On the parallelization of the sparse grid approach for data mining

Jochen Garcke and Michael Griebel

Institut für Angewandte Mathematik
Abteilung für wissenschaftliches Rechnen und numerische Simulation
Rheinische Friedrich-Wilhelms-Universität Bonn
D-53115 Bonn
{garckej, griebel}@iam.uni-bonn.de

Abstract. Recently we presented a new approach [5, 6] to the classification problem arising in data mining. It is based on the regularization network approach, but in contrast to other methods which employ ansatz functions associated to data points, we use basis functions coming from a grid in the usually high-dimensional feature space for the minimization process. Here, to cope with the curse of dimensionality, we employ so-called sparse grids. To be precise we use the sparse grid combination technique [11] 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 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. The computation on each grid of the sequence of grids is independent of each other and therefore can be done in parallel already on a coarse grain level. A second level of parallelization on a fine grain level can be introduced on each grid through the use of threading on shared-memory multi-processor computers.
We describe the sparse grid combination technique for the classification problem, we discuss the two ways of parallelisation, and we report on the results on a 10 dimensional data set.

AMS subject classification. 62H30, 65D10, 68Q22, 68T10

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

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 [2, 4].

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.

In [5, 6] we presented a new approach to the classification problem. It is based on the regularization network approach but, in contrast to other classification methods which employ mostly global ansatz functions associated to data points, we use an independent grid with associated local ansatz functions in the minimization process. This is similar to the numerical treatment of partial differential equations.

Here, a uniform grid would result in $O(h_n^{-d})$ grid points, where d denotes the dimension of the feature space 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. However, there is the so-called sparse grid 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 [1, 3, 11, 15]. 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.

To be precise, we apply the sparse grid combination technique [11] to the classification problem. For that the regularization network problem is discretized and solved on a certain sequence of conventional grids with uniform mesh sizes in each coordinate direction. The sparse grid solution is then obtained from the solutions on these different grids by linear combination. Thus the classifier is build on sparse grid points and not on data points. A discussion of the complexity of the method gives that the method scales only linearly with the amount of data to be classified. The method is well suited for data mining applications where the dimension of the feature space is moderately high after some preprocessing steps but the amount of data is very large. In [5, 6] we showed that the new method achieves correctness rates which are competitive to that of the best existing methods.

In this paper we describe how the combination method is parallelized in a natural and straightforward way on a coarse grain level. A second level of parallelization on a fine grain level through the use of threading on shared-memory multi-processor machines is also discussed.

The remainder of this paper is organised as follows: In Section 2 we describe the classification problem in the framework of regularization networks as minimization of a (quadratic) functional. We then discretize the feature space and derive the associated linear problem. Here we focus on grid-based discretization techniques. Then, we introduce the sparse grid combination technique for the classification problem and discuss its properties. Section 3 presents the results

of numerical experiments conducted with the sparse grid combination method. Some final remarks conclude the paper.

2 The problem

Classification of data can be interpreted as traditional scattered data approximation problem with certain additional regularization terms. In contrast to conventional scattered data approximation applications, we now encounter quite high-dimensional spaces. To this end, the approach of regularization networks [8] 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 [7].

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 the 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 [13, 14] 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.

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 $\{\psi_j\}_{j=1}^N$ 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. In the remainder of this paper, we restrict ourselves to the choice

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

and

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

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

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

in the finite dimensional space V_N. We plug (2) into (4) and obtain after differentiation with respect to α_k, $k = 1, \ldots, N$, see [6],

$$\sum_{j=1}^{N} \alpha_j \left[M\lambda (P\varphi_j, P\varphi_k)_{L_2} + \sum_{i=1}^{M} \varphi_j(\boldsymbol{x}_i) \cdot \varphi_k(\boldsymbol{x}_i) \right] = \sum_{i=1}^{M} y_i \varphi_k(\boldsymbol{x}_i). \tag{5}$$

In matrix notation we end up with the linear system

$$(\lambda C + B \cdot B^T)\alpha = By. \tag{6}$$

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

2.2 Grid based discrete approximation

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

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

test- and trial-functions $\phi_{n,\mathbf{j}}(\boldsymbol{x})$ on grid Ω_n now would give the classifier $f_N(\boldsymbol{x}) = f_n(\boldsymbol{x})$ as

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

and the variational procedure (4) - (5) would result in the discrete linear system

$$(\lambda C_n + B_n \cdot B_n^T)\alpha_n = B_n y \tag{7}$$

of size $(2^n + 1)^d$ and matrix entries corresponding to (6). Note that f_n lives in the space

