Adaptive Sparse Grid Techniques for Data Mining

H.-J. Bungartz, D. Pflüger, and S. Zimmer

Department of Informatics, Technische Universität München,
Boltzmannstraße 3, 85748 Garching, Germany
{bungartz, pflueged, zimmer}@in.tum.de

Summary. It was shown in [GaGT01] that the task of classification in data mining can be tackled by employing ansatz functions associated to grid points in the (often high dimensional) feature-space rather than using data-centered ansatz functions. To cope with the curse of dimensionality, sparse grids have been used.

Based on this approach we propose an efficient finite-element-like discretization technique for classification instead of the combination technique used in [GaGT01]. The main goal of our method is to make use of adaptivity to further reduce the number of grid points needed. Employing adaptivity in classification is reasonable as the target function contains smooth regions as well as rough ones. Regarding implementational issues we present an algorithm for the fast multiplication of the vector of unknowns with the coefficient matrix. We give an example for the adaptive selection of grid points and show that special care has to be taken regarding the boundary values, as adaptive techniques commonly used for solving PDEs are not optimal here. Results for some typical classification tasks, including a problem from the UCI repository, are presented.

1 Introduction

Due to technical and scientific progress an ever increasing amount of data can be created, collected and stored. Typical examples are medical datasets or datasets collected in e-commerce or via geological observations as in tsunami warning systems. The availability of vast datasets increases the need for efficient algorithms in the field of data mining that can handle large datasets and extract hidden patterns and retrieve unknown information.

1.1 Data Mining

Our focus is on classification, the machine learning of a two-class problem, which plays an important role within the field of predictive modelling in data mining. Classification algorithms try to find a function that correctly assigns

To Appear in:
Modelling, Simulation and Optimization of Complex Processes, Proc. Int. Conf. HPSC, Hanoi, Vietnam, 2006.

data to one of several given classes. Given is a set of preclassified data, the training dataset

$$S = \{(\mathbf{x}_i, y_i) \in [0,1]^d \times \{-1,1\}\}_{i=1}^{M}.$$

For a finite set of data points the feature space can always be normalized to fit in $[0,1]^d$. Each of the M training data points is assigned to one of the two class labels, $+1$ or -1.

The aim is to compute a classificator or machine learner (ml)

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

based on the set of preclassified training data, which can be used to obtain class predictions applied on previously unseen data points. In generally, f can obtain any arbitrary real number, so a class prediction of greater or equal than zero is usually mapped to the positive class and a prediction of lower than zero to the negative one. The output of f for any given data point can then be considered as a measure for the confidence the ml has in its prediction: The higher the absolute value is, the higher is the confidence that the data point belongs to the corresponding class.

To compute the ml we follow [GaGT01] and minimize the functional

$$H[f] = \frac{1}{M}\sum_{i=1}^{M}(y_i - f(\mathbf{x}_i))^2 + \lambda \|\nabla f\|_{L_2}^2,$$

where $(y_i - f(\mathbf{x}_i))^2$ is a cost or error function ensuring that the target function f is somehow close to the training dataset, $\|\nabla f\|_{L_2}^2$ is the regularization operator or stabilizer incorporating the smoothness assumptions, and λ is the regularization parameter that controls the trade-off between approximation error and smoothness of f.

1.2 Discretization

Common classification algorithms use mostly global ansatz functions associated to data points to reconstruct f, with the aim to reduce the number of ansatz functions needed. The draw-back is that they typically scale quadratic or even worse in the number of training data points. This does not impose a problem for applications where preclassifying data points is very expensive or hardly possible as this results in very small datasets for training anyway.

However, there are many applications with plenty of training data – e.g. in e-commerce or engineering – where the correct classification of data points can be obtained automatically, say via observing the completion of user transactions. The idea therefore is to find a classification algorithm which is independent of the training dataset and which scales only linearly in the number of training data points. Additionally, many classification algorithms are restricted to learning certain kinds of problems. It is known for instance that

To Appear in:
Modelling, Simulation and Optimization of Complex Processes, Proc. Int. Conf. HPSC, Hanoi, Vietnam, 2006.

some Neural Network topologies are incapable of separating complex structures as two intertwined spirals [Sing98], whereas it was shown in [GaGT01] and [Pflü05] that sparse grids can cope with completely different kinds of classification tasks.

