Data Mining with Spare Grids

Seminar: Computational Aspects of Machine Learning

Sebastian Kreisel
October 4, 2015

Technische Universität München

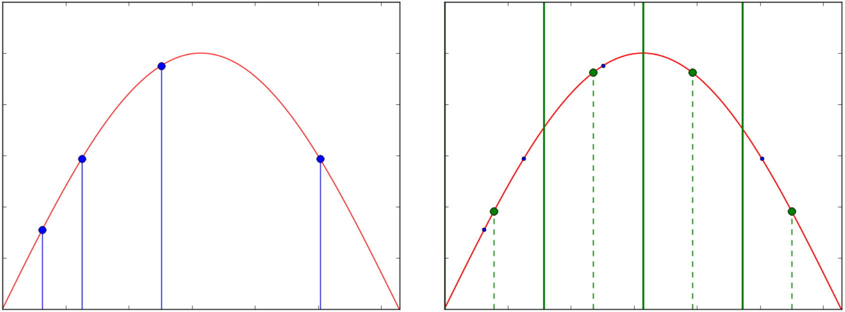
Overview

- Motivation for Sparse Grids
- Sparse Grids: Basics
- Sparse Grids: Machine Learning
- Examples with Data Sets
- Parallelization and Implementation

Motivation for Sparse Grids

Grid based approaches in ML

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



Point based Grid based

Motivation for Sparse Grids

Suitable for

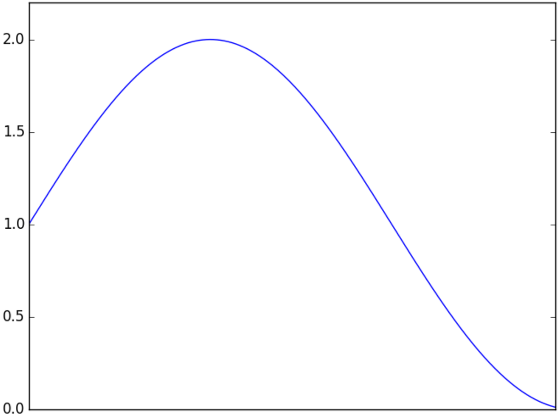
- Big datasets
- Easily/automatically classifiable data
- Medical, seismic, commercial data

Curse of dimensionality

- The volume of a space is exponential in it's dimensions
- The amount of training data required becomes unmanageable
  - because of lacking computational/storage capacities
  - because data aquesition is expensive
- Becomes relevant for $d > 3$
- **Applies to full-grid discretization**

