha g_\alpha,$$

and experimental evidence shows that the performance is in many cases close to that of the multiplicative Schwarz method, which in statistics is known under the term of backfitting [18]. The approaches above can be further improved by iterative refinement. Note that in the following we assume that the projections commute and use the normal combination coefficients c_α from [11, 16].

The interesting aspect of these problems which we will discuss here is the opportunity for parallel processing and the trade-off between parallel processing and the performance of the solvers. We use a two-level iterative solver and parallelism is exploited at both levels.

2.1. Multiresolution analysis and sparse grids The sparse grid idea stems from a hierarchical subspace splitting [25]. Consider the discrete function space $V_{\underline{l}}$, with $\underline{l} = (l_1, \dots, l_d) \in \mathbb{N}_0^d$, of piecewise d-linear functions which is spanned by the usual d-linear 'hat' functions

$$\varphi_{\underline{l},\underline{j}}(\underline{x}) := \prod_{t=1}^{d} \varphi_{l_t,j_t}(x_t), \qquad j_t = 0, \dots, 2^{l_t}.$$

Here, the one-dimensional functions $\varphi_{l,j}(x)$ are

$$\varphi_{l,j}(x) = \begin{cases} 1 - |2^l \cdot x - j|, & x \in [(j-1)/2^l, (j+1)/2^l]; \\ 0, & \text{otherwise.} \end{cases}$$

The number of basis functions needed to resolve any $f \in V_l := V_{l,\dots,l}$ is now larger than 2^{ld}. With a resolution of just 17 points in each dimension ($l = 4$), say, a ten-dimensional problem would require computation and storage of about 2×10^{12} coefficients which is more than one can expect on computers available today. We therefore encounter the curse of dimensionality.

Now we define the difference spaces $W_{\underline{l}}$, with $\underline{e}_t$ denoting the t-th unit vector,

$$W_{\underline{l}} := V_{\underline{l}} \setminus \bigoplus_{t=1}^{d} V_{\underline{l}-\underline{e}_t}.$$

https://doi.org/10.1017/S1446181100003382 Published online by Cambridge University Press

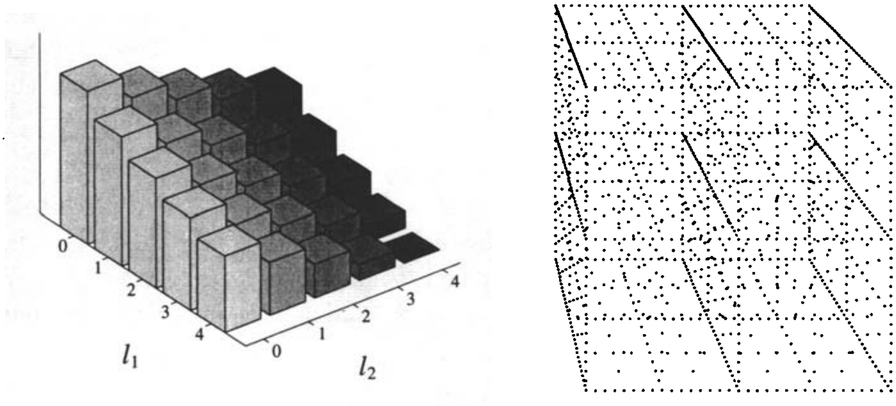

FIGURE 1. Left: Norms of errors of the difference spaces $W_{\underline{l}}$ on a logarithmic scale. The example function is $u(x_1, x_2) = e^{-(x_1^2+x_2^2)}$ with $(x_1, x_2) \in [0, 1] \times [0, 1]$. Right: Sparse grid of refinement level $l = 5$ in three dimensions.