A promising approach, followed in [GaGT01], is to discretize the feature space and to use ansatz functions associated to grid points. A suitable basis $\{\phi_i\}_{i=1}^N$ has to be introduced, and the problem is restricted to a finite dimensional space V_N spanned by the basis functions,

$$f_N(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \phi_j(\mathbf{x}).$$

In the following we consider the space of piecewise d-linear functions.

Minimisation of $H[f]$ leads to a linear system with N unknowns

$$\left(\lambda MC + B \cdot B^T\right) \boldsymbol{\alpha} = B\mathbf{y}, \tag{1}$$

with $C_{ij} = (\nabla\phi_i(\mathbf{x}), \nabla\phi_j(\mathbf{x}))_{L_2}$ and $B_{ij} = \phi_i(\mathbf{x}_j)$, which can be solved iteratively, e.g. with the CG method. But one encounters the curse of dimensionality: Using a regular grid with n grid points in one dimension results in n^d grid points for d dimensions. A straightforward discretization of space is therefore infeasible even when dealing with only low dimensional feature spaces and small values of n.

1.3 Sparse Grids

Sparse grids cope with the curse of dimensionality and have been successfully applied in various fields of application [BuGr04]. A sparse grid needs far less grid points – only $\mathcal{O}(N \log(N)^{d-1})$ rather than $\mathcal{O}(N^d)$ – with only slightly deteriorated accuracy.

The underlying principle is a hierarchical formulation of the basis functions. We use the standard hierarchical basis

$$\Phi_l := \left\{\phi_{l',i} : l' \leq l, i \leq 2^{l'} - 1 \wedge i \text{ odd}\right\}.$$

with piecewise linear ansatz functions $\phi_{l,i}(x) := \phi\left(x \cdot 2^l - i\right)$ and $\phi(x) := \max(1-|x|, 0)$. Note that all basis functions on one level have pairwise disjoint supports and cover the whole domain.

The hierarchical basis functions can be extended to d dimensions via a tensor product approach, and are defined as

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

$\mathbf{l}$ and $\mathbf{i}$ are multi indices, indicating level and index of the underlying one-dimensional hat functions for each dimension.

To Appear in:
Modelling, Simulation and Optimization of Complex Processes, Proc. Int. Conf. HPSC, Hanoi, Vietnam, 2006.

The basis

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

span subspaces $W_\mathbf{l}$. Again, all basis functions have pairwise disjoint, equally sized supports and cover the whole domain. The space of d-linear functions $V_n := V_n^\infty$ with mesh width 2^{-n+1} can be written as a sum of subspaces $W_\mathbf{l}$:

$$V_n^{(\infty)} := \sum_{l_1=1}^{n} \cdots \sum_{l_d=1}^{n} W_{(l_1,\ldots,l_d)} = \bigoplus_{|\mathbf{l}|_\infty \leq n} W_\mathbf{l}.$$

The hierarchical basis allows to choose subspaces according to their contribution to the approximation. This can be done by an a priori selection [BuGr04], resulting for example in the space $V_n^{(1)}$:

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

Figure 1 shows the tableau of subspaces in two dimensions for the sparse grid $V_3^{(1)}$.

As an alternative, adaptive methods for selecting grid points have also turned out to be comparatively easily implemented in various applications. In the next two sections we will focus on an efficient way of solving the system of linear equations and on the use of an adaptive creation of the underlying sparse grid.

