

$$f(x) \approx \hat{f}(x) = \sum_{i \in G} \alpha_i \phi_i(x)$$

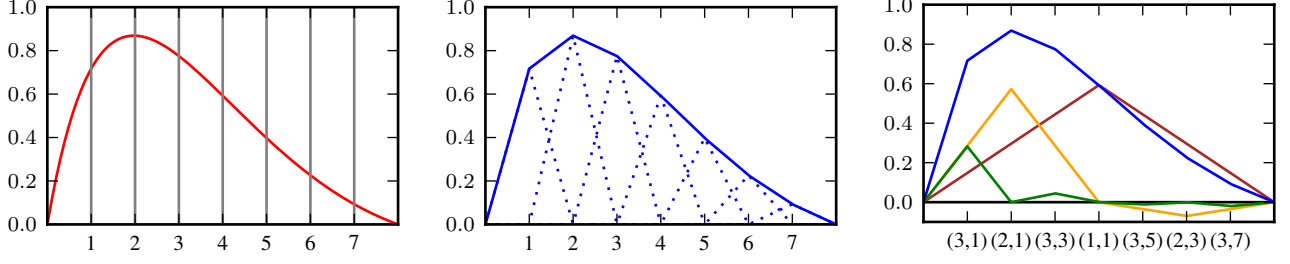


Fig. 1: A function  $f$ , red, to be discretized. Seven grid points discretizing the space (left). Full grid discretization by nodal basis (middle) and hierarchical basis, scaled by  $\alpha_i$  and  $\alpha_{l,i}$  respectively (right).

over all weighted basis-functions  $\phi_i$  discretizes (approximates)  $f$  [2]. Fig. 1 illustrates this.

For  $f(\vec{x})$  with  $d > 1$ , grid point representation is extended to a  $d$ -tuple of indices, i.e.  $(1, 3, 1)$  denoting the grid point with position  $x = 1$ ,  $y = 3$ ,  $z = 1$  in the dimensions  $x, y, z$ . The related basis function

$$\phi_i(\vec{x}) = \prod_{j=1}^d \phi_{i,j}(x_j)$$

gets extended to  $d$  dimensions using the tensor product over the previously defined one-dimensional hat functions  $\phi_{i,j}(x_j)$  with  $x_j$  being the  $j$ -th element of  $\vec{x}$  and  $\phi_{i,j}$  denoting the basis-function of grid point  $i$  in the dimension  $j$  [2]. To improve readability the dimension-related index  $j$  of  $\phi_{i,j}$  will be omitted in the following.

### B. Hierarchical basis

Besides constructing the grid in the simple way described in Sec. II-A, more sophisticated methods are available. In order to make a grid sparse and still keep a sufficient accuracy the *hierarchical basis* is introduced.

We first examine the case  $d = 1$ . Let  $l \in \{1, 2, \dots\}$  be the *level* with  $|G| = 2^{(l-1)}$  associated grid points on each level. This level hierarchy groups grid points into sets

$$G_l = \{i \in \mathbb{N} \mid 1 \leq i \leq 2^l, i \text{ odd}\},$$

omitting every second grid point. Together the adjusted hat function

$$\phi_{l,i}(x) = \max\{0, 1 - |2^l x - i|\}$$

this forms the hierarchical basis in one dimension up to a level  $n$  [2]. By disregarding every even grid point, the local supports of basis functions on the *same* level are mutually exclusive and for each value of  $x$  exactly one basis function is not zero.

A grid point is now referred to as grid point in the level  $l$  with index  $i$ . Note, that the indices alone do not uniquely define a grid point (i.e. grid points of index  $i = 1$  are element in every  $G_l$ ). The same applies to the corresponding  $\phi$  and  $\alpha$ .

Taking the weighted sum over all levels and all grid points in one dimension

$$f(x) \approx \hat{f}(x) = \sum_{l \leq n, i \in G_l} \alpha_{l,i} \phi_{l,i}(x)$$

discretizes  $f$  on a full grid [2].

In contrast to the conventional approach from Sec. II-A the surpluses now are calculated differently. Let  $x_{l,i}$  be the  $x$ -value of the grid point given by  $l$  and  $i$ . Then

$$a_{l,i} = f(x_{l,i}) - \frac{f(x_{l,i} - 2^{-l}) + f(x_{l,i} + 2^{-l})}{2}$$

is the *hierarchical surplus* for the grid point  $(l, i)$ . The function value at the grid point is taken and the function values at neighbouring grid points are subtracted (“neighbouring” is disregarding  $l$ ) [4]. For instance, for  $\alpha_{2,1}$  we get  $x_{2,1} = \frac{1}{4}$  and

$$\alpha_{2,1} = f\left(\frac{1}{4}\right) - \frac{f\left(\frac{1}{4} - 2^{-2}\right) + f\left(\frac{1}{4} + 2^{-2}\right)}{2}.$$

For  $d > 1$  we combine the one dimensional basis functions to  $d$ -dimensional basis functions using the tensor product, analogous to Sec. II-A. This is done for all possible combinations of  $l$  and  $i$  in all dimensions. This process of building  $d$ -dimensional basis functions through combining over the level in different dimensions leads to a subspaces defined by the level-vector  $\vec{l} = (l_x, l_y, \dots)$  as shown for  $d = 2$  in Fig. 2. Taking the sum  $\sum_{l,i} \alpha_{l,i} \phi_{l,i}(x)$  over all

$$\phi_{l,i}(\vec{x}) = \prod_j \phi_{l,i,j}(x_j)$$

discretizes  $f(\vec{x})$  on a hierarchical structured grid [2].

