Data Mining with Spare Grids

Seminar: Computational Aspects of Machine Learning

Sebastian Kreisel

October 27, 2015

Technische Universität München

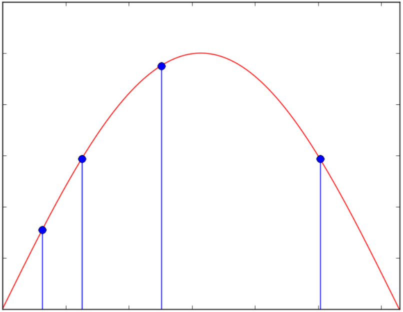
Overview

- Motivation
- Sparse Grids: Basics
- Sparse Grids: Data Mining
- Examples
- Implementation

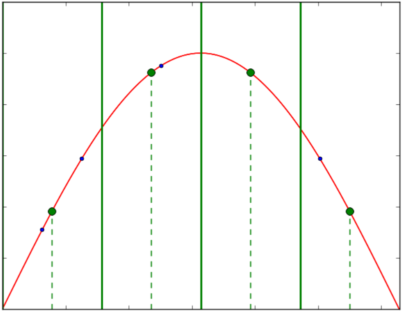
Motivation

Grid based approaches in ML

- Discretizes the space into a grid
- Basis functions around grid points, not data points



Point based



Grid based

Motivation

Suitable for

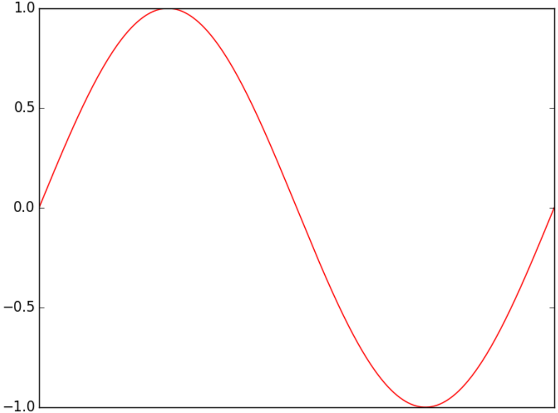
- Large, high-dimensional datasets
- Medical data, seismic data, finance, astrophysics

Curse of dimensionality

- The volume of a space is exponential in its dimensions
- The amount of training data required becomes unmanageable
  - because of lacking computational power and storage capacities
  - because data acquisition can be expensive
- Becomes relevant for $d > 3$

