Adaptive Sparse Grid Classification using Grid Environments

Dirk Pflüger, Ioan Lucian Muntean, and Hans-Joachim Bungartz

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

Abstract. Common techniques tackling the task of classification in data mining employ ansatz functions associated to training data points to fit the data as well as possible. Instead, the feature space can be discretized and ansatz functions centered on grid points can be used. This allows for classification algorithms scaling only linearly in the number of training data points, enabling to learn from data sets with millions of data points. As the curse of dimensionality prohibits the use of standard grids, sparse grids have to be used.
Adaptive sparse grids allow to get a trade-off between both worlds by refining in rough regions of the target function rather than in smooth ones. We present new results for some typical classification tasks and show first observations of dimension adaptivity. As the study of the critical parameters during development involves many computations for different parameter values, we used a grid environment which we present.

Key words: data mining, classification, adaptive sparse grids, grid environment

1 Introduction

Today, an ever increasing amount of data is available in various fields such as medicine, e-commerce, or geology. Classification is a common task making use of previously known data and making predictions for new, yet unknown data. Efficient algorithms that can process vast datasets are sought for. The basics of sparse grid classification have already been described in [1, 2], for example. Therefore, in this section, we summarize the main ideas only very briefly and refer to the references cited above for further information.

We focus on binary classification. Given is a preclassified set of M data points for training, $S = \{(\boldsymbol{x}_i, y_i) \in [0,1]^d \times \{-1,1\}\}_{i=1}^M$, normalized to the d-dimensional unit hypercube. The aim is to compute a classifier $f : [0,1]^d \to \{-1,1\}$ to obtain a prediction of the class -1 or $+1$ for previously unseen data points. To compute f, we follow the Regularization Network approach and minimize the functional

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

Appeared in:
Y. Shi et al. (Eds.): ICCS 2007, Part I, LNCS 2287, pp. 708-715, 2007.
http://www.springerlink.com/content/670mx3w5202688p3/

with the cost function $(y_i - f(\boldsymbol{x}_i))^2$ ensuring good approximation of the training data by f and the regularization operator $\|\nabla f\|^2_{L_2}$ guaranteeing that f is somehow smooth, which is necessary as the classifier should generalize from S. The regularization parameter λ stirs the trade-off between accuracy and smoothness.

Rather than common algorithms which employ mostly global ansatz functions associated to data points, scaling typically quadratically or worse in M, we follow a somehow data-independent approach and discretize the feature space to obtain a classification algorithm that scales linearly in M: We restrict the problem to a finite dimensional space V_N spanned by N basis functions ϕ_j to obtain our classifier $f_N(\boldsymbol{x}) = \sum_{j=1}^N \alpha_j \phi_j(\boldsymbol{x})$, in our case the space of d-linear functions. Minimization of $H[f]$ leads to a linear system with N unknowns,

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

with $C_{ij} = \left(\nabla\phi_i(\boldsymbol{x}), \nabla\phi_j(\boldsymbol{x})\right)_{L_2}$ and $B_{ij} = \phi_i(\boldsymbol{x}_j)$.

To counter the curse of dimensionality and to avoid N^d unknowns in d dimensions, we use sparse grids, described for example in [3]. Regular sparse grids $V_n^{(1)}$ up to level n in each direction base on a hierarchical formulation of basis functions and an a priori selection of grid points – needing only $\mathcal{O}(N \log(N)^{d-1})$ grid points with just slightly deteriorated accuracy. Sparse grids have been used for classification via the combination technique [1], where the sparse grid solution is approximated by a combination of solutions for multiple, but smaller and regular grids. Sparse grids have the nice property that they are inherently adaptive, which is what we will make use of in the following sections.

Using sparse grids, the system of linear equations can be solved iteratively, each iteration scaling only linearly in the number of training data points and grid points, respectively. The underlying so-called UpDown algorithm which was shown in [2] bases on traversals of the tree of basis functions.

2 Grid-Based Development

For classification using regular sparse grids there are two important parameters determining the accuracy of the classifier to be learned. First, we have n, the maximum level of the grid. For low values of n there are usually not enough degrees of freedom for the classifier to be able to learn the underlying structure, whereas large n lead to overfitting, as each noisy point in the training data set can be learnt. Second and closely related, there is the regularization parameter λ, steering the trade-off between smoothness and approximation accuracy. Given enough basis functions, λ determines the degree of generalization.