These hierarchical difference spaces lead to the definition of a multilevel subspace splitting, that is, the definition of the space V_l as a direct sum of subspaces,

$$V_l := \bigoplus_{l_1=0}^{l} \cdots \bigoplus_{l_d=0}^{l} W_{\underline{l}} = \bigoplus_{|\underline{l}|_\infty \le l} W_{\underline{l}}. \tag{2.3}$$

Figure 1, showing on a logarithmic scale the norms of the errors for the reconstruction of a two-dimensional sufficiently smooth function, indicates that spaces $W_{\underline{l}}$ with large $|\underline{l}|_1 := l_1 + \cdots + l_d$ contribute very little. In fact, it can be shown [6, 22, 25] that the size of the error committed by removing the space $W_{\underline{l}}$ is proportional to $2^{-r|\underline{l}|_1}$, where $r = 2$ in the case of piecewise linear functions. This suggests removing all spaces where the sum of resolutions is 'large'. The choice of $|\underline{l}|_1 \le l$ in (2.3) results in the sparse grid of [25] (see Figure 1 for an example in three dimensions with $l = 5$) but the grids can be chosen more generally.

Sparse grid spaces can also be achieved with the so-called combination technique [16] through the combination of certain spaces $V_{\underline{l}}$ instead of the difference spaces $W_{\underline{l}}$. In particular the spaces with

$$|\underline{l}|_1 = l_1 + \cdots + l_d = n - q, \quad q = 0, \ldots, d-1, \quad l_t \ge 0$$

are used for a sparse grid of level n. Note that the formula for the combination technique is here

$$f_n^c(\underline{x}) := \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{|\underline{l}|_1 = n-q} f_{\underline{l}}(\underline{x}). \tag{2.4}$$

https://doi.org/10.1017/S1446181100003382 Published online by Cambridge University Press

As mentioned the grids can be chosen more generally, so we will now let the indices belong to an unspecified index set I, which leads to the generalised sparse grid space

$$S_I^d = \bigoplus_{\underline{l} \in I} V_{\underline{l}}. \tag{2.5}$$

Each term in this sum is a tensor product of one-dimensional spaces spanned by hat functions, but they are now restricted by the index set I and will generally have a much lower complexity. We will call the grids belonging to I the *collection of grids.*

See [6, 11, 16, 21, 22] for further details and additional references relevant to this subsection.

2.2. Penalised least squares on sparse grids To compute the partial functions $f_\alpha = f_{\underline{l}} \in V_{\underline{l}}$ on each grid, the functional $J(f)$ in Equation (2.2) has to be minimised. Substituting the representation of $f_{\underline{l}} = \sum_{i=1}^m \alpha_i \varphi_i$, with $\{\varphi_i\}_{i=1}^m$ a basis of $V_{\underline{l}}$, into (2.2) and differentiation with respect to α_i results in the linear system, in matrix notation,

$$(B^T B + \lambda C)\alpha = B^T y. \tag{2.6}$$

Here C is a symmetric $m \times m$ matrix with entries $C_{j,k} = n\,(L\varphi_j, L\varphi_k)_{L_2}$, where $(\cdot, \cdot)_{L_2}$ denotes the standard scalar product in L_2. Note that we use $L = \nabla$ as in [10, 11]. B is a rectangular $n \times m$ matrix with entries $B_{i,j} = \varphi_j(\underline{x}^{(i)})$; $i = 1, \ldots, n$ and $j = 1, \ldots, m$. Since $n \gg m$ one stores $B^T B$ and not B, this also allows us to use only one matrix structure for both C and $B^T B$. The vector $B^T y$ is computed once and then stored for the iterative solution process. We determine the regularisation parameter β with cross-validation [24]. See [10, 11] for further details.