Full Grid Discretization

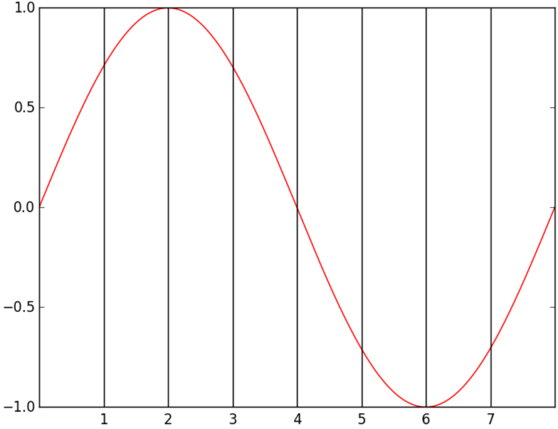
Function to discretize



$f(x) = \sin(2\pi x)$

Full Grid Discretization

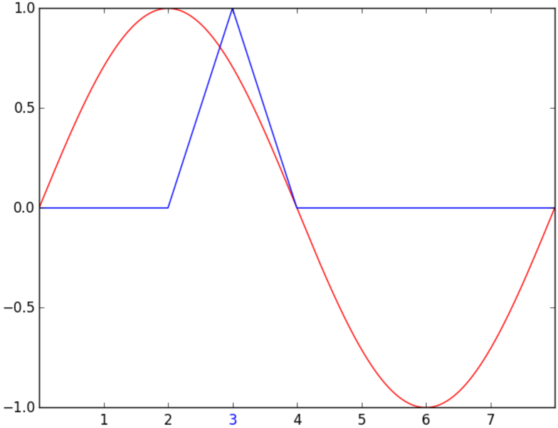
Full, regular grid



7 grid points $i \in \{1..7\}$

Full Grid Discretization

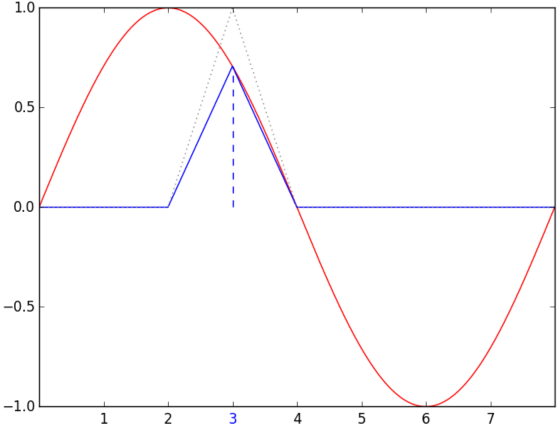
Basis functions (standard hat function)



$\phi(x-3)$ with $\phi(x) = \max(1-|x|, 0)$

Full Grid Discretization

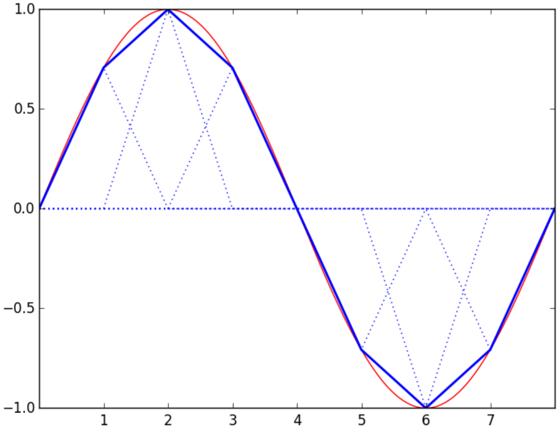
4. Coefficient α (surplus)



$\alpha_3 \cdot \phi_3(x)$

Full Grid Discretization

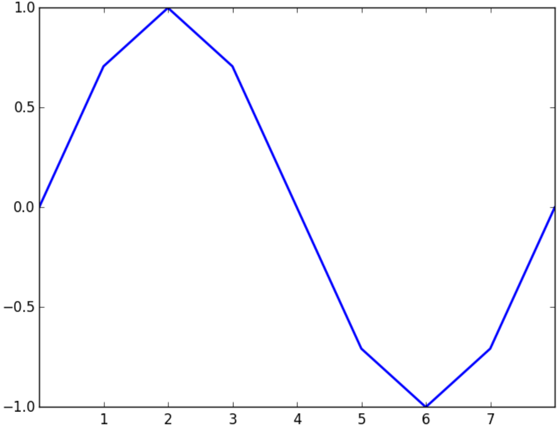
Sum over all weighted basis functions



$$f(x) \approx \hat{f}(x) = \sum_i \alpha_i \phi_i$$

Full Grid Discretization

Sum over all weighted basis functions



$$f(x) \approx \hat{f}(x) = \sum_i \alpha_i \phi_i$$

Full Grid Discretization

Full grid discretization in one dimension

1. A function $f(x)$ to discretize
2. Grid points indexed by $i \in \{1, 2, \ldots, N\}$
3. Basis functions; i.e. hat function $\phi(x) = \max(1 - |x|, 0)$
4. Coefficients α_i (surpluses)

$$f(x) \approx \hat{f}(x) = \sum_i \alpha_i \phi_i(x)$$

Full Grid Discretization

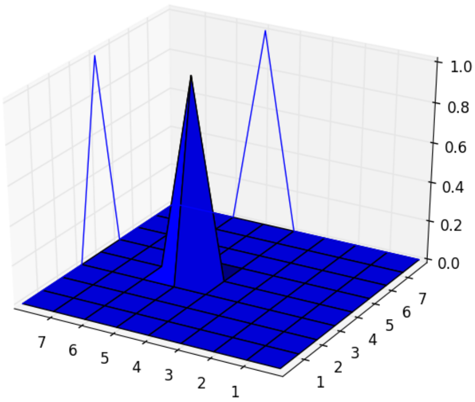
$d > 1$ dimensions

- Grid points as index vector $\vec{i}$, i.e. $(1, 3, 1)$
- Tensor product over the one dimensional basis functions