To find good values for these parameters, heuristics or experience from other problems can be used. For a fixed n the accuracy is maximized for a certain value of λ and decreases for higher or lower values, oscillating only little, for example. Especially during the development stage, heuristics are not sufficient; an optimal combination of both parameters for a certain parameter resolution is of interest. To make things even harder, systems under development are usually

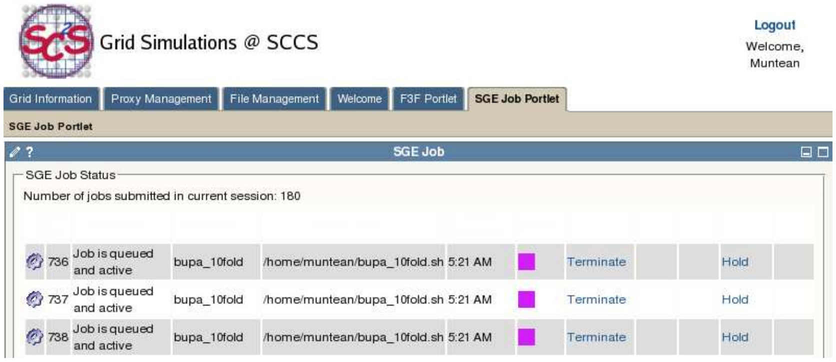
Appeared in:
Y. Shi et al. (Eds.): ICCS 2007, Part I, LNCS 2287, pp. 708-715, 2007.
http://www.springerlink.com/content/670mx3w5202688p3/

not optimized for efficiency, but rather designed to be able to be flexible so that different approaches can be tested.

Such parameter studies typically demand a significant computational effort, are not very time critical, but should be performed within a reasonable amount of time. This requires that sufficient computational power and/or storage space is available to the developer at the moment when needed. Grid environments grant users access to such resources. Additionally, they provide an easy access and therefore simplify the interaction of the users – in our setting the algorithm developers – with various resources [4]. In grid middleware, for example the Globus Toolkit [5], this is achieved through mechanisms such as single sign-on, delegation, or by providing extensive support for job processing. Parameter studies are currently among the most widespreaded applications of grid computing.

For the current work, we used the grid research framework GridSFEA (Grid-based Simulation Framework for Engineering Applications) [6]. This research environment aims at providing various engineering simulation tasks, such as fluid dynamics, in a grid environment. GridSFEA uses the Globus Toolkit V4. It comprises both server-side components (grid tools and services) and client-side tools (such as an OGCE2 grid portal, using the portlet technology as a way to customize the access to the grid environment).

The grid portal was extended by a portlet adapted to the requirements of sparse grid classification. Thus, support was introduced for sampling the parameter space, for automatic generation and submission of corresponding jobs, and for collecting and processing of results. Figure 1 shows a snapshot of our current portlet, running a sparse grid classification for various values of λ.



Number of jobs submitted in current session: 180

736	Job is queued and active	bupa_10fold	/home/muntean/bupa_10fold.sh	5:21 AM	🟪	Terminate			Hold	
737	Job is queued and active	bupa_10fold	/home/muntean/bupa_10fold.sh	5:21 AM	🟪	Terminate			Hold	
738	Job is queued and active	bupa_10fold	/home/muntean/bupa_10fold.sh	5:21 AM	🟪	Terminate			Hold	

Fig. 1. Performing sparse grid parameter studies using a portlet of the GridSFEA portal

The usage of this grid portal enables the algorithm developer to focus on the design of the algorithm and the interpretation of the results, and to leave the management of the computations to the framework. Furthermore, the portal of GridSFEA allows the user to choose the grid environment to be used. This way, the computations – here parameter studies – can be performed at different computing sites, such as supercomputing centres.

Appeared in:
Y. Shi et al. (Eds.): ICCS 2007, Part I, LNCS 2287, pp. 708-715, 2007.
http://www.springerlink.com/content/670mx3w5202688p3/

3 Adaptivity and Results

The need for a classification algorithm that scales only linearly in the number of training data points forces us in the first place to trade the possibility to make use of the structure of the training data for a somehow data-independent discretization of the feature space. Therefore we have to deal with a higher number of basis functions than common classification algorithms which try to fit the data with as few basis functions as possible. Even though one iteration of a sparse grid solver scales only linearly in the number of unknowns, this still imposes restrictions on the maximum depth of the underlying grid.

The idea of adaptive sparse grid classification is to obtain a trade-off between common, data-dependent algorithms and the data independence of sparse grid classification. The aim is to reduce the order of magnitude of unknowns while keeping the linear time training complexity: Especially those grid points should be invested that contribute most to the classification boundary, as the zero-crossing of the classifier determines the class prediction on new data. For the classification task, the exact magnitude of the approximation error is not important in regions that are purely positive or negative. It is reasonable to spend more effort in rough regions of the target function than in smooth ones.