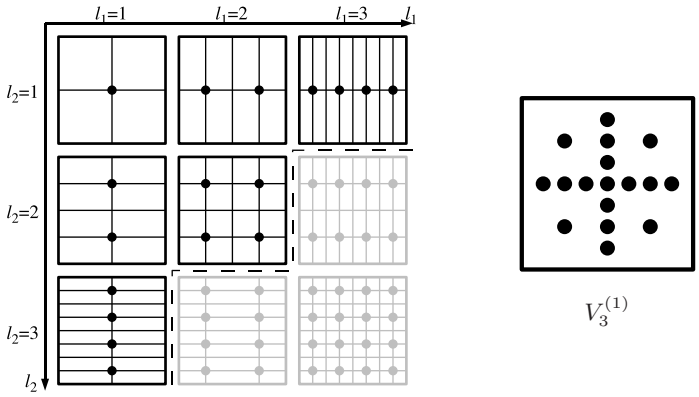

Fig. 1. Scheme of subspaces and sparse grid for level $n = 3$, 2D

2 Solving the System of Linear Equations

The solution of (1) can be approximated using the combination technique as shown in [GaGT01]. The classification problem is solved for multiple, but

To Appear in:
Modelling, Simulation and Optimization of Complex Processes, Proc. Int. Conf. HPSC, Hanoi, Vietnam, 2006.

smaller regular grids. The solution is obtained by a linear combination of the partial solutions for those grids. The main advantage is that one can apply the whole machinery available for regular grids.

Alternatively, the system can be solved for the sparse grid with a direct finite-element-like technique, but the overall system $(\lambda MC + BB^T)$ is everything but sparse. Iterative solvers have to be used, for instance a preconditioned Conjugated Gradient method.

For efficiency reasons, the matrices should not be assembled directly. Only the application of the matrices to a vector should be implemented. The crucial part hereby is the application of the stiffness matrix C. This is known to be algorithmically complicated. With an efficient realization, one iteration scales only linearly in the number of training data points and in the number of grid points, respectively.

Our focus is on the direct finite element technique, as this allows for adaptive grid generation. In the remaining part of this section we will present the main ideas of the application of the matrices and sketch the algorithmic difficulties and how to tackle them.

2.1 Application of B and B^T

The application of the matrices B and B^T can be done in a straightforward way: B^T is a $(M \times N)$-matrix, with M being the number of training points and N the number of grid points (unknowns). It is

$$(B^T \boldsymbol{\alpha})_i = \sum_{j=1}^{N} \phi_j(\mathbf{x}_i)\boldsymbol{\alpha}_j, \quad 1 \leq i \leq M,$$

so for all training data points $\mathbf{x}_i$ one has to descend in the tree of subspaces once. For each subspace $W_{\mathbf{l}}$ there can be only one nonzero basis function for the current data point $\mathbf{x}_i$. It can be identified and evaluated by a constant number of operations.

Fixing the level for all but one dimension and varying the remaining one, the basis functions for $d-1$ dimensions stay the same and do not have to be identified and evaluated multiple times. Additional operations can be saved if this is taken into consideration.

The application of B^T to a vector $\boldsymbol{\alpha}$ can be done in $\mathcal{O}(M \log N)$. The multiplication of the matrix B with a vector $\boldsymbol{\alpha}$ can be done analogously.

2.2 Application of C

For the application of the stiffness matrix C to a vector $\boldsymbol{\alpha}$ we will consider the one-dimensional case first. Let the coefficients $\boldsymbol{\alpha}_i$ be arranged in a binary tree, according to the hierarchy. Then the application of C can be split in a *down*-part (from the root towards the leaves) and in an *up*-part (towards

To Appear in:
Modelling, Simulation and Optimization of Complex Processes, Proc. Int. Conf. HPSC, Hanoi, Vietnam, 2006.

the root), each with one traversal of the binary tree, and therefore in $\mathcal{O}(N)$ operations.

Extending this to two dimensions we have products of up- and down-processes for each dimension due to the tensor product construction of the basis functions. Let up_i be the information propagation upwards in dimension i, down_i downwards. Then we have to apply $(\text{up}_1 + \text{down}_1)(\text{up}_2 + \text{down}_2)$.

This is where the main difficulties arise when we try to handle one dimension after the other. Figure 2 shows the information propagation between the basis functions associated to the grid points labeled with 1 and 2. Whereas the information of 1 can be propagated upwards to the common root node in x_1-direction and downwards in x_2-direction in the second step to node 2, this does not work for $\text{down}_1\ \text{up}_2$ from node 2 to 1. Here three grid points are missing to store the propagated information intermediately.