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

Full Grid Discretization

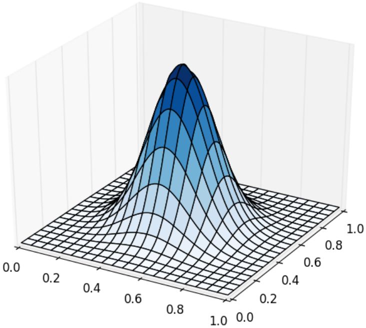
Tensor product for $d = 2$



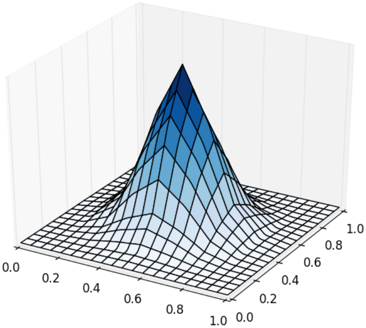
$\phi_5(x_1) \cdot \phi_4(x_2)$

Full Grid Discretization

Full grid with $d = 2$



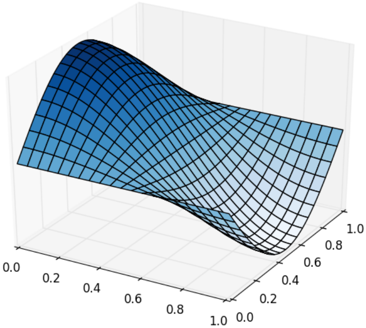
$f(x) = \mathcal{N}(\vec{x})$



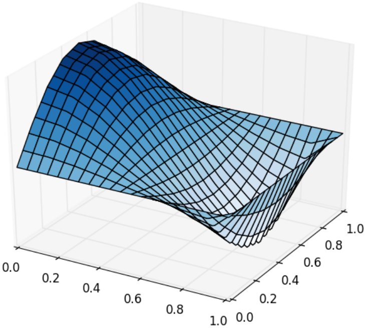
Full grid discretization $\hat{f}(\vec{x})$

Full Grid Discretization

Full grid with $d = 2$



$f(x) = \sin(x_1) \cdot \cos(x_2)$



Full grid discretization $\hat{f}(\vec{x})$

Full Grid Discretization

Summary

- **Grid points** $\vec{i} \in \{1, 2, \ldots, N\}^d$ defining $\phi_i(x)$
- **Product** of 1D basis functions:

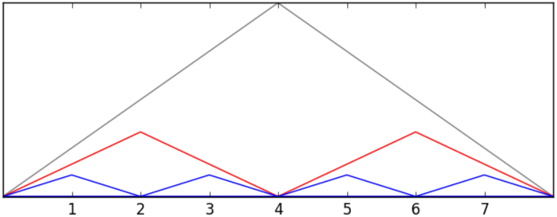
$$\phi_i(\vec{x}) = \prod_{j=1}^{d} \phi_{i_j}(x_j)$$

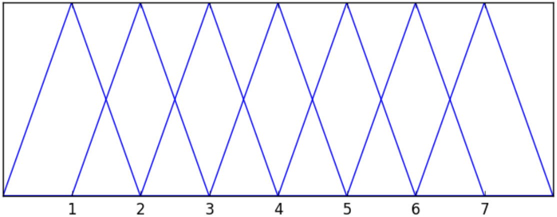
- **Sum** over all weighted basis functions:

$$\hat{f}(\vec{x}) = \sum_{i=1}^{N} \alpha_i \phi_i(\vec{x})$$

Sparse Grids – Basics

Hierarchical vs. nodal basis





Sparse Grids – Basics

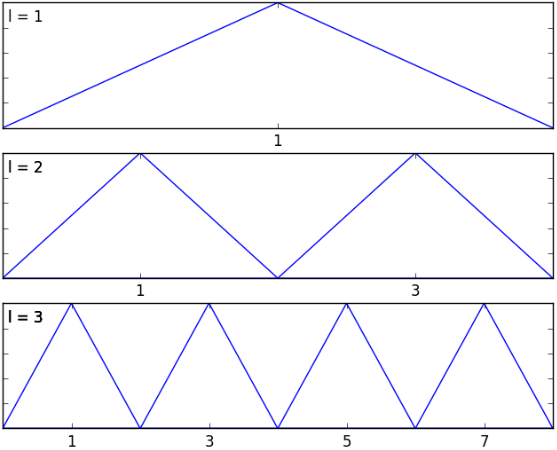
Hierarchical basis

- Grouping grid points into levels $l \in \{1, 2, 3 \ldots, n\}$
- Basis function by index **and** level: $\phi_{l,i}(x)$

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

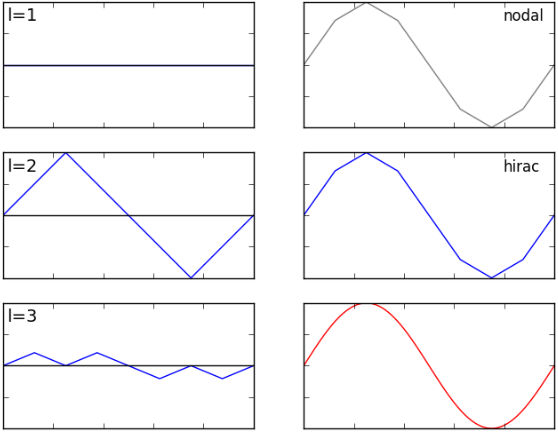
Sparse Grids – Basics

Hierarchical Basis



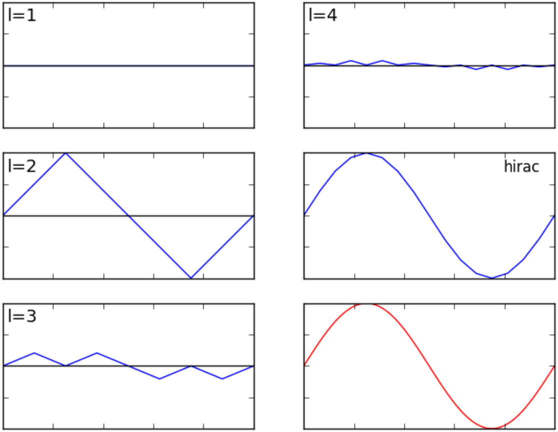
Sparse Grids – Basics

Discretization with hierarchical basis



Sparse Grids – Basics

Discretization with hierarchical basis



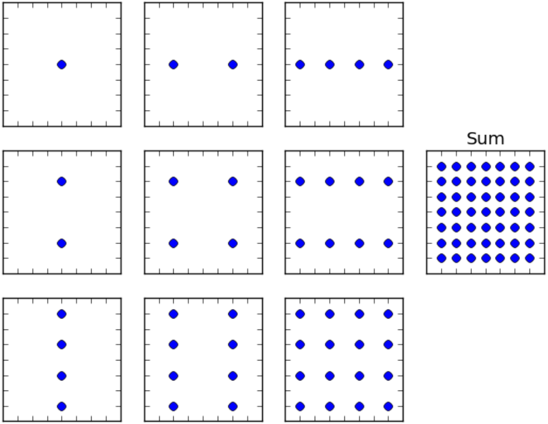
Sparse Grids – Basics

Hierarchical basis for $d > 1$

- Index and level vector $\vec{i}$, $\vec{l}$
- Combination of levels in all dimensions
- Notion of subspaces

Sparse Grids – Basics

Hierarchical basis: Grid points



Sparse Grids – Basics

Sparse grid – Changes

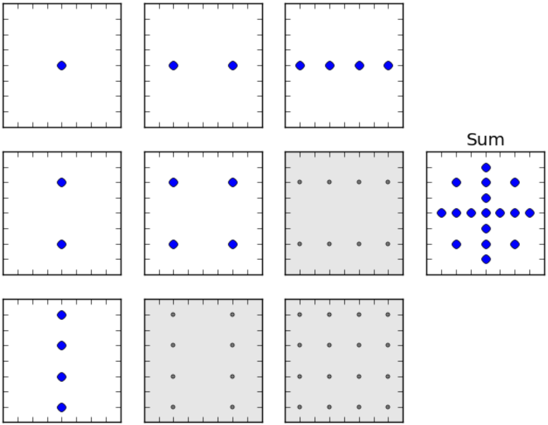
- Certain subspaces get disregarded
- Finding those is a *a-priori* solvable optimization problem
- The resulting grid is **sparse**