Full Grid Discretization

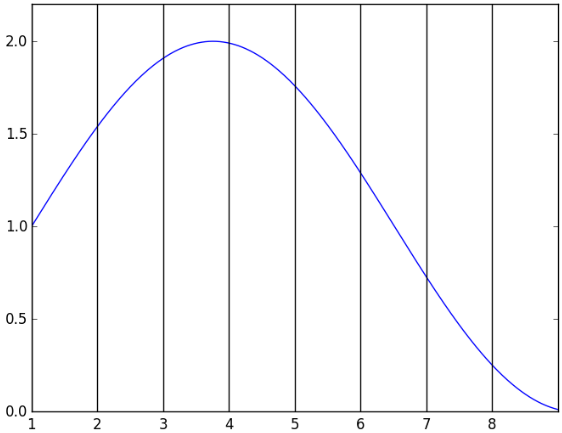
1. Function to discretize



$$f(x) = 1 + \sin(1.45\pi \cdot x)$$

Full Grid Discretization

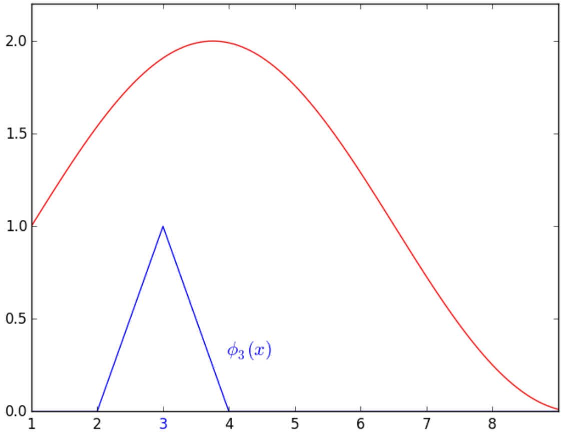
2. Full, regular grid



Eight centered gridpoints $i \in \{1..8\}$

Full Grid Discretization

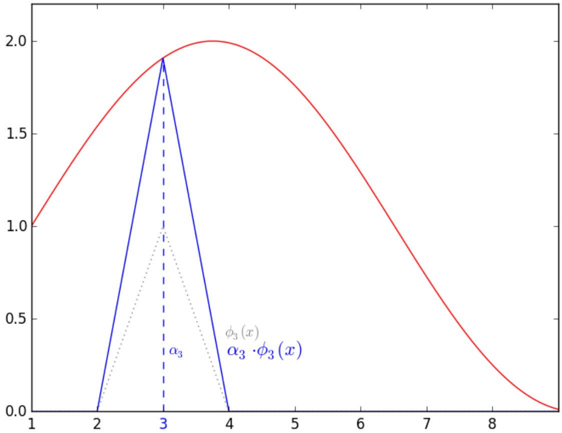
3. Basis function (standard hat function)



$\phi(x-2)$ with $\phi(x) = \max(abs(1-x), 0)$

Full Grid Discretization

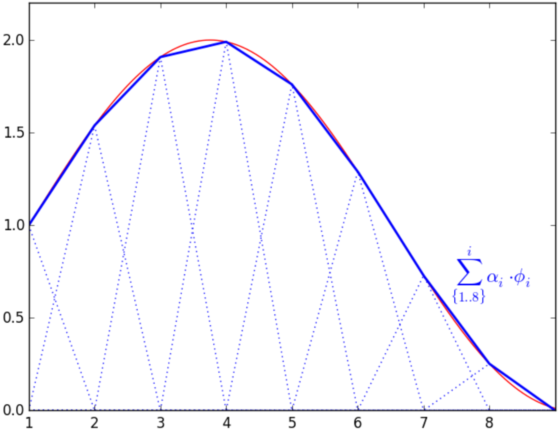
4. Coefficient α (Surplus)



$\alpha_3 \cdot \phi_3(x)$

Full Grid Discretization

Sum over all 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. Gridpoints indexed by $i \in \{1, 2, \dots\}$
3. Basis/Ansatz functions; i.e. hat function $\phi(x) = \max(1 - |x|, 0)$
4. Coefficients α_i (Surplusses)

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

Full Grid Discretization

Level of detail

- Level $l \in \{1, 2, 4, 8 \dots\}$ defining the number of gridpoints i

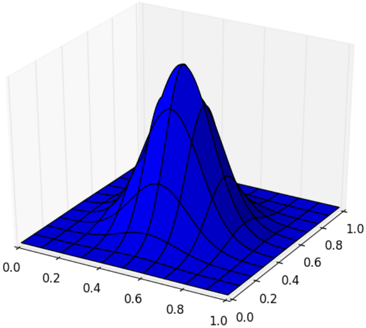
$d > 1$ dimensions

- For each dimension $\hat{f} = \sum \dots$
- Tensor product over all dimensions $\prod \hat{f}(x)$

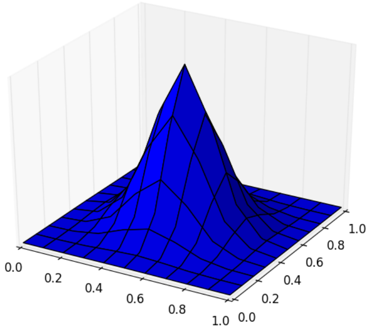
$$\underbrace{\sum_i \alpha_i \phi_i(x_1)}_{d=1} \quad \cdot \quad \underbrace{\sum_j \alpha_j \phi_j(x_2)}_{d=2} \quad \cdot \quad \dots$$

Full Grid Discretization

Full grid with $d = 2$



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



Full grid discretization $\hat{f}_1(x_1) \cdot \hat{f}_2(x_2)$

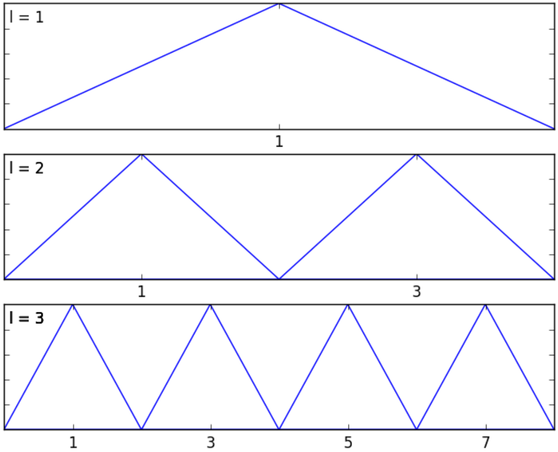
Full Grid Discretization

Summary

- Girdpoints $i \in \{1..2^l\}$ defining $\phi_i(x)$
- For *each* dimension: **sum over basisfunctions** $\hat{f}(x) = \sum_i \alpha_i \phi_i(x)$
- For *all* dimensions: **product over dimensions** $\prod \hat{f}(x)$