3. Solution of the penalised least squares system

For the solution of the penalised least squares problem (2.2) one can use a hierarchical sparse grid basis and solve the corresponding linear system of equations (2.6) by preconditioned conjugate gradient methods. The matrix $B^T B + \lambda C$ in this case is typically positive definite, often ill-conditioned and has a complex, relatively dense nonzero structure. Although the operation of the matrix C can be implemented in a number of operations proportional to the size of the vector by use of the *unidirectional principle*, this is quite a challenging part of sparse grid implementation for more general operators, see [6] for detailed references in this regard. Furthermore, applying the unidirectional principle for the computation of $B^T B$ scales with the number of data points since it is an on-the-fly-computation, which has to be avoided for complexity reasons [10, 11].

https://doi.org/10.1017/S1446181100003382 Published online by Cambridge University Press

An alternative approach, which is based on the *combination technique*, does not have these problems [9, 10, 11]. In this approach the sparse grid is represented as the union of regular grids, and, correspondingly, the sparse grid space is equal to the sum of the component grid spaces, see Equation (2.5). The combination technique approach comes at the cost of some overhead, as typically the component spaces include some redundancy. In the combination technique the functional $J(f)$ of (2.2) is minimised for all component spaces $V_{\underline{l}}$. The optimum of $J(f)$ over the sparse grid is then approximated by a linear combination of the partial solutions:

$$f(\underline{x}) = \sum_{\underline{l} \in I} c_{\underline{l}} f_{\underline{l}}(\underline{x}),$$

where the combination coefficients $c_{\underline{l}}$ are independent of the data and are given for the case of a regular sparse grid in (2.4). For the generalised sparse grid space these can be found, for example, in [8, 20].

Consider the coefficients α of f with respect to a sparse grid basis. Let $E_{\underline{l}}$ be be the interpolation matrix mapping the coefficients in the subspaces to the coefficients of the sparse grid space. With $B_{\underline{l}} = BE_{\underline{l}}$ and $C_{\underline{l}} = E_{\underline{l}}^T C E_{\underline{l}}$ one gets

$$\alpha = \sum_{\underline{l} \in I} c_{\underline{l}} E_{\underline{l}} \left(B_{\underline{l}}^T B_{\underline{l}} + \lambda C_{\underline{l}} \right)^{-1} E_{\underline{l}}^T B^T y.$$

It follows that the combination technique approximates the inverse $(B^T B + \lambda C)^{-1}$ by the sum $\sum_{\underline{l} \in I} c_{\underline{l}} E_{\underline{l}} \left(B_{\underline{l}}^T B_{\underline{l}} + \lambda C_{\underline{l}} \right)^{-1} E_{\underline{l}}^T$.

The application of the combination method has given good results in practice and it is known that the method gives exact results under certain circumstances [20]. An important property of the sparse grid combination technique is that the solutions on the partial grids can be computed in parallel. We will discuss the consequences of this observation in the next section.

4. Strategies for exploiting parallelism

In this section we describe how the combination technique can be parallelised. The following is valid for both solution strategies of the last section. For ease of presentation we will focus on the standard combination technique.

In general, coarse grain parallelism is preferred, in particular for the application of distributed memory computing [4] as it typically has less (communication) overhead. However, in many cases the amount of available coarse grain parallelism of the algorithm is limited. If one would like to further parallelise the computations (if sufficient parallel resources are available) one would need to look at utilising fine

https://doi.org/10.1017/S1446181100003382 Published online by Cambridge University Press

grain parallelism as well. Such fine grain parallelism is also well suited to some shared memory and vector computations, in which case it can be competitive with coarse grain parallelism.

The combination technique can be straightforwardly parallelised on a coarse grain level [13]. A second level of parallelisation on a fine grain level for each problem in the collection of grids can be achieved through the use of threads on shared-memory multi-processor machines. Both parallelisation strategies, that is, the direct coarse grain parallel treatment of the different grids and the fine grain approach via threads, can also be combined and used simultaneously. This leads to a parallel method which is well suited for a cluster of multi-processor machines. See [9] for first results concerning speedups and efficiency.

Another aspect concerns the situation when the partial problems of the combination technique already need so much main memory that only one can be treated on a multi-processor machine. This situation can arise for a high number of dimensions or massive data sets. In this case the fine grain approach in addition to the coarse grain one still allows an efficient parallel treatment of the combination technique on such systems.