Profit

- Reducing the computational effort "a lot"
- Maintaining "high" accuracy

Sparse Grids – Basics

A sparse grid



Sparse Grids – Basics

A sparse grid

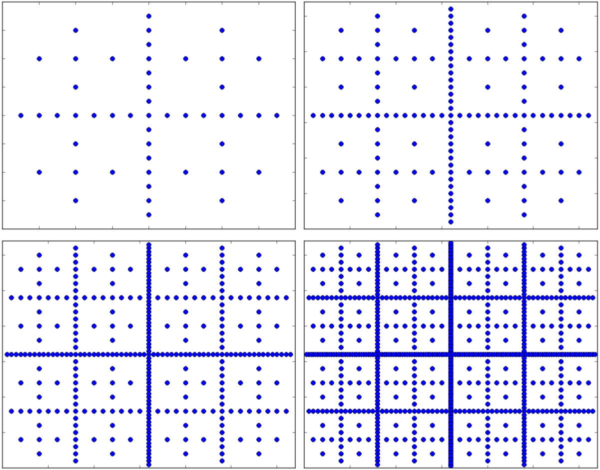
d	1	2	3	5	10	20
Full	15	225	3375	$> 10^5$	$> 10^{11}$	$> 10^{23}$
Sparse	15	49	111	351	2001	13201

Full grid vs. sparse grid with $n = 4$

- Massive difference in high dimensions
- Note: No boundary grid points!

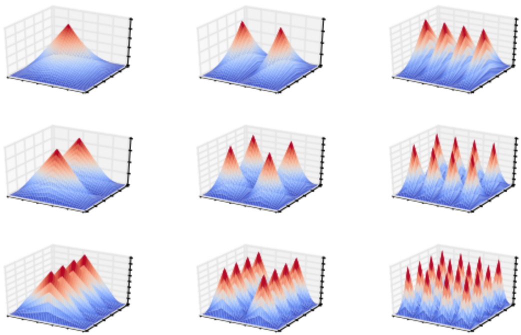
Sparse Grids – Basics

Sparse grid for $n = \{4, 5, 6, 7\}$



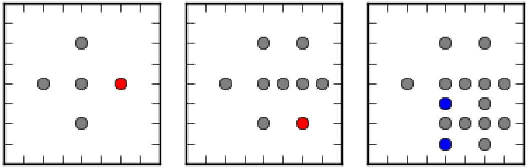
Sparse Grids – Basics

Hierarchical subspaces



Sparse Grids – Basics

Adaptivity



- *A-posteriori* modifications to model the function better
- Adding level of detail around a grid point: $l + 1$
- Overfitting possible and considerable computational effort
- Multiple strategies possible

Sparse Grids – Basics

Summary

- Hierarchical basis through grouping grid points into levels
- Creating subspaces through combination of levels in dimensions
- Selecting only certain subspaces
- Adaptive refinement

$$\hat{f}(\vec{x}) = \sum_{l,i} \alpha_{l,i} \phi_{l,i}(\vec{x})$$

Sparse Grids – Data Mining

Machine learning tasks

- Classification
- Regression

Training-data

$$X = \{x^{(j)} \mid x^{(j)} \in [0,1]^d\}_{j=1}^{M}$$

$$Y = \{y^{(j)} \mid y^{(j)} \in \mathbb{R}\}_{j=1}^{M}$$

Sparse Grids – Data Mining

Least squares

$$\hat{c} = \arg\min_f \left(\frac{1}{M} \sum_{j=0}^{M} \left(y^{(j)} - f(x^{(j)}) \right)^2 \; + \; \lambda R(f) \right)$$