However, this does not lead to a sparse grid immediately. So far the grid points only got regrouped and for a the maximum level  $n$  this results in  $|G| = 2^n - 1$  basis functions for each dimension. This further leads to an exponential dependency of the number of grid points and  $d$ , thus having no effect on mitigating the curse of dimensionality [2].

### C. Sparse grid discretization

In order to make the hierarchical grid *sparse*, we now disregard certain subspaces with their associated grid points. The goal is to reduced the total number of grid points by finding and disregarding those that contribute the least to the discretization of  $f$ . Which grid points that are is a *a-priori* solvable optimization problem [2]. Thus, independent of  $f$  all

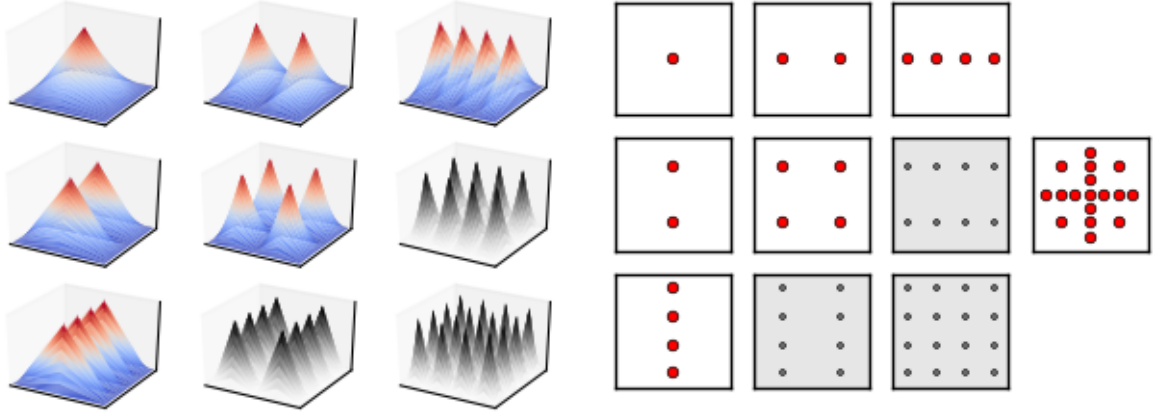


Fig. 2: Hierarchical subspaces (left) and corresponding grid points (right) ( $d = 2$ ) with the last column being the complete sparse grid. For the sparse grid omitted subspaces/grid points in grey.

$\phi_{l,i}$  related to the subspaces in the lower right of the diagonal in Fig. 2 will be left out of the sum

$$\hat{f}(x) = \sum_{\substack{l \leq n, i \in G_l \\ |l| \leq n+d-1}} \alpha_i \phi_{l,i}(x)$$

from Sec. II-A [1], [2].

By doing so, we reduced the total number of grid points drastically from  $\mathcal{O}(2^{nd})$  to  $\mathcal{O}(2^n \cdot n^{d-1})$  where  $n$  is the maximal level in the hierarchical structure. The asymptotic error on the other hand only slightly increases from  $\mathcal{O}(2^{-2n})$  to  $\mathcal{O}(2^{-2n} \cdot n^{d-1})$  [2]. Tab. I illustrates the quickly growing gap between the number of grid points in a full and sparse grid.

d	1	2	3	5	10	20
Full	15	225	3375	$> 10^5$	$> 10^{11}$	$> 10^{23}$
Sparse	15	49	111	351	2001	13201

TABLE I: Number of grid points in a full and sparse grid (without points on the boundaries) with maximal level  $n = 4$  and growing dimension  $d$ .

It is important to note that the numbers in Tab. I are taken for a sparse grid without points on the boundaries. In case the function to be discretized is not zero on the boundaries, such points become necessary. Treatment of the boundaries might require considerably more grid points. For further information please refer to [2], [3].

#### D. Adaptive sparse grids

Even though sparse grid offer an optimal choice for a general function  $f$  the discretization error is in praxis often unacceptable due to the properties of  $f$  itself. If  $f$  exhibits steep, complex or discontinuous areas, which the *a-priori* distribution of grid points can not capture, additional grid points have to be added locally [2]. Two different approaches are possible:

Either we test all potential grid points on how much they

would contribute to the discretization of  $f$  and chose the those that contribute most. Or we use information about  $f$  itself, i.e. the location steep areas and refine the grid there.