4.1. Parallelisation across grids The linear systems (2.6) for the partial functions f_α of the collection of grids can be computed independently, therefore their approximate solution in each outer iteration step can easily be done completely in parallel. Each process computes the solution on a certain number of grids. If in the extreme case as many processors are available as there are grids in the collection of grids then each processor computes the solution for only one grid. The control process collects the results and computes the final function f on the sparse grid. Just a short setup or gather phase, respectively, is necessary. Since the cost of computation is roughly known *a priori*, a simple but effective static load balancing strategy is available; see [15]. This strategy proceeds by first allocating the large grids and then the smaller ones.

Note that for large data sets the dominant factor in the calculation time for each grid is by far the processing of the data points to compute the matrix $B^T B$ in (2.6) and not the solution of the linear equation system, see [10, 11]. This reduces the burden on the load balancing strategy for the distribution of the grids involved in the combination technique since different grid sizes do not result in significantly different run times.

4.2. Parallelisation across data To compute $B^T B$ in (2.6) for each data instance $\underline{x}^{(i)}$, the product $\varphi_j(\underline{x}^{(i)}) \cdot \varphi_k(\underline{x}^{(i)})$ of the values of all basis functions which are non-zero at $\underline{x}^{(i)}$ has to be calculated and the results have to be written into the matrix structure

https://doi.org/10.1017/S1446181100003382 Published online by Cambridge University Press

at $(B^T B)_{j,k}$, that is,

$$(B^T B)_{j,k} = \sum_{i \le n} \varphi_j(\underline{x}^{(i)}) \cdot \varphi_k(\underline{x}^{(i)}).$$

These computations only depend on one data point at a time and therefore can be done independently for all instances. Therefore the $d \times n$ array of the training set can be separated into p parts, where p is the number of processors available in the shared-memory environment. Each processor now computes the matrix entries for n/p instances. Some overhead is introduced to avoid memory conflicts when writing into the matrix structure. In a similar way the evaluation of the sparse grid function describing the classifier on the data points can be threaded in the evaluation phase.

4.3. Parallelisation of solvers After the matrix is built, threading can be used on SMP architectures in the solution phase as fine grain parallelism. We are using on each partial grid an iterative solver with diagonal preconditioning, therefore most of the computing time is spent on matrix-vector multiplication of the form $\alpha = A\beta$, where A is stored in a sparse-matrix structure. We use the simple approach of a stripe decomposition where the vector α of size m is virtually split into p parts and each processor now computes the action of the matrix for a vector of size m/p. In this way each vector component is only changed by one thread and therefore the storing of the intermediate results into the matrix structure during the computation can be done unprotected, that is, without the need for read/write blocking, see Figure 2.

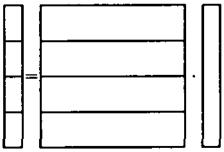

FIGURE 2. Splitting of the matrix and vector in p (here $p = 4$) parts for the matrix-vector-multiplication

Practical experience has shown that diagonal preconditioning is typically good enough for the penalised least squares problems considered. This is in large part due to the observation that for large data sets most time is consumed in the data processing phase, so that more sophisticated preconditioners, or parallel strategies in the solution step, would not result in significant run time improvements.

4.4. Combination of coarse and fine grain parallelism We have seen in [9] that coarse grain parallelism yields the highest speedups and better efficiency. However, the number of grids may be such that the parallel resources are not fully utilised. Here we show how much additional speedup can be expected when using fine grain parallelism in addition to coarse grain parallelism.

https://doi.org/10.1017/S1446181100003382 Published online by Cambridge University Press

If p processors are used to parallelise a computation with k grids one can expect a maximal speedup of $k/\lceil k/p \rceil$. This is displayed for the case of $p = 30$ and $k = 1, \dots, 200$ in Figure 3. In order to utilise the fine grain parallelism we use the coarse grain approach for a first stage where $p\lfloor k/p \rfloor$ grids are processed and then, in a second stage the processors are distributed evenly among the remaining tasks. After the first step there are $p_x = k - p\lfloor k/p \rfloor < p$ tasks remaining. Thus one has $p_l = \lfloor p/p_x \rfloor$ processors per remaining partial grid. These are then used to parallelise on the fine grain scale all the remaining tasks and one thus gets a total combined speedup (that is, for $p_x > 0$) of

$$S_{p,k} = \frac{k}{\lfloor k/p \rfloor + (0.1 + 0.9/p_l)}.$$

This is again displayed in Figure 3. We assume here that the fine grain parallelism has ten percent overhead which cannot be parallelised. Note that in [9] a five percent overhead was observed.