Sparse Grids – Data Mining

Sparse grid setting

- Do least squares in a sparse grid setting
- Discretize $\hat{c}$ using a sparse grid

$$\hat{c} = \arg\min_{\alpha} \left(\frac{1}{M} \sum_{j=0}^{M} \left(y^{(j)} - \sum_i \alpha_i \phi_i(x^{(j)}) \right)^2 + \lambda \sum_i \alpha_i^2 \right)$$

Sparse Grids – Data Mining

Matrix formulation

$$\left(\frac{1}{M}BB^T + \lambda I\right)\alpha = \frac{1}{M}By$$

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

Sparse Grids – Data Mining

Observations

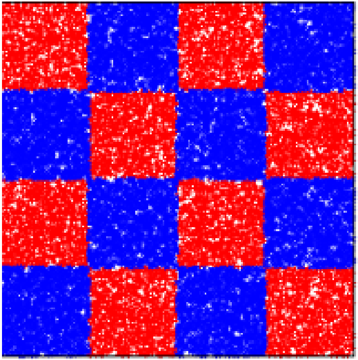
- $BB^T \in \mathbb{R}^{N \times N}$ where $N =$ number of grid points
- Number of freedoms not dependent on M
- Linear scaling in M

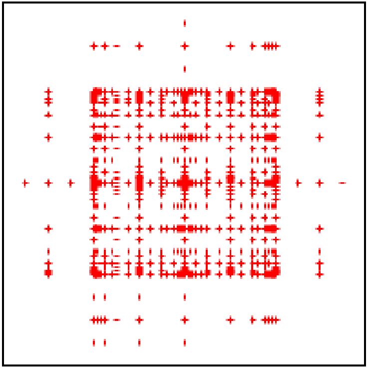
Solving the system of linear equations

- The SLE is not sparse but positive definite
- Numerical methods like Conjugate Gradients (GC)

Sparse Grids – Examples

Checkerboard dataset: non-linearity



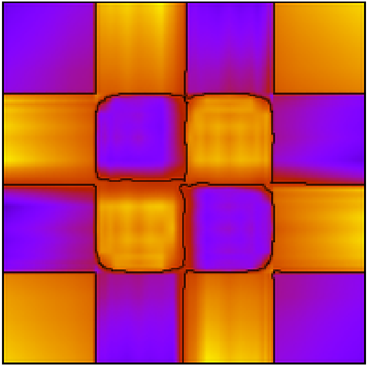


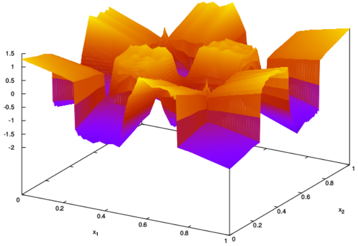
Checkerboard dataset and sparse grid (293 grid points) after 40 refinements

(Dirk Pflüger – Spatially Adaptive Sparse Grids for High-Dimensional Problems)

Sparse Grids – Examples

Checkerboard dataset: non-linearity



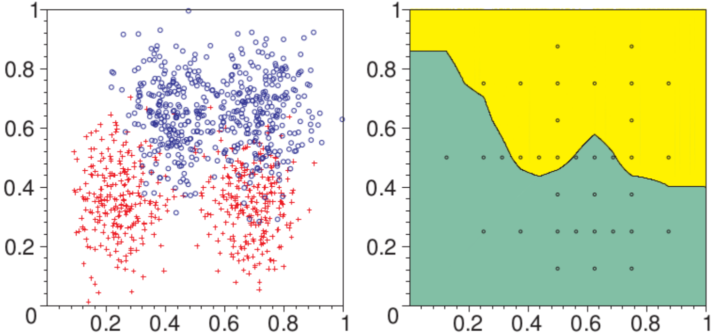


Decision manifold

(Dirk Pflüger – Spatially Adaptive Sparse Grids for High-Dimensional Problems)

Sparse Grids – Examples

Ripely dataset: noise and overfitting



Ripely dataset and sparse grid (34 grid points) after 8 refinements

(Dirk Pflüger – Spatially Adaptive Sparse Grids for High-Dimensional Problems)