As we already showed in [2], special consideration has to be put on treating the boundary of the feature space. Normally, in sparse grid applications, grid points have to be invested on the boundary to allow for non-zero function values there. Employing adaptivity in classification allows us to neglect those grid points as long as it is guaranteed that no data points are located exactly on the boundary. This corresponds to the assumption that the data belonging to a certain class is somehow clustered together – and therefore can be separated from other data – which means that regions towards the border of the feature space usually belong to the same class. In regions where the classification boundary is close to the border of the feature space, adaptivity will take care of this by refining the grid where necessary and by creating basis functions that take the place of common boundary functions.

Thus we normalize our data to fit in a domain which is slightly smaller than the d-dimensional unit hypercube, the domain of f_N. This way we can start the adaptive classification with the d-dimensional sparse grid for level 2, $V_2^{(1)}$, without the boundary and therefore only $2d+1$ grid points, rather than using grid points on the boundary, which would result in $\sum_{j=0}^{d} \binom{d}{j} 2^{d-j}(1+2j) = 3^d + 2d \cdot 3^{d-1} \in \mathcal{O}(d \cdot 3^d)$ unknowns.

For the remaining part of this section we proceed as follows: Starting with $V_2^{(1)}$, we use a preconditioned Conjugated Gradient method for a few iterations to train our classifier. Out of all of the refinement candidates we choose the grid point with the highest surplus, add all the children to our current grid and repeat this until satisfied. One has to take care not to violate one of the basic properties of sparse grids: For each basis function all parents in the hierarchical tree of basis functions have to exist. All missing ancestors have to be created, which can be done recursively.

Appeared in:
Y. Shi et al. (Eds.): ICCS 2007, Part I, LNCS 2287, pp. 708-715, 2007.
http://www.springerlink.com/content/670mx3w5202688p3/

3.1 Classification Examples

The first example, taken from Ripley [7], is a two-dimensional artificial dataset which was constructed to contain 8% of noise. It consists of 250 data points for training and 1000 points to test on. Being neither linear separable nor very complicated, it shows typical characteristics of real world data sets. Looking for a good value of λ we evaluated our classifier after six refinement steps for $\lambda = 10^{-i}$, $i = 0 \dots 6$, took the ones with the best and the two neighbouring accuracies, namely 0.01, 0.001 and 0.0001, and looked once more for a better value of lambda in between those by evaluating for $\lambda = 0.01 - 0.001 \cdot i$, $i = 1 \dots 9$ and $\lambda = 0.001 - 0.0001 \cdot i$, $i = 1 \dots 9$. The best λ we got this way was 0.004.

Figure 2 shows the training data (left) as well as the classification for $\lambda = 0.004$ and the underlying adaptive sparse grid after only 7 refinement steps (right). Even though there are only a few steps of refinement, it can clearly be observed that more grid points are spent in the region with the most noise and that regions with training data belonging to the same class are neglected. Note that one refinement step in x_1-direction (along $x_2 = 0.5$) causes the creation of two children nodes in x_2-direction (at $x_2 = 0.25$ and $x_2 = 0.75$).

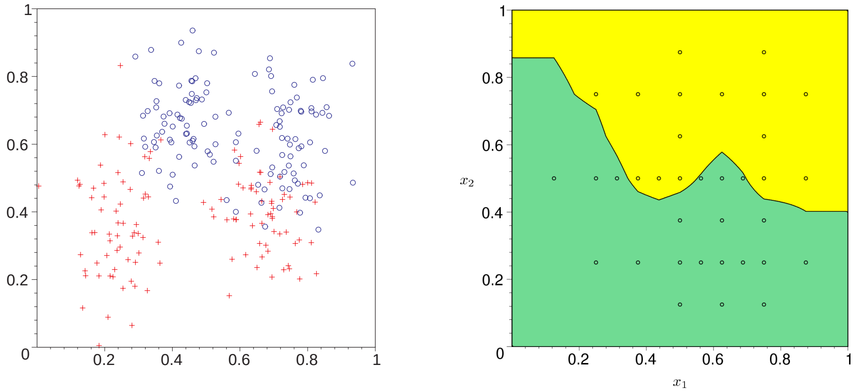


Fig. 2. Ripley dataset: training data and classification for $\lambda = 0.004$

Table 1 shows some results comparing different sparse grid techniques. For the combination technique and sparse grids with and without grid points on the boundary, we calculated the accuracy for level one to four and $\lambda := 10^{-j} - i \cdot 10^{-j-1}$, $i = 0 \dots 8, j = 0 \dots 5$, each. Increasing the level further does not improve the results as this quickly leads to overfitting. We show the number of unknowns for each grid, the best value of λ, and the best accuracy achieved on the test data. A general property of sparse grid classification can be observed: The higher the number of unknowns, the more important the smoothness functional gets and therefore the value for the regularization parameter λ increases. Using a coarser grid induces a higher degree of smoothness, which is a well-known phenomenon.