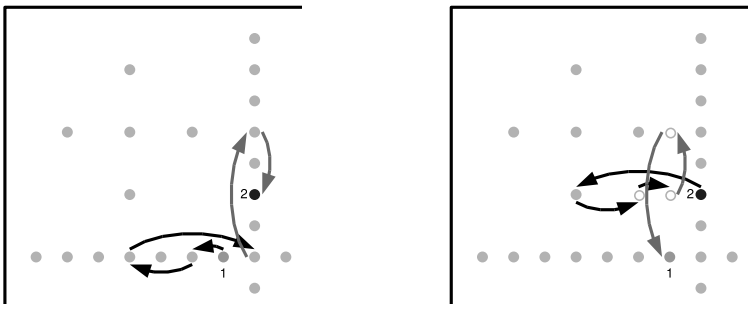

Fig. 2. Flow of information: (left) $\text{up}_1\ \text{down}_2$, $1 \to 2$, (right) $\text{down}_1\ \text{up}_2$, $2 \to 1$

Unfortunately, the missing grid points cannot be created on the fly as this would result in a full regular grid again. Instead, the up- and down-directions have to be reordered so that all propagations upwards are done before the first propagation downwards, regardless of the dimension. For d-dimensional problems, we get a similar structure with up- and down-processes in each direction. This leads to an application of the matrix C to a vector which is, again, linear in the number of grid points. For further details of our implementation, see [Pflü05].

3 Adaptivity

Common classification algorithms use mostly global basis functions centered according to the training dataset and hence depend algorithmically on it. Contrary, discretizing feature space provides the possibility to classify somehow independent of the training data. The advantage of scaling only linearly in the number of training points comes with a higher number of basis functions. The idea of adaptive sparse grid classification is to make use of both

To Appear in:
Modelling, Simulation and Optimization of Complex Processes, Proc. Int. Conf. HPSC, Hanoi, Vietnam, 2006.

worlds. It seems to be promising to aim for an algorithm that scales only linearly regarding the cardinality of the training dataset, but still spends grid points especially in regions where it is most necessary. It is reasonable to apply adaptivity in classification in any case, as the target function contains smooth regions as well as rough ones.

Applying adaptivity, we start with the regular sparse grid on level two, $V_2^{(1)}$. We use a Conjugated Gradient method for a few iterations for training. For the following examples, we use a very simple refinement strategy which we adopted from solving differential equations: Grid points are refined according to the surplus – the contribution or computed coefficient – of the basis functions. But rather than refining all grid points with a surplus exceeding some fixed threshold, it is more useful to refine a certain percentage with the highest surplus out of those grid points that can be refined.

A basic requirement of sparse grids is that for each single basis function in the structure of binary trees all of its possible ancestors exist. This has to be taken into account when creating new grid points. All missing ancestors have to be created recursively up to the root basis function.

In the following, we show some observations for a two-dimensional classification problem first, as it can be visualized in contrast to high-dimensional ones. The benefit of employing adaptivity increases with growing dimensionality. For this we present some results for a real-world medical dataset.

The first example, the so-called Ripley dataset, has been taken from [RiHj95]. The training dataset consists of 250 points. Additionally, a dataset with 1000 points for testing is provided. The Ripley dataset serves as a benchmark for classification algorithms as it is known to contain 8% of noise.

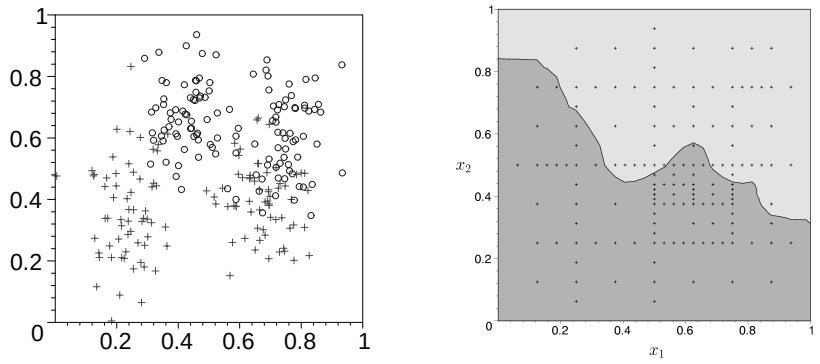

Fig. 3. Ripley dataset: (left) training data, (right) adaptive grid and classification