Sparse Grids – Implementation

Computational efficient implementation

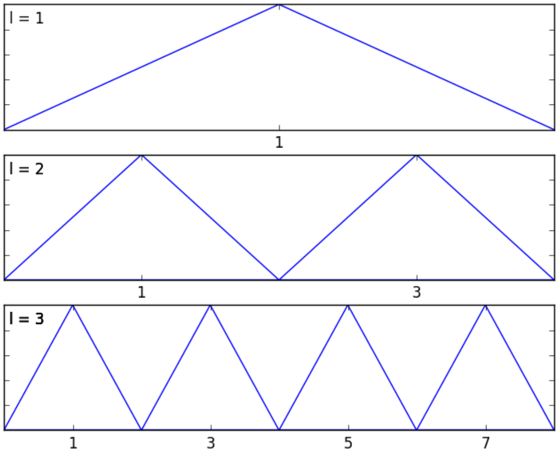
- Tensor-product structure
- Nested structure
- Recursive, tree-like traversal

Problems

- Scattered memory access patterns
- Inherently recursive, hard to parallelize

Sparse Grids – Basics

Hierarchical, nested structure



Sparse Grids – Implementation

Iterative implementation

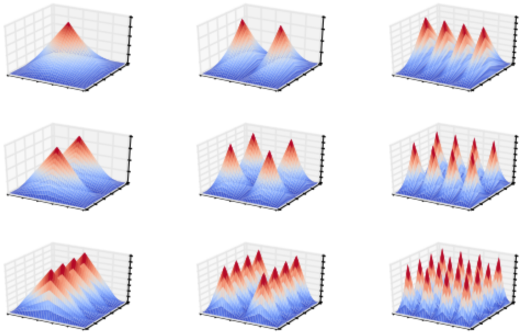
- Disregards the hierarchical, nested structure
- Calculates $\phi_i(x^{(j)})$ for every ϕ_i and $x^{(j)}$
- 3 for-loops: data points, grid points, dimensions

Trade-off

- Parallelization/Vectorization
- Linear memory access allowing pre-fetching
- Many (!) unnecessary computations (zero-evaluations)

Sparse Grids – Implementation

Hierarchical subspaces



Summary

Data mining with sparse grids

- Discretization with hierarchical basis
- Sparseness and adaptivity
- Classification/Regression through least squares
- Iterative implementation to exploit parallelization

Conclusions

- Linear scaling in the number of data points
- Mitigation of the curse of dimensionality
- Robust technique capable of dealing with non-linear data

Backup

Backup

Number of grid points vs. accuracy

- Holds only if $D^2 f$ is bounded
- Maximal level n, maximal dimension d

- Number of grid points:

$$\mathcal{O}(2^{nd}) \quad \rightarrow \quad \mathcal{O}(2^n \cdot n^{d-1})$$

- Error:

$$\mathcal{O}(2^{-2n}) \quad \rightarrow \quad \mathcal{O}(2^{-2n} \cdot n^{d-1})$$

Backup

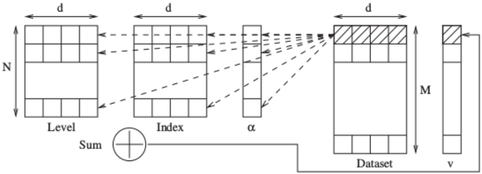
Implementation

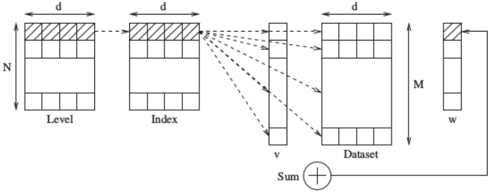
$$\left(\frac{1}{M}BB^T + \lambda I\right)\vec{\alpha} = \frac{1}{M}B\vec{y}$$
$$\equiv \lambda I\vec{\alpha} + \frac{1}{M}B(B^T)\vec{\alpha} = \frac{1}{M}B\vec{y}$$

- $\vec{v}_j = (B^T\vec{\alpha})_j = \hat{f}(x^{(j)})$
- $\vec{w} = B\vec{v}$

Backup

Implementation





(Alexander F. Heinecke – Boosting Scientific Computing Applications through Leveraging Data Parallel Architectures)