Appeared in:
Y. Shi et al. (Eds.): ICCS 2007, Part I, LNCS 2287, pp. 708-715, 2007.
http://www.springerlink.com/content/670mx3w5202688p3/

Table 1. Ripley dataset: accuracies [%] obtained for the combination technique, regular sparse grids with and without boundary functions, and the adaptive technique

	comb. techn.			sg boundary			sg			adapt. sg $\lambda = 0.004$	
n	|grids|	λ	acc.	|grid|	λ	acc.	|grid|	λ	acc.	|grid|	acc.
1	9	$6 \cdot 10^{-5}$	90.3	9	$6 \cdot 10^{-5}$	90.3	1	*	50.0	5	89.9
2	39	0.0005	90.8	21	0.0004	90.7	5	0.0005	90.3	9	90.2
3	109	0.005	91.1	49	0.006	91.2	17	0.005	91.2	13	89.8
4	271	0.007	91.1	113	0.005	91.2	49	0.007	91.2	19	91.1
										21	91.2
										24	91.2
										28	91.3
										32	91.4
										35	91.5

Another observation is that the assumption that grid points on the boundary can be neglected, which holds here even for regular grids. Of course, the sparse grid on level 1 can do nothing but guess one class value for all data points and five unknowns are not enough to achieve an accuracy of 90.7%, but already for level 3 the same accuracy as for the grid with boundary values is reached.

For the adaptive sparse grid classification we show the results only for $\lambda = 0.004$ for eight times of refinement. With less grid points than the sparse grid with boundary on level 3, we achieved an excellent accuracy of 91.5% – 0.3% higher than what we were able to get using regular sparse grids. This is due to the fact that increasing a full level results in a better classification boundary in some parts, but leads to overfitting in other parts at the same time. Here, adaptivity helps to overcome this problem.

We used our grid environment to compute and gather the results for these 180 runs for each level, even though the problem sizes are quite low: Our implementation of the combination technique has just been done for comparing it with the direct sparse grid technique, for example, and it is therefore far from being efficient. We neither use an efficient solver as a multigrid technique for example, nor care about the amount of memory we consume. We are just interested in getting eventually some results, a problem suited for grid environments. As the portlet mentioned in Sec. 2 allows us to easily specify a set of parameters, the adaptation for the use in GridSFEA was just a matter of minutes.

A second, 6-dimensional dataset we used is a real-world dataset taken from [8] which contains medical data of 345 patients, mainly blood test values. The class value indicates whether a liver disease occurred or not. Because of the limited data available we did a stratified 10-fold cross-validation. To find a good value for λ we calculated the accuracy for different values of the regularization parameter as we did above for the regular sparse grids. This time we calculated the accuracy of the prediction on the 10-fold test data. Again we used GridSFEA to compute and gather the results. Theoretically one could even submit a job for each fold of the 10-fold cross-validation and gather the different accuracies

Appeared in:
Y. Shi et al. (Eds.): ICCS 2007, Part I, LNCS 2287, pp. 708-715, 2007.
http://www.springerlink.com/content/670mx3w5202688p3/

afterwards, but as the coefficients computed for the first fold can be used as a very good starting vector for the PCG in the other 9 folds, this was neglected.

Table 2 shows some results, comparing the results of the adaptive technique (zero boundary conditions) with the best ones taken from [9], the combination technique (anisotropic grid, linear basis functions), and from [10], both linear and non-linear support vector machines (SVM). The adaptive sparse grid technique was able to outperform the other techniques. For a relatively large value of λ and after only 9 refinement steps we got a 10-fold testing accuracy which is more than 2.3% higher than our reference results – with only 1302 unknowns. Again it proved to be useful to refine in the most critical regions while neglecting parts of the domain where the target function is smooth.