Figure 3 shows the training dataset and the adaptive sparse grid together with the corresponding classification boundary. We used $\lambda = 0.01$ and refined the top 15% each. It can be seen that most grid points have been spent in the

To Appear in:
Modelling, Simulation and Optimization of Complex Processes, Proc. Int. Conf. HPSC, Hanoi, Vietnam, 2006.

critical region, the region with the most noise. The accuracy on the test set is 90.9% which is a very good result for a dataset containing 8% of noise.

As a higher dimensional example we chose the Bupa liver dataset, obtained from the UCI Repository [NHBM98]. It contains real vital data of 345 patients, mainly blood test values, and describes symptoms for liver disorders. We used the same value for the regularisation parameter again, $\lambda = 0.01$.

Table 1. Comparison of non-adaptive and adaptive sparse grids, Bupa liver dataset

regular sparse grid				adaptive grid refinement			
# grid points	max l	# iterations	acc.	# grid points	max l	# iterations	acc.
1	1	1	58.0				
13	2	3	59.7	13	2	3	59.7
97	3	7	61.4	77	3	8	62.0
545	4	18	66.1	243	4	17	65.5
				655	4	28	71.6
2561	5	35	72.8	**1543**	**5**	**44**	**81.2**
				3981	5	93	86.7
10625	**6**	**65**	**81.7**	8783	6	158	91.9

Table 1 clearly demonstrates the gain of using adaptivity. The left hand side of the table shows the classification for the regular sparse grids for level one to eight, the right hand side for the adaptive sparse grid. There we start with $V_2^{(1)}$ and continue with six times of refinement. The table shows the number of grid points involved, the maximum level of a basis function, the number of iterations of the diagonally preconditioned CG, and the accuracy obtained on the training data.

The first time an accuracy of more than 81% is reached is for level six in the case of the regular sparse grids. This involves 10625 grid points and 65 iterations. Using the adaptive sparse grid is by far better: After only four times of refinement with merely 1543 basis functions and 44 iterations we already get an accuracy better than 81%.

4 Boundary Considerations

For the previous examples we did not use any grid points on the boundary – the functions we learned were all zero on the boundary of the feature space. Nevertheless, we got excellent results. In the remainder of this section we will point out why it is advantageous not to spend extra computational effort on the boundary in classification.

To allow for boundary values apart from zero, additional grid points can be spent on the boundary. But this results in a much larger number of unknowns, especially for higher dimensions. For example, the simple ten dimensional sparse grid $V_2^{(1)}$ consists of 21 points. Employing grid points on the boundary

To Appear in:
Modelling, Simulation and Optimization of Complex Processes, Proc. Int. Conf. HPSC, Hanoi, Vietnam, 2006.

results in 452,709 points instead. To save precious computation time one would be better off without plenty of additional basis functions.

As an alternative, the basis functions can be modified, especially those adjacent to the boundary. On the first level, we use the constant function 1. On all other level we fold up the hat functions next to the boundary. This leads to the following basis functions in 1D, higher dimensional functions are constructed via the tensor product approach again:

$$\phi_{l,i}(x) := \begin{cases} 1 & \text{if } l = 1 \wedge i = 1 \\ \left\{ \begin{array}{ll} 2 - 2^l \cdot x & \text{if } x \in \left[0, \frac{1}{2^{l-1}}\right] \\ 0 & \text{otherwise} \end{array} \right\} & \text{if } l > 1 \wedge i = 1 \\ \left\{ \begin{array}{ll} 2^l \cdot x + 1 - i & \text{if } x \in \left[1 - \frac{1}{2^{l-1}}, 1\right] \\ 0 & \text{otherwise} \end{array} \right\} & \text{if } l > 1 \wedge i = 2^l - 1 \\ \phi\left(x \cdot 2^l - j\right) & \text{otherwise.} \end{cases}$$

Figure 4 (left) shows the sparse grid for level two with additional grid points on the boundary. The common hat basis functions span a space of piecewise linear functions with a mesh width of 2^{-2}. If the modified boundary functions are used, the same function space can be obtained by using additionally the basis functions adjacent to the boundary of the next higher level. This results in the same number of unknowns and in the grid on the right.

