Dimension-adaptive Sparse Grids

Jörg Blank

April 16, 2008

Abstract

In many situations it is often necessary to compute high dimensional integrals. Due to the curse of dimensionality naive methods are not viable. In this paper a method to solve this problem is proposed. Using datamining as an example, dimension-adaptive Sparse Grids are introduced.

1 Data Mining

Data mining influences every aspect of our live. Large amounts of data are gathered every minute and new techniques are needed to cope with. One special aspect is the recovery of a lost function from measurements. We define a dataset as

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

with S being a d-dimensional dataset with M entries. It is possible to restrict y_i to a arbitiary number of classes. That leads to classification. We now assume that the data points are evaluations of an unkown function $y = f(x_1, x_2, \ldots, x_d)$, possibly with measurement errors. f should be taken from a arbitiary function space V over R^d.

1.1 Regularisation

Now we want to recover f as good as possible. There should be a f such that it minimises a functional R.

$$\min_{f \in V} R(f)$$

We define this functional as:

$$R(f) = \frac{1}{M} \sum_{i=1}^{M} \Psi(f(\mathbf{x}_i), y_i) + \lambda \Phi(f)$$

For the next step we have to choose our function space. We confine V to a discrete space V_N. Functions in this space can be written as:

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

ϕ_j denotes the base functions and α_j the weights we are interested in. We still have to chose which base functions we want to use and this choice has a major impact on the viability of a method. A naive choice will generate points in the size of $O(n^d)$, which

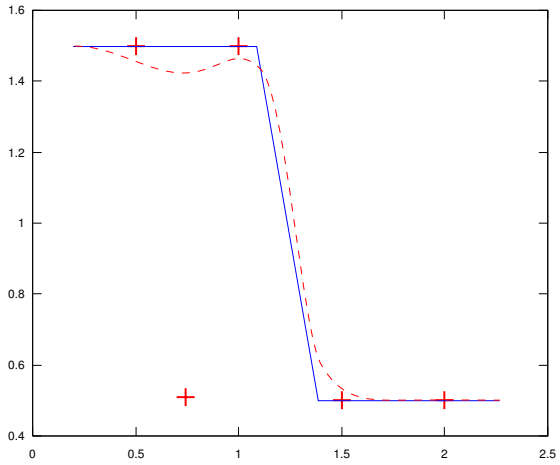

Figure 1: Function reconstruction

becomes impossible to compute very fast. By using a function $\Phi(f)$ to regularise our expression, we can be sure, that there is only one solution. We can now aquire the minimum by differentating after α_k with $k \in \{1 \dots N\}$.

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

For the full calculation refer to [3]. This is a system of linear equations and can be written in a shorter matrix form.

$$(\lambda C + B \cdot B^T)\alpha = By$$

The system is symmetric and positiv definite and can be solved by a standard solver like Conjugated Gradients method.

2 Sparse Grids

2.1 Introduction

As mentioned is the choice of the basis functions a major contributor to the success of a method. Using a naive approach like a nodal basis leads to a grid of size $O(n^d)$. Instead we just use grids with less grid points. We have to define a base function first. One of the easiest is the hat function.

$$\phi(x) = \begin{cases} 1 - |x| & \text{if } x \in [-1, 1] \\ 0 & \text{otherwise} \end{cases}$$

We can easily apply proper scaling for a given level l and an index i describing the position:

$$\phi_{l,i}(x) = \phi(\frac{x - i \cdot h_l}{h_l}) = \phi(\frac{x - i \cdot 2^{-l}}{2^{-l}}) = \phi(x \cdot 2^l - i)$$

Now we can construct an hierarchical basis as seen in figure 2. In a next step we take

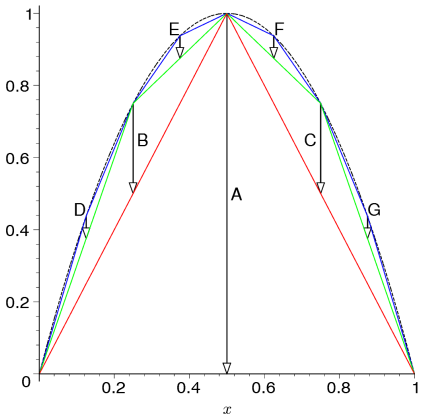

Figure 2: Datamining mit Dünnen Gittern, Pflüger

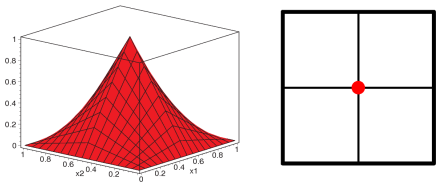

Figure 3: a 2-dimensional pagoda. AWR2, Bungartz

the d-dimensional tensor product of our scaled hat function and extend it to a pagoda form.

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

After extending the hierarchical pattern into higher dimensions we get the structure shown in figure 4. Now it is possible to chose grid points with large contribution to the solution. It can be shown that the right choices in figure 5 is optimal. Such grids only consist of $O(n * log(n)^{d-1})$ points.

