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 amount of data points and high–dimensional feature spaces, posing these problems:
Often, algorithms scale quadratic or worse in 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.
By focusing on grid points instead of the data points them self grid–based methods allow for better handling of large amounts of data. 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 PDA, model order reduction or numerical quadrature.

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

Grid-based approaches introduce an additional set G 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 divinding the space into a grid. This, by the grid *discretized*, space will then be used instead of working with the datapoints in the original feature space directly.

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 chose the grid points G equidistant, without grid points lying on the borders.
We first consider the case of a one-dimensional f being discretized. Around each gridpoint 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

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

Taking the sum

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

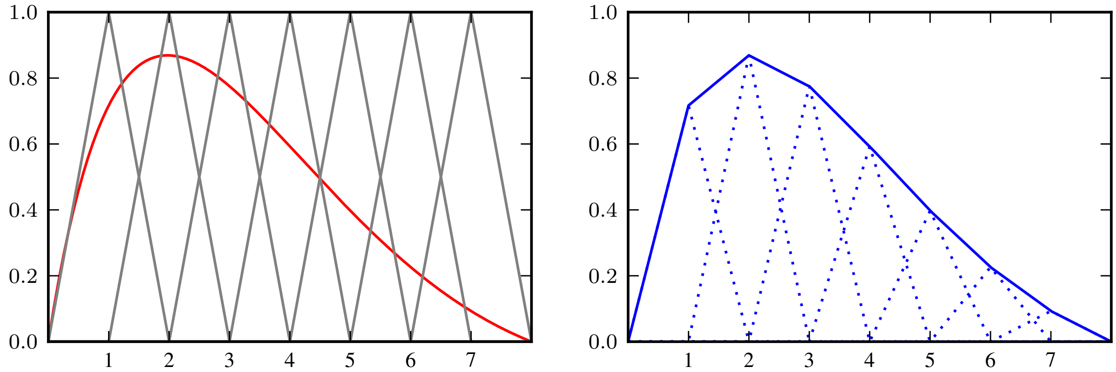

Fig. 1: A function f, red, to be discretized. Seven grid points discretizing the space and their associated basis functions (left). By α_i weighted basis functions, dashed, and sum thereof (right) discretizing f.

over all weighted basis-functions ϕ_i discretizes (approximates) f. 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 $x = 1,\ y = 3,\ z = 3$ 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. 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 as described in Sec. II-A, more sophisticated methods are available. In order to make a grid sparse and still keep a sufficient accuracy the following *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. 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.

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

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

discretizes f on a full grid. In contrast to the conventional approach from Sec. II-A the hierarchical surpluses now are calculated using adjacent gird points $i-2$ and $i+2$ such that

$$\alpha_i = f\big(t(i)\big) - \frac{1}{2}\Big(f\big(t(i-2)\big) + f\big(t(i+2)\big)\Big)\ .$$

To transfer from grid-index i with corresponding level l to x the function $t(i) = 2^{-l} i$ is used.

For $d > 1$ we combine one the 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 as shown for $d = 2$ in FIGURE. This process of building d-dimensional basis functions leads to a hierarchy of subspaces which, summed up and weighted, discretize f.

However, this does not lead to a sparse gird immediately. So far the gridpoints 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.

C. *Sparse grid discretization*

In order to make the hierarchical grid *sparse*, we now disregard certain grid points with their associated subspaces defined by the combinations of $\phi_{i,l}$ in d dimensions. FIGURE illustrates this.
Which gridpoints contribute the least to the grid is a *a-priori* solvable optimization problem. Thus, independend of f all $\phi_{(l,i)}$ related to the subspaces in the lower right of the diagonal in FIGURE will be left out of the sum

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

from FIGURE.

III. Sparse Grids in Machine Learning

- Quick note on classification/regression
- Least squares

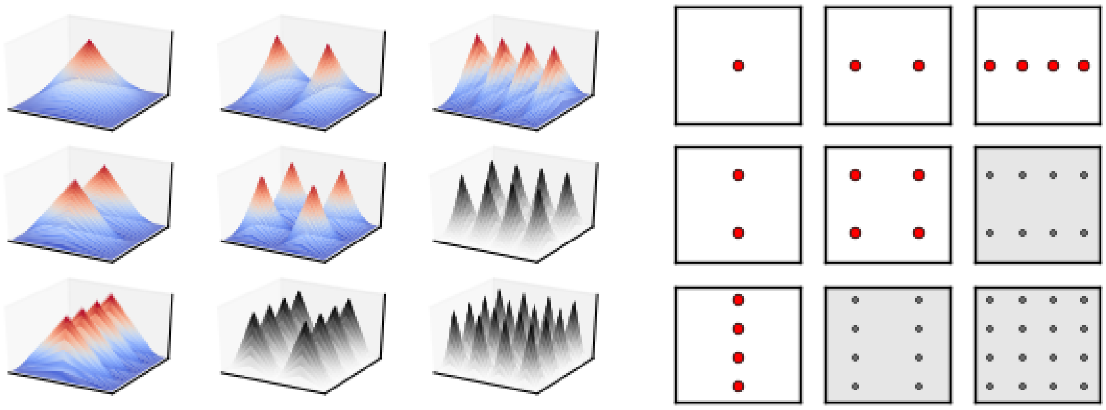

Fig. 2: .

- Least squarse with sparse grids
- Matrix formulation
- Notes on matrix solving etc.

IV. Something something Implemenation

-

V. Conclusion

-

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.