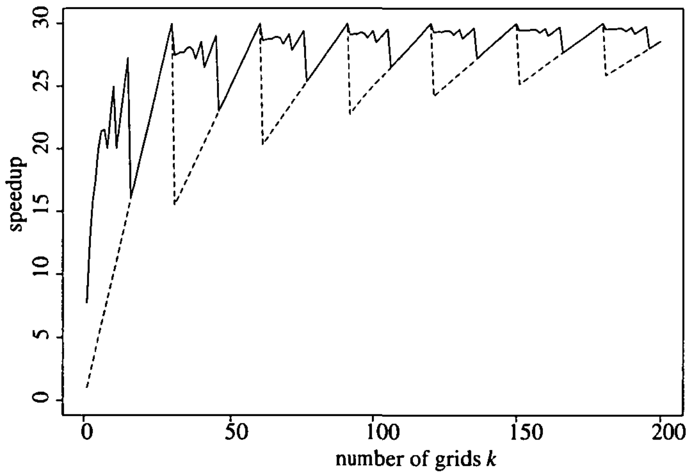

FIGURE 3. Theoretical speedups for coarse grain parallelism (dashed line) and coarse grain with added fine grain (bold line).

This approach can be refined further through the concurrent use of fine and coarse grain parallelism, for example, using $p/2$ grids in the parallelism across grids and a 2-processor shared memory parallelism for each of these grids. This way the drops in the speedup still observed in Figure 3 for the coarse grain with added fine grain parallelism can be reduced further.

https://doi.org/10.1017/S1446181100003382 Published online by Cambridge University Press

5. Conclusion

In this paper we extend results from [9], where two parallelisation strategies for the sparse grid combination technique were shown. Through the combination of both the fine and the coarse grain strategies the speedup results can be further improved. This leads to a parallel method which is well suited for a cluster of multi-processor machines.

Note that besides using the combination technique as an approximation method it also can be employed as a preconditioner for the solution of the sparse grid problem in the hierarchical basis. For difficult problems, one can get an improvement with this ansatz over the one using the combination technique directly for the discretisation. For the case of boundary value problems, this approach has been discussed in [14]. For this ansatz the presented combined parallelisation approaches can be used as well. Additional difficulties arise here since the operator has to be evaluated in the iterative procedure in the hierarchical sparse grids basis as well. In the presented case of data fitting the operator can be split into the data part $B^T B$, which only needs function evaluations, and the stiffness matrix. The latter can be efficiently computed using the combination grids with the approach presented in [12]. This way one can avoid building the needed matrices in the hierarchical basis and can compute the effect of the operator on a sparse grid function by employing (again in parallel) the combination grids.

Acknowledgements

Part of the work was supported by the German Bundesministerium für Bildung und Forschung (BMBF) within the project 03GRM6BN.

References

[1] M. J. A. Berry and G. S. Linoff, *Mastering Data Mining* (Wiley, New York, 2000).

[2] C. M. Bishop, *Neural Networks for Pattern Recognition* (Oxford University Press, Oxford UK, 1995).

[3] J. A. Blackard, "Comparison of neural networks and discriminant analysis in predicting forest cover types", Ph. D. Thesis, Department of Forest Sciences. Colorado State University, Fort Collins, Colorado, 1998.

[4] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker and R. C. Whaley, *ScaLAPACK Users' Guide* (Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997).