3 Combination technique

Working directly on Sparse Grids requires a lot of overhead. It is however possible to use several smaller regular grids and combine them to get a Sparse Grid. For non-adaptive Sparse Grids we can exploit combinatorial 'inclusion-exclusion' principle to get a general formula[1]:

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

Using the combination technique has several advantages. Existing codes can be used to calculate a solution for the smaller regular grids. It is also very easy to parallelize because each of the subgrids can be computed without any communication. The major disadvantage is, that only regular Sparse Grids can be created.

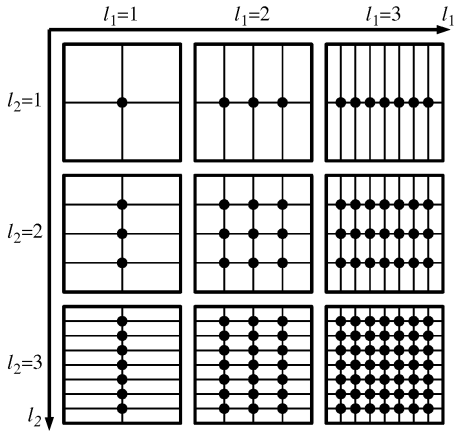

Figure 4: subgrid pattern. AWR2, Bungartz

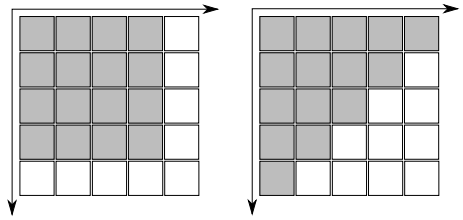

Figure 5: subgrid choices

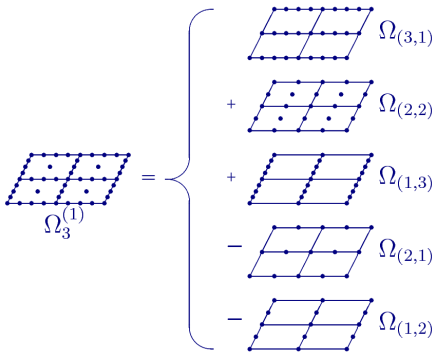

Figure 6: combination technique. AWR2, Bungartz

3.1 Admissibility

We do not want to allow all subspace combination. Instead we require all combinations to be admissible[2]. We define $\mathcal{I}$ as the set of selected indices. I should now fullfill:

- $\underline{k} \in \mathcal{I}$
- $\underline{j} \leq \underline{k} \Rightarrow \underline{j} \in \mathcal{I}$

3.2 Adaptivity

After defining the smallest grid as $\underline{1} = (1, \ldots, 1)$, we can now add new grid points. This points must remain admissible and must provide a large contribution to the solution. The contribution can be measured by calculation $\varepsilon = R(f)$ without the regularisation term. A large ε is a sign for a bad fitting. Generally it is wise to refine in that direction.

3.3 Algorithm

This leads to the following algorithm:

- Initizalize index set $I = \{\underline{1}\}$
- Initizalize old index set $O = \{\}$
- Solve problem on $\underline{1}$
- while global $\varepsilon >$ bound
  - Choose $\underline{i} \in I$ with largest $\varepsilon_{\underline{i}}$
  - Refine in all dimensions, if admissible in O
  - Move $\underline{i}$ to O
  - Calculate problems and ε on new indexes
  - Update global ε

Figure 7 shows which refinement steps are allowed and which are forbidden.

3.4 Conclusion

Combination Sparse Grids are a easy to implement solution, because old codes can be reused. The technique requires grid points on the border, which easily becomes unbearable in higher dimensions. This can be eased using dimension-adaptive Sparse Grids, especially when working with additive functions like $f(\mathbf{x}) = f_1(x_1) + f_2(x_2) + \ldots + f_d(x_d)$.

References

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

[2] M. Hegland. Adaptive sparse grids.

[3] Dirk Pflüger. Data mining mit dünnen gittern. Diplomarbeit, IPVS, Universität Stuttgart, March 2005.

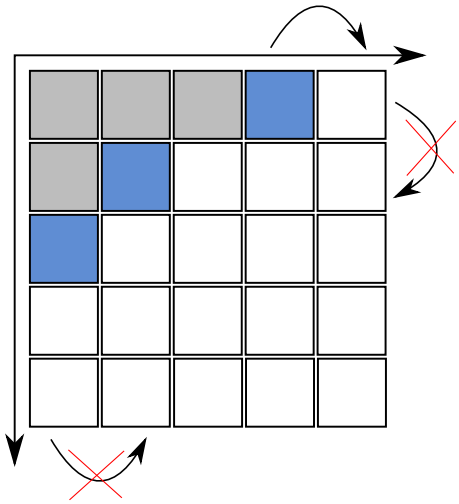

Figure 7: dimension-adaptive algorithm