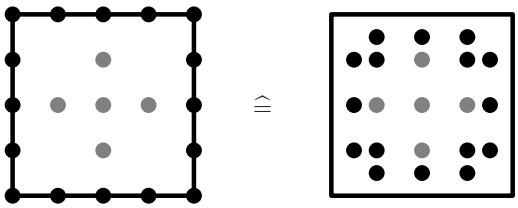

Fig. 4. $V_2^{(1)}$ with (left) common hat basis functions, (right) modified basis functions

Considering the modified basis, adaptivity can take care of the boundary functions: They are automatically created wherever needed. Classifications showed that it is hardly necessary to modify the basis functions if there are no data points located exactly on the boundary as in the case of the Ripley dataset, for example. Table 2 shows that there is almost no difference in the number of grid points created. But as the condition number of the matrix detoriates using the modified boundary functions, it takes more than twice as many iterations to reduce the norm of the residuum below 10^{-8}.

If the datasets are normalized not to the unit hypercube $[0, 1]^d$ but to a slightly smaller region then it suffices to take the normal hat function basis when classifying adaptively. This allows to start with $2d+1$ grid points for the sparse grid on level two, creating "boundary" grid points only when necessary.

To Appear in:
Modelling, Simulation and Optimization of Complex Processes, Proc. Int. Conf. HPSC, Hanoi, Vietnam, 2006.

Table 2. Conventional hat basis functions vs. modified boundary functions

hat functions				modified boundary functions			
# grid points	max l	# iterations	acc.	# grid points	max l	# iterations	acc.
5	2	5	89.9	5	2	5	89.9
17	3	18	91.1	17	3	20	91.0
49	4	35	91.0	49	4	57	91.0
123	5	50	91.0	117	5	115	91.0
263	6	64	90.7	232	6	169	90.6
506	7	85	90.8	470	6	235	90.7
866	7	96	90.7	812	7	252	90.8

5 Summary

We presented an adaptive classification algorithm using sparse grids to discretize feature space. The algorithmically hard part, the multiplication with the stiffness matrix, was sketched. The algorithm allows for the classifcation of large datasets as it scales only linearly in the number of training data points.

Using adaptivity in sparse grid classification allows to reduce the number of grid points significantly. An adaptive selection of grid points is especially useful for higher dimensional feature spaces.

Special care has to be taken regarding the boundary values. Sparse grids usually employ grid points located on the boundary. A vast number of grid points can be saved if those are omitted and if the datasets are normalized so that no data points are located on the boundary instead. This allows to start with only $2d + 1$ grid points; adaptivity takes care about the creation of grid points next to the boundary.

Ongoing research includes investigations on employing other refinement criteria which are suited better for classification and the use of other basis functions to further improve adaptive sparse grid classification.

References

[BuGr04] H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica* Volume 13, 2004, p. 147–269.

[GaGT01] J. Garcke, M. Griebel and M. Thess. Data Mining with Sparse Grids. *Computing* 67(3), 2001, p. 225–253.

[NHBM98] D. Newman, S. Hettich, C. Blake and C. Merz. UCI Repository of machine learning databases, 1998.

[Pflü05] D. Pflüger. Data Mining mit Dünnen Gittern. Diplomarbeit, IPVS, Universität Stuttgart, March 2005.

[RiHj95] B. D. Ripley and N. L. Hjort. *Pattern Recognition and Neural Networks.* Cambridge University Press, New York, NY, USA. 1995.

[Sing98] S. Singh. 2D spiral pattern recognition with possibilistic measures. *Pattern Recogn. Lett.* 19(2), 1998, p. 141–147.

To Appear in:
Modelling, Simulation and Optimization of Complex Processes, Proc. Int. Conf. HPSC, Hanoi, Vietnam, 2006.