Data Mining with Sparse Grids

Seminar Computational Aspects of Machine Learning

Sebastian Kreisel
Department for Informatics
Technische Universitt Mnchen
Email: sebastian.kreisel@tum.de

***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 problems including a large amount of datapoints and high–dimensional feature spaces:
Often, algorithms scale quadratic or worse in the number of datapoints 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 gridpoints instead of the datapoints them self, grid–based methods are able to deal much better with large datasets. Making the grids *sparse*, combats the curse of dimensionality while losing accuracy.

Grid–based methods are not restricted to data mining but are also well 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 mitigation 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 modified *least square estimation*. To confirm the capabilities of sparse grids widely applied, difficult, test–datasets for regression and classification like the *checker–board* dataset will be used.

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 trainings–data set 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 an associated solution–set Y in case of supervised learning.

Grid–based approaches introduce an additional set G of N *gridpoints* with

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

For each dimension of the feature space a separate G (with possibly different N) is constructed forming a grid over the feature space. This, by the grid discretized space, will then be used instead of working with the datapoints in the original feature space directly.

To examine sparse grids the following section will first introduce full grid discretization.

A. Full grid

To construct a *full* grid we chose the gridpoints G equidistant, without gridpoints lying on the borders. Further, the space to be discretized is assumed to be a unit–hypercube.

First, let $d = 1$. 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 the standard hat–function, centered around i and dilated to have local support between the gridpoints $i-1$ and $i+1$. **??** 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 gridpoint i. This coefficient is defined to be f evaluated at the gridpoint i

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

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. **??** illustrates this.

For $f(\vec{x})$ with $d > 1$, each dimension j is treated separately using the above scheme. This results in d one–dimensional functions $\hat{f}_j(x_j)$ with x_j denoting the j–th element in the vector $\vec{x}$. Taking the tensor product

$$f(\vec{x}) \approx \hat{f}(\vec{x}) = \prod_{j=1}^{d} \hat{f}_j(x_j)$$

approximates $f(\hat{x})$.

B. *Hierachial basis functions*

- Basic notion
- Hierachial surplus
- Hierachial subspaces

C. *Sparse grid*

- Disregarding subspaces
- Trade-off
- Spartial adaptivity
- Boundry and smoothness note

III. Sparse grids in Machine Learning

- Quick note on classification/regression
- Least squares
- Least squarse with sparse grids
- Matrix formulation
- Notes on matrix solving etc.

IV. Something something Implemenation

-

V. Conclusion

-