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

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 $\Omega_\mathbf{l}$, $\mathbf{l} = (l_1, ..., l_d) \in \mathbb{N}^d$, 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

$$l_1 + ... + l_d = n + (d - 1) - q, \quad q = 0, .., d - 1, \quad l_t > 0. \tag{8}$$

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

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

and the variational procedure (4) - (5) would result in the discrete system

$$(\lambda C_\mathbf{l} + B_\mathbf{l} \cdot B_\mathbf{l}^T)\alpha_\mathbf{l} = B_\mathbf{l} y \tag{9}$$

with the matrices

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

$j_t, k_t = 0, ..., 2^{l_t}, t = 1, ..., d, i = 1, ..., M$, and the unknown vector $(\alpha_\mathbf{l})_\mathbf{j}$, $j_t = 0, ..., 2^{l_t}, t = 1, ..., d$. We then solve these problems by a feasible method. To

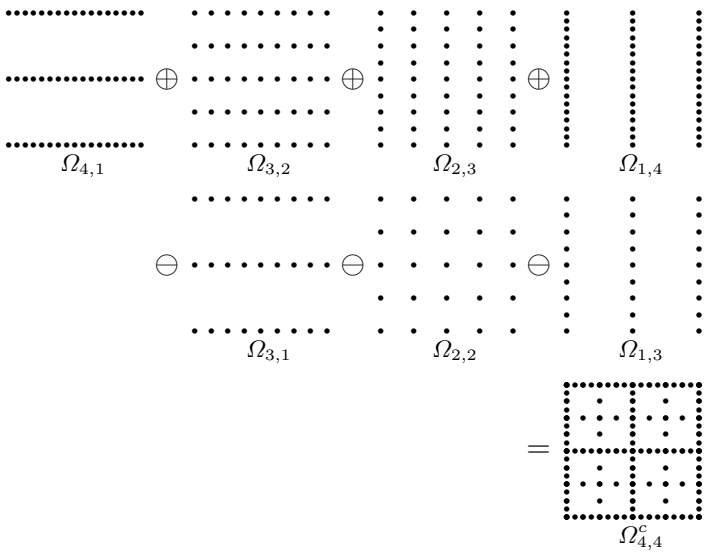

Fig. 1. Combination technique on level 4, $d = 2, q = 4$

this end we use here a diagonally preconditioned conjugate gradient algorithm. But also an appropriate multi-grid method with partial semi-coarsening can be applied. The discrete solutions $f_{\mathbf{l}}$ belong to the spaces

$$V_{\mathbf{l}} \;:=\; \text{span}\{\phi_{\mathbf{l},\mathbf{j}}, j_t = 0, ..., 2^{l_t}, t = 1, ..., d\}, \tag{10}$$

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

Note that all these problems are substantially reduced in size in comparison to (7). 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 $\dim(V_{\mathbf{l}}) = O(h_n^{-1}) = O(2^n)$.

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

$$f_n^{(c)}(\boldsymbol{x}) := \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{l_1+...+l_d = n+(d-1)-q} f_{\mathbf{l}}(\boldsymbol{x}). \tag{11}$$

The resulting function $f_n^{(c)}$ lives in the 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 sparse grid 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 [3].

Note that 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 (11) acting directly on the functions $f_{\mathbf{l}}$, i.e.

$$\mathrm{F}(f_n^{(c)}) = \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{l_1+\ldots+l_d=n+(d-1)-q} \mathrm{F}(f_{\mathbf{l}}). \tag{12}$$

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

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

we just form the combination of the associated values for $f_{\mathbf{l}}$ according to (11). 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.

So far we only used d-linear basis functions based on a tensor-product approach, this case was presented in detail in [6]. Note that also linear ansatz functions based on a simplicial discretization can be applied on the grids of the combination technique; this variant was introduced in [5]. The complexities of the simplicial version 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 (7), see [5]. 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 $d \leq 8$ for d-linear basis functions and $d \leq 10$ for linear basis functions. A decomposition of the matrix entries over several computers in a parallel environment would permit more dimensions.

2.4 Parallelization

The combination technique is straightforwardly parallel on a coarse grain level [9]. The partial classifiers $f_{\mathbf{l}}$, i.e. $\alpha_{\mathbf{l}}$ in the discrete system (9), in the sequence of grids (8) can be computed independently of each other, therefore their computation can be done completely in parallel. Each process computes the solution on a certain number of grids. If as many processors are available as there are grids in the sequence of grids (8) then each processor computes the solution for only one grid. The control process collects the results and computes the final classifier $f_n^{(c)}$ on the sparse grid Ω_n^c. 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 [10].

A second level of parallelization on a fine grain level for each problem (9) in the sequence of grids (8) can be achieved through the use of threads on shared-memory multi-processor machines. This concerns the assembly of the data dependent part of the system matrix, the matrix-vector-multiplication in the iterative solver, and the evaluation phase.

