Data Mining with Sparse Grids

Seminar: Computational Aspects of Machine Learning

Sebastian Kreisel
Department for Informatics
Technische Universität München
Email: sebastian.kreisel@tum.de

***Abstract*—516.0ptTODO: Abstract TODO: Abstract**

***Index Terms*—Sparse grids; Data mining; Hierarchical discretization; Curse of dimensionality**

I. Introduction

Large datasets and high dimensional data remain challenging aspects of data mining. Even with growing computational power, many problems require specialized algorithms to archive accurate results within the given time and cost restraints.

Sparse grids belong to a more general class of grid-based discretization methods. These methods are primarily applied to tackle scenarios with large amounts of data points and high–dimensional feature spaces [1], that pose the following problems:
Often, algorithms scale quadratic or worse with respect to the number of data points and thus quickly leading to time and cost related issues. High dimensionality introduces a problem widely known as the *Curse of Dimensionality*, denoting an exponential dependency between computational effort and the number of dimensions of the data [1], [2].
By focusing on carefully chosen grid points instead of the data points themself grid–based methods allow for better handling of large amounts of data. Furthermore, *sparse* grids specifically combat the curse of dimensionality and mitigate the exponential dependency.
Note also, that grid-based approaches are not applicable to data mining exclusively, but are also suited for a number of different areas including PDEs, model order reduction [3] or numerical quadrature [4].

In this paper the sparse grid technique is applied to data mining, investigating the mitigation capabilities to the previously mentioned issues. First, this paper introduces grid discretization and sparse grids in general as well as related topics like spatial adaptivity.
Then, sparse grids will be applied to machine learning through *least squares estimation*. To confirm the capabilities of sparse grids, the results of employing difficult test-datasets for regression and classification (e.g. checker-board dataset) will be discussed.
Lastly the efficient implementation on modern systems and parallelization of sparse grids will be examined.

II. Grid discretization

In machine learning, algorithms usually focus on a given training dataset X, for instance

$$X = \{x^{(i)} |\ x^{(i)} \in [0,1]^d\}_{i=1}^M \ , \quad Y = \{y^{(i)} |\ y^{(i)} \in \mathbb{R}\}_{i=1}^M$$

with, in case of supervised learning, an associated solution set Y [1].

Grid-based approaches introduce an additional set G consisting of N *grid points* with

$$G = \{1, 2, \dots, N\} \ .$$

For each dimension of the feature space a separate G (with possibly different N) is constructed dividing the space into a grid. This discretized space will then be used instead of directly working with the data points in the original feature space.

A. Full grid discretization

In the following, functions will be restricted to the unit hypercube

$$f : [0,1]^d \to \mathbb{R} \ .$$

To construct a *full* grid, we choose the grid points G equidistant without grid points lying on the borders.
We first consider the case of a one-dimensional f. Around each grid point i we center a one-dimensional *basis function*

$$\phi_i(x) = \max\{0, 1 - |(N+1)x - i|\} \ .$$

$\phi_i(x)$ is a standard hat function centered around i and dilated to have local support between the grid points $i-1$ and $i+1$. Fig. 1 shows $G = \{1, 2, \dots, 7\}$ and the related basis-functions.

To discretize a function $f(x)$ we introduce a coefficient (surplus) α_i for each grid point i. This coefficient is defined to be f evaluated at the grid point i

$$\alpha_i = f(\frac{i}{N+1}) \ .$$

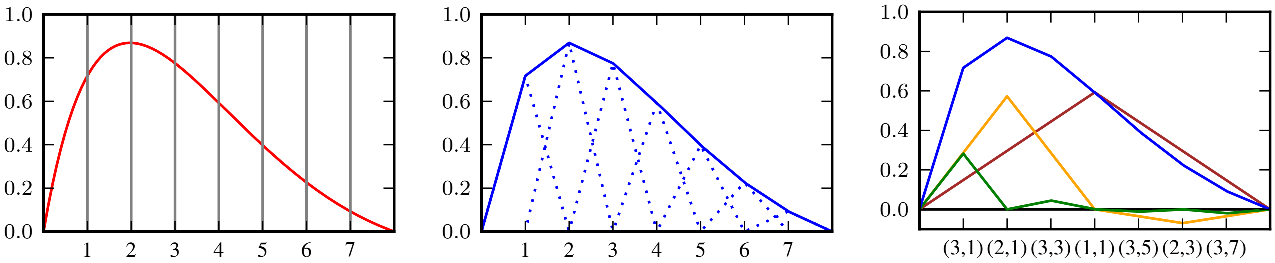

Fig. 1: A function f, red, to be discretized. Seven grid points discretizing the space (left). Full grid discretization by equidistant basis (middle) and hierarchical basis (right), scaled by α_i and $\alpha_{l,i}$ respectively.