[5] L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification and Regression Trees*, Statistics/Probability Series (Wadsworth Publishing Company, Belmont, California, U.S.A., 1984).

https://doi.org/10.1017/S1446181100003382 Published online by Cambridge University Press

[6] H.-J. Bungartz and M. Griebel, "Sparse grids", *Acta Numer.* **13** (2004) 1–123.

[7] J. H. Friedman, "Multivariate adaptive regression splines", *Ann. Statist.* **19** (1) (1991) 1–141, With discussion and a rejoinder by the author.

[8] J. Garcke, "Maschinelles Lernen durch Funktionsrekonstruktion mit verallgemeinerten dünnen Gittern", Ph. D. Thesis, Institut für Numerische Simulation, Universität Bonn, 2004.

[9] J. Garcke and M. Griebel, "On the parallelization of the sparse grid approach for data mining", in *Large-Scale Scientific Computations, Third International Conference, Sozopol, Bulgaria* (eds. S. Margenov, J. Wasniewski and P. Yalamov), Lecture Notes in Computer Science 2179, (Springer, Berlin, 2001), 22–32.

[10] J. Garcke and M. Griebel, "Classification with sparse grids using simplicial basis functions", *Intell. Data Anal.* **6** (6) (2002) 483–502, (shortened version appeared in KDD 2001, Proc. Seventh ACM SIGKDD, F. Provost and R. Srikant (eds.), pages 87–96, ACM, 2001).

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

[12] M. Griebel, "A parallelizable and vectorizable multi-level algorithm on sparse grids", in *Parallel algorithms for partial differential equations (Kiel, 1990)* (ed. W. Hackbusch), Notes Numer. Fluid Mech. 31, (Vieweg, Braunschweig, 1991) 94–100.

[13] M. Griebel, "The combination technique for the sparse grid solution of PDEs on multiprocessor machines", *Par. Proc. Lett.* **2** (1992) 61–70.

[14] M. Griebel, "A domain decomposition method using sparse grids", in *Domain decomposition methods in science and engineering (Como, 1992)*, Contemp. Math. 157 (American Mathematical Society, Providence, RI, 1994) 255–261.

[15] 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* (eds. A. Bode and M. Dal Cin), Lecture Notes in Computer Science 732, (Springer, Berlin, 1993), 276–291.

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

[17] T. Hastie and R. Tibshirani, "Generalized additive models", *Statist. Sci.* **1** (1986) 297–318, With discussion.

[18] T. J. Hastie and R. J. Tibshirani, *Generalized additive models*, Monographs on Statistics and Applied Probability 43 (Chapman and Hall Ltd., London, 1990).

[19] D. Heckerman, "A tutorial on learning with Bayesian networks", in *Learning in graphical models* (ed. M. I. Jordan), (Kluwer, Dordrecht, Netherlands, 1998).

[20] M. Hegland, "Adaptive Sparse Grids", *ANZIAM J.* **44** (E) (2003) C335–C353.

[21] M. Hegland, "Additive sparse grid fitting", in *Curve and surface fitting (Saint-Malo, 2002)*, Mod. Methods Math., (Nashboro Press, Brentwood, TN, 2003) 209–218.

[22] M. Hegland, O. M. Nielsen and Z. Shen, "Multidimensional smoothing using hyperbolic interpolatory wavelets", *Electronic Trans. Numer. Anal.* **17** (2004) 168–180.

[23] V. N. Vapnik, *The Nature of Statistical Learning Theory*, second ed. (Springer, New York, 2000).

[24] G. Wahba, *Spline models for observational data*, CBMS-NSF Regional Conference Series in Applied Mathematics 59 (Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1990).

[25] C. Zenger, "Sparse grids", in *Parallel Algorithms for Partial Differential Equations, Proceedings of the Sixth GAMM-Seminar, Kiel, 1990* (ed. W. Hackbusch), Notes on Num. Fluid Mech. 31 (Vieweg, Braunschweig, 1991) 241–251.

https://doi.org/10.1017/S1446181100003382 Published online by Cambridge University Press