The first approach quickly becomes infeasible because in order to test a potential grid point a number of (possibly expensive) function evaluations of  $f$  become necessary. Additionally the number of candidates is even in few dimension very high [2].

The second approach requires to find criteria based on  $f$  to find interesting areas to refine but is generally much better due to the cost-related problems of the first approach [2].

The refinement process itself consist of finding a grid point to refine, then adding neighbouring grid points around it with  $l + 1$  in each dimension. Often it is necessary to also add the hierarchical parents of the newly added grid points to ensure consistency required by many algorithms on the grid [2]. Fig. REF illustrates this refinement process for one point.

Spatial adaptivity is crucial for applications in machine learning, due to often complex functions arising from regression or decision boundaries [1], [2].

### III. SPARSE GRIDS IN MACHINE LEARNING

As mentioned in the introduction, data mining scenarios with large datasets and high-dimensional data are in general difficult to deal with. Areas of application that show these characteristics include engineering (e.g. heat-transfer modeling) [3], machine learning and data mining (e.g. estimation of cosmological redshifts and option pricing in finance) [2]. In this section we apply sparse grids to machine learning tasks (classification and regression) relevant for data mining.

#### A. Least squares estimation

A commonly used method in machine learning is *least squares estimation*

$$\hat{c} = \arg \min_f \left( \frac{1}{M} \sum_{i=1}^M \left( y^{(i)} - f(x^{(i)}) \right)^2 + \lambda R(f) \right)$$

using the notation for training data introduced in Sec. II. With this formula we derive an estimator  $\hat{c}$  that is the function which models the data  $X \times Y$  with the least error (minimization over  $f$ ). The term  $R(f)$  is a regularization term to enforce smoothness. A common choice is the  $L_2$ -Norm of the gradient of  $f$ ,  $R(f) = \|\nabla f\|_2$ .

We will now use least squares estimation in a sparse grid setting. Intuitively,  $f$  and ultimately  $\hat{c}$  are functions discretized on a sparse grid and represented accordingly:

$$\hat{c} = \arg \min_{\alpha} \left( \frac{1}{M} \sum_{i=1}^M \left( y^{(i)} - \sum_{j=1}^N \alpha_j \phi_j(x^{(i)}) \right)^2 + \lambda \sum_{j=1}^N \alpha_j^2 \right).$$

Note, that for better readability the level-index  $l$  is omitted here and we now simply enumerate all grid points, basis functions  $\phi_j$  and surpluses  $\alpha_j$  by the index  $j \in \{1..N\}$ . We now minimize with respect to the hierarchical surpluses  $\alpha_j$ , which are the defining factor for  $\hat{c}$ . When minimizing with respect to coefficients a common choice for  $R(f)$  is  $\sum_j \alpha_j^2$ . Compared to the gradient the expressions in the following get simplified using this regularization [2].

### B. Matrix formulation

Minimizing with respect to alpha leads to a system of linear equations that can be formulated using the matrix expression

$$\left( \frac{1}{M} B B^T + \lambda I \right) \vec{\alpha} = \frac{1}{M} B \vec{y}.$$

The matrix

$$\mathbb{R}^{N \times M} \ni B = \begin{bmatrix} \phi_1(x^{(1)}) & \dots & \phi_1(x^{(M)}) \\ \vdots & \ddots & \vdots \\ \phi_N(x^{(1)}) & \dots & \phi_N(x^{(M)}) \end{bmatrix}$$

sets the grid points in relation to the data points in  $X$ . By solving this positive definite system we obtain the unique surplus-vector  $\vec{\alpha}$  defining  $\hat{c}$  [5].

The matrix product  $B B^T \in \mathbb{R}^{N \times N}$  demonstrates that the number of freedoms is not dependent on the number of data points. Thus, the computational effort scales only linearly in the number of data points  $M$  [2]. Because the matrix  $B$  might have a lot of entries and is not sparse, methods like Conjugate Gradients can be used to efficiently solve this system of linear equations [5].

### C. Example datasets

## IV. SOMETHING SOMETHING IMPLEMENATION

•

## V. CONCLUSION

•

- [1] H. Bungartz, D. Pflüger, and S. Zimmer, "Adaptive sparse grid techniques for data mining," *Modelling, Simulation and Optimization of Complex Processes*, 2006.
- [2] D. Pflüger, "Spatially adaptive sparse grids for high-dimensional problems," Ph.D. dissertation, Technische Universität München, 2010.
- [3] B. Peherstorfer, "Model order reduction of parametrized systems with sparse grid learning techniques," Ph.D. dissertation, Technische Universität München, 2013.
- [4] H. Bungartz and M. Griebel, "Sparse grids," *Acta Numerica*, pp. 1–43, 2004.
- [5] A. Heinecke, "Boosting scientific computing applications through leveraging data parallel architectures," Ph.D. dissertation, Technische Universität München, 2013.