Sparse Grids – Basics

Hirachial Basis



Sparse Grids – Basics

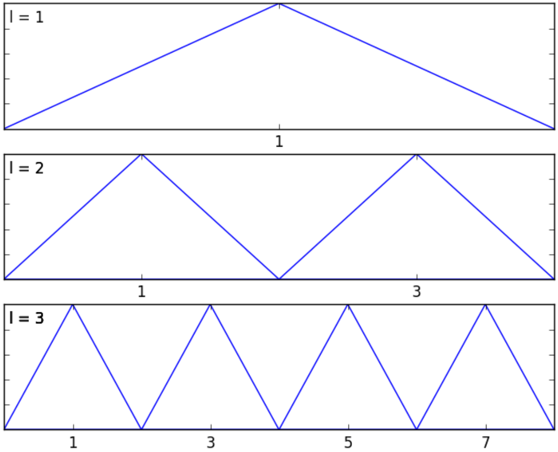
Hirachial basis (vs nodal basis)

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

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

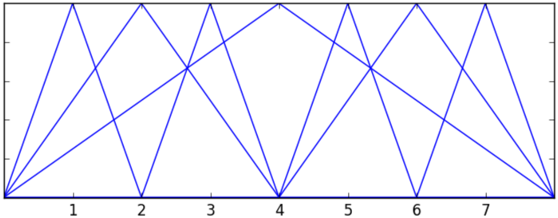
Sparse Grids – Basics

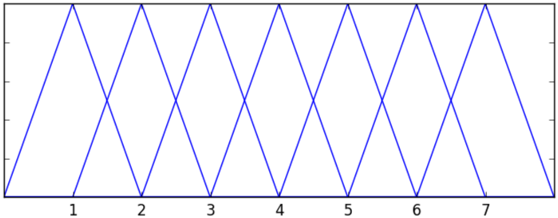
Hirachial Basis



Sparse Grids – Basics

Hirachial vs. nodal basis





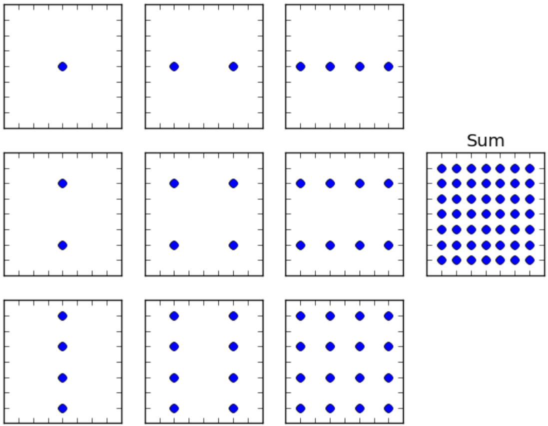
Sparse Grids – Basics

Basis function subspaces

- Combination of levels through all dimensions

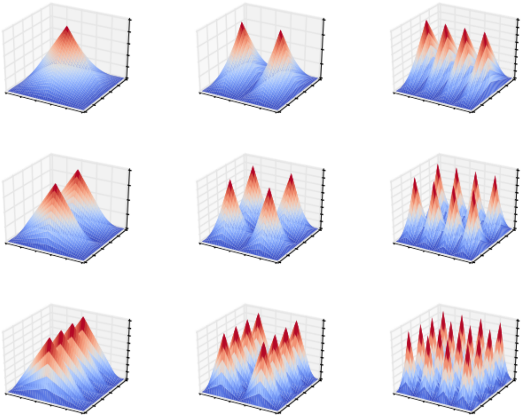
Sparse Grids – Basics

Hirachical gridpoints



Sparse Grids – Basics

Hirachical subspaces



Sparse Grids – Basics

Sparse grid – Changes