Taking the sum

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

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

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

The related basis function gets extended to d dimensions using the tensor product

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

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 $l \in \{1, 2, \ldots, n\}$ be the *level* with $2^{(l-1)}$ associated grid points on each level. Through the level we group 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 functions

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

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

A grid point is now referred to as grid point of 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 ϕ and α.

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] (see Fig. 1).

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

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

describes 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 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 combination over the level in different dimensions leads to subspaces defined by the level-vector $\vec{l} = (l_x, l_y, \ldots)$ as illustrated in Fig. 2.

Taking the sum $\sum_{l,i} \alpha_{l,i} \phi_{l,i}(x)$ over all

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

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

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

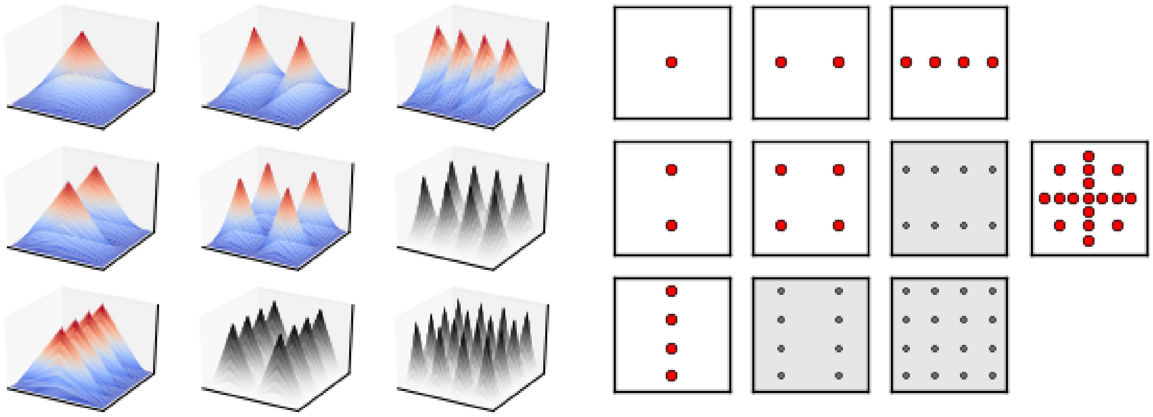

Fig. 2: Hierarchical subspaces (left) and corresponding grid points (right) with the last column being the complete sparse grid. For the sparse grid omitted subspaces/grid points are colored grey.

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 approximation of f. Which grid points that are is a *a-priori* solvable optimization problem [2]. Thus, independent of f all $\phi_{l,i}$ related to the subspaces in the lower right of the diagonal in Fig. 2 will be left out of the sum from Sec. II-A [1], [2]:

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

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 grids offer an optimal choice for a general function f, the discretization error is 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 cannot 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 contribute to the discretization of f and choose those that contribute most. Or we use information about f itself, for example the location of 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 already 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 newly added grid points to ensure consistency required by many algorithms on the grid [2]. Fig. 3 illustrates this refinement process for two points.

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.

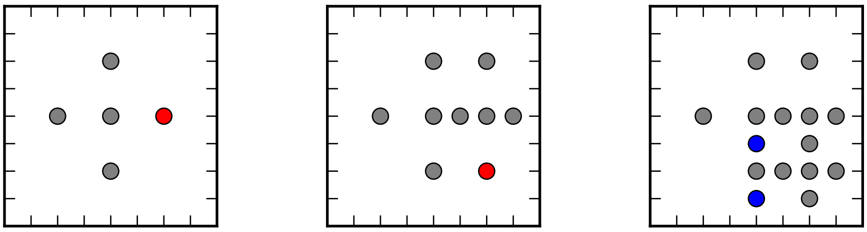

Fig. 3: Adative refinement of the grid points in red. One point of a regular sparse grid with $n = 2$ (left) gets refined (middle) followed by a second refinement, resulting in the grid on the right. The second refinement requires the addition of the blue colored hierarchical parents.