To compute $B_{\mathbf{l}} \cdot B_{\mathbf{l}}^T$ in (9) for each data instance computations have to be made and the results have to be written into the matrix structure. These computations only depend on the data and therefore can be done independently for all instances. Therefore the $d \times M$ array of the training set can be separated in p parts, where p is the number of processors available in the shared-memory environment. Each processor now computes the matrix entries for M/p instances. Some overhead is introduced to avoid memory conflicts when writing into the matrix structure. In a similar way the evaluation of the classifier on the data points can be threaded in the evaluation phase.

After the matrix is built threading can also be used in the solution phase on a fine grain level. Since we are using an iterative solver most of the computing time is used for the matrix-vector-multiplication. Here the vector $\alpha_{\mathbf{l}}$ in (9) of size N can be split into p parts and each processor now computes the action of the matrix on a vector of size N/p.

Both parallelization strategies, i.e. the direct coarse grain parallel treatment of the problems in (8) 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.

3 Numerical results

In [5, 6] we showed that our new method achieves correctness rates which are competitive to that of the best existing methods. Therefore we concentrate here on the effects of the parallelization approaches on the run time.

To measure the performance we produced with DatGen [12] a 10-dimensional test case with 5 million training points and 50 000 points for testing. We used the call datgen -r1 -X0/200,R,O:0/200,R,O:0/200,R,O:0/200,R,O:0/200,R,O:0/200, R,O:0/200,R,O:0/200,R,O:0/200,R,O:0/200,R,O -R2 -C2/6 -D2/7 -T10/60 -p -O5050000 -e0.15. The achieved testing correctness rate for $\lambda = 0.01$ is 97.4 % on level 1, 97.9 % on level 2, and 97.7 % on level 3..

More than 50 % of the run time is spent for the assembly of the data dependent part of the system matrix, i.e. $B_{\mathbf{l}} \cdot B_{\mathbf{l}}^T$ in (9), and the time needed for this matrix part scales linearly with the number of instances [5, 6].

First we look at the results using the natural coarse grain parallelism of the combination technique, i.e. the distribution of the partial problems in the sequence of grids (8) onto different processors. We used a machine of 24 UltraSPARC-III (750MHz) CPUs. The run times for level 2 are shown in Table 1. With 11 processors a speed-up of 9.7 with a parallel efficiency of 0.88 is achieved. Since only 11 grids have to be calculated for level 2 no more than 11 nodes are needed.

In Table 2 we present the run times for the fine grain parallelization on a shared-memory computer with 24 UltraSPARC-III (750MHz) processors. Here we compute all problems from the sequence of grids on one machine sequentially, but use the fine grain parallelization with threads. Overall we achieve acceptable speed-ups from 1.8 for two processors up to 12.3 for 24 processors. As one would expect the efficiency decreases with the number of processors. This is usual for

# processors	time (sec.)	speed-up	efficiency
1	6124	-	-
2	3400	1.80	0.90
3	2311	2.65	0.88
4	1752	3.50	0.87
5	1689	3.62	0.72
6	1207	5.07	0.85
7	1191	5.14	0.73
8	1199	5.11	0.64
9	1188	5.15	0.57
10	1126	5.44	0.54
11	630	9.72	0.88

Table 1. Parallel run time results for a 10D synthetic massive data set using the coarse grain level parallelization of the combination technique

any shared memory system. Note that the speed-up and the efficiency is better for the computation on level 2 since the non-threadable part of the algorithm has less impact on the total run time.

In Table 3 we show the results which can be achieved when both parallelization strategies, i.e. on the coarse and the fine grain level, are used simultaneously. Here we use a cluster of four shared-memory machines, each with 24 UltraSPARC-III (750MHz) CPUs. We give the run times on level 2 for different combinations of the number of processes used for the coarse grain parallelization and of the number of threads used by each of these coarse grain processes. The resulting speed-ups and efficiencies are almost the products of the respective entries from Table 1 and 2.

As a last example we give results for level 3, here 66 grids have to be considered. The computation in the serial version takes 43345 seconds. Using 33 processes for the coarse grain parallelization with two threads each for the fine grain parallelism the run time is 830 seconds, resulting in a speed-up of 52.2 and an efficiency of 0.79. With 66 processes, only used for the coarse grain parallelism, 802 seconds are needed, here the speed-up is 54.1 and the efficiency is 0.82.

4 Conclusions

We presented two parallelization strategies of the sparse grid combination technique for the classification of data. One parallelizes the combination technique on a coarse grain level, the other one uses threads on a fine grain level to parallelize the computation on each grid of the combination technique. A simultaneous use of both approaches is also possible on suitable parallel computers, i.e. a cluster of SMP-machines.