Table 2. Accuracies [%] (10-fold testing) for the Bupa liver dataset

adapt. sg			comb. techn.	SVM	SVM
$\lambda = 0.3$			lin. anisotrop.	linear	non-linear
# refinements	$\lvert\text{grid}\rvert$	acc.	acc.	acc.	acc.
7	485	70.71	73.9	70.0	73.7
8	863	75.64			
9	1302	76.25			

3.2 Dimension Adaptivity

Similar observations can be made for a third dataset, the Pima diabetes dataset, taken again from [8]. 769 women of Pima Indian heritage living near Phoenix were tested for diabetes mellitus. 8 features describe the results of different examinations, such as blood pressure measurements. For reasons of shortness we will focus only on some observations of the dimension adaptivity of the adaptive sparse grid technique.

As assumed, the adaptive refinement neglects dimensions containing no information but only noise quite well. Extending the diabetes dataset for example by one additional feature with all values set to 0.5 leads to two more grid points for $V_2^{(1)}$. As the two additional surpluses are close to zero, the effects on BB^T and $B\mathbf{y}$ are just minor, see (1). But extending the dimensionality modifies the smoothness functional. There are stronger impacts on C and therefore the trade-off between smoothness and approximation error changes. One could expect that the same effects could be produced by changing the value of λ suitably. And in fact, for training on the first 567 instances and testing on the remaining 192 data points, almost identical accuracies (about 74.5%) can be achieved when changing λ to 0.0002 compared to extending the dimensionality by one for $\lambda = 0.001$ – at least for the first few refinements.

Further improvement could be achieved by omitting the "weakest" dimension, the one with the lowest surpluses, during refinement. When some attributes are expected to be less important than others, this could lead to further improvements considering the number of unknowns needed.

Appeared in:
Y. Shi et al. (Eds.): ICCS 2007, Part I, LNCS 2287, pp. 708-715, 2007.
http://www.springerlink.com/content/670mx3w5202688p3/

4 Summary

We showed that adaptive sparse grid classification is not only possible, but useful. It makes use of both worlds: the data-independent and linear runtime world and the data-dependent, non-linear one that reduces the number of unknowns.

If increasing the level of a regular sparse grid leads to overfitting in some regions but improves the quality of the classifier in others, adaptivity can take care of this by refining just in rough regions of the target function. In comparison to regular sparse grids, the adaptive ones need far less unknowns. This can reduce the computational costs significantly.

Considering dimension adaptivity we pointed out first observations. Further research has to be invested here.

Finally we demonstrated the use of GridFSEA, our grid framework, for parameter studies, especially during algorithm development stage.

Theoretically, even high dimensional classification problems could be tackled by adaptive sparse grid classification as we showed that we can start with a number of grid points which is linear in the dimensionality. Practically, our current smoothness functional permits high dimensionalities by introducing a factor of 2^d in the number of operations. Therefore we intend to investigate the use of alternative smoothness functionals in the near future to avoid this.

References

1. Garcke, J., Griebel, M., Thess, M.: Data mining with sparse grids. Computing **67**(3) (2001) 225–253
2. Bungartz, H.J., Pflüger, D., Zimmer, S.: Adaptive Sparse Grid Techniques for Data Mining. In: Modelling, Simulation and Optimization of Complex Processes, Proc. Int. Conf. HPSC, Hanoi, Vietnam, Springer (2007) to appear.
3. Bungartz, H.J., Griebel, M.: Sparse grids. Acta Numerica **13** (2004) 147–269
4. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers (2005)
5. Foster, I.: Globus toolkit version 4: Software for service-oriented systems. In: IFIP International Conference on Network and Parallel Computing. Volume 3779 of LNCS., Springer-Verlag (2005) 2–13
6. Muntean, I.L., Mundani, R.P.: GridSFEA - Grid-based Simulation Framework for Engineering Applications. `http://www5.in.tum.de/forschung/grid/gridsfea` (2007)
7. Ripley, B.D., Hjort, N.L.: Pattern Recognition and Neural Networks. Cambridge University Press, New York, NY, USA (1995)
8. Newman, D., Hettich, S., Blake, C., Merz, C.: UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html` (1998)
9. Garcke, J., Griebel, M.: Classification with sparse grids using simplicial basis functions. Intelligent Data Analysis **6**(6) (2002) 483–502
10. Fung, G., Mangasarian, O.L.: Proximal support vector machine classifiers. In: KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM Press (2001) 77–86

Appeared in:
Y. Shi et al. (Eds.): ICCS 2007, Part I, LNCS 2287, pp. 708-715, 2007.
http://www.springerlink.com/content/670mx3w5202688p3/