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:

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

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

B. Matrix formulation

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

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

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 $BB^T \in R^{N \times N}$ demonstrates that the number of freedoms is not dependent on the number of data points. Thus, the computational effort scales only linearly with respect to 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. Testing sparse grids

In order to get a good solution, several factors have to be determined before sparse grids can be employed. The regularization parameter λ and the amount of refinement steps can be determined using cross-validation. A straightforward way to refine grid points is to choose candidates with high surplus α. For more details on the refining strategy refer to [2].

First we examine the results of the checker-board dataset, a highly non-linear classification test set (see Fig. 4). While containing no noise, the checker-board-structured distribution of points contains very steep areas, which are difficult to model. Starting with a maximal level of $n = 4$ (49 grid points) the grid begins to capture the structure of the data with approximately 70% correct classifications. With a level of $n > 7$ (796 grid points) more than 98% are classified correctly [2]. This demonstrates the ability of sparse grids to model difficult non-linear datasets.

The second problem we examine is a real world regression task. A phenomenon similar to the doppler-effect called cosmological redshift is modeled. The training data consists of photometric measurements from known astronomical objects. By applying regression the redshift of newly observed objects is to be estimated using sparse grids. The dataset available contains 60,000 data points with 5 dimensions. Even though the shape of the data is inherently difficult to model, adaptive refinement helps to capture the core features. In contrast to the previous example, drastically more grid points are required to archive a low mean squared error. At approximately 20,000 grid points the error drops considerably. For more detailed examination refer to [2].

Those two examples demonstrate that sparse grids are able to cope with non-linear, large and moderately high dimensional datasets. As previously mentioned, spatial adaptivity is crucial

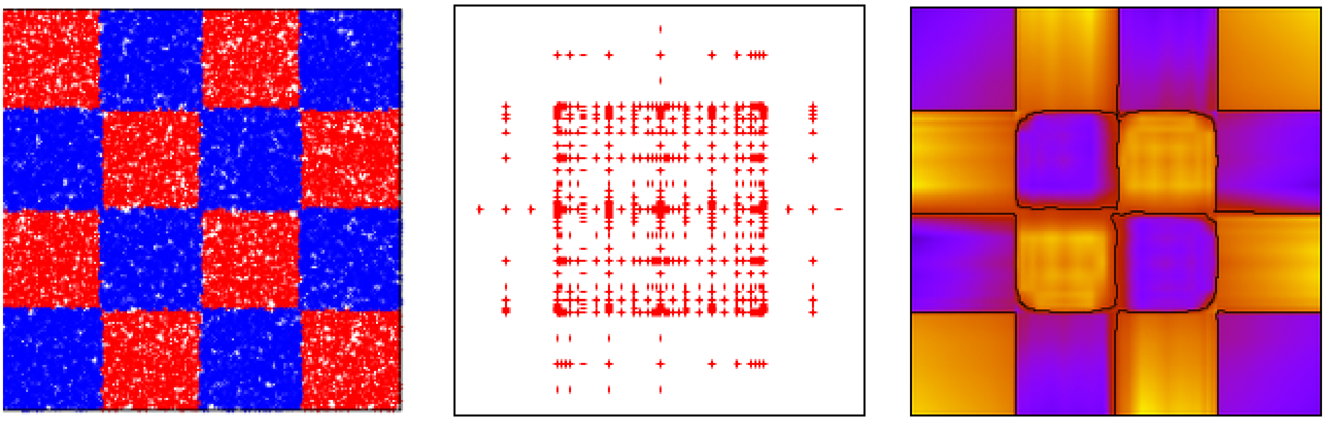

Fig. 4: The checker-board test set. 40,000 sampled data points with $Y = \{-1, 1\}$ (colored red and blue respectively) used as training data (left). The refined sparse grid with 298 grid points (middle) and the projected decision manifold (right) showing high accuracy, especially towards the borders [2].

to capture the features of the problem and enables sparse grids to be a competitive method for data mining and machine learning.

IV. Implementation of Sparse Grids

Many real-world problems require a large number grid points as mentioned in Sec. III-C. Together with an enormous number of data points and high dimensionality, a lot of raw computational power is required.

V. Conclusion

-

References

[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.