Both variants and their combination resulted in significant speed-ups of the overall algorithm.

	Level 1			Level 2		
# threads	time (sec.)	speed-up	efficiency	time (sec.)	speed-up	efficiency
1	540	-	-	6124	-	-
2	305	1.77	0.89	3363	1.82	0.91
3	223	2.42	0.81	2452	2.50	0.83
4	176	3.07	0.77	1895	3.23	0.81
5	152	3.55	0.71	1596	3.84	0.77
6	134	4.03	0.67	1369	4.47	0.75
7	121	4.46	0.63	1222	5.01	0.72
8	110	4.91	0.61	1084	5.65	0.71
9	102	5.29	0.59	1012	6.05	0.67
10	94	5.74	0.57	919	6.66	0.67
11	90	6.00	0.55	866	7.07	0.64
12	83	6.51	0.54	796	7.69	0.64
13	80	6.75	0.52	759	8.07	0.62
14	77	7.01	0.50	715	8.57	0.61
15	74	7.30	0.49	688	8.90	0.59
16	71	7.61	0.48	647	9.47	0.59
17	69	7.83	0.46	637	9.61	0.57
18	67	8.06	0.45	600	10.21	0.57
19	65	8.31	0.44	593	10.33	0.54
20	64	8.44	0.42	560	10.93	0.55
21	63	8.57	0.41	543	11.28	0.54
22	60	9.00	0.41	522	11.73	0.53
23	58	9.31	0.40	511	11.98	0.52
24	57	9.47	0.39	499	12.27	0.51

Table 2. Run time results for a 10D synthetic massive data set using threads

# processes	# threads	time (sec.)	speed-up	efficiency
1	1	6124	-	-
2	2	1853	3.26	0.82
2	4	1049	5.84	0.73
4	2	978	6.26	0.78
4	4	567	10.80	0.68
4	6	391	15.55	0.65
6	2	684	8.95	0.75
6	3	483	12.68	0.70
6	4	380	16.12	0.67
6	6	277	22.11	0.61
11	2	349	17.55	0.80
11	3	254	24.11	0.73
11	4	204	30.02	0.68
11	6	154	39.77	0.60

Table 3. Run time results for a 10D data set using both parallelization strategies

5 Acknowledgements

Part of the work was supported by the German Bundesministerium für Bildung und Forschung (BMB+F) within the project 03GRM6BN. We would like to thank the Rechenzentrum Ulm for the computing time on their Computeserver of four SUN-machines with 24 UltraSPARC-III (750MHz) CPUs each.

References

1. R. Balder. *Adaptive Verfahren für elliptische und parabolische Differentialgleichungen auf dünnen Gittern*. Dissertation, Technische Universität München, 1994.
2. M. J. A. Berry and G. S. Linoff. *Mastering Data Mining*. Wiley, 2000.
3. H.-J. Bungartz. *Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimensionalen Poisson-Gleichung*. Dissertation, Institut für Informatik, Technische Universität München, 1992.
4. K. Cios, W. Pedrycz, and R. Swiniarski. *Data Mining Methods for Knowledge Discovery*. Kluwer, 1998.
5. J. Garcke and M. Griebel. Data mining with sparse grids using simplicial basis functions. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001. also as SFB 256 Preprint 713, Universität Bonn, 2001.
6. J. Garcke, M. Griebel, and M. Thess. Data mining with sparse grids. *Computing*, 2001. to appear, also as SFB 256 Preprint 675, Institut für Angewandte Mathematik, Universität Bonn, 2000.
7. F. Girosi. An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10(6):1455–1480, 1998.
8. F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7:219–265, 1995.
9. M. Griebel. The combination technique for the sparse grid solution of PDEs on multiprocessor machines. *Parallel Processing Letters*, 2(1):61–70, 1992. also as SFB Bericht 342/14/91 A, Institut für Informatik, TU München, 1991.
10. 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. Dal Cin, editors, *Parallel Computer Architectures: Theory, Hardware, Software, Applications*, volume 732 of *Lecture Notes in Computer Science*, pages 276–291. Springer Verlag, 1993.
11. 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. also as SFB Bericht, 342/19/90 A, Institut für Informatik, TU München, 1990.
12. G. Melli. Datgen: A program that creates structured data. Website. http://www.datasetgenerator.com.
13. A. N. Tikhonov and V. A. Arsenin. *Solutions of ill-posed problems*. W.H. Winston, Washington D.C., 1977.
14. G. Wahba. *Spline models for observational data*, volume 59 of *Series in Applied Mathematics*. SIAM, Philadelphia, 1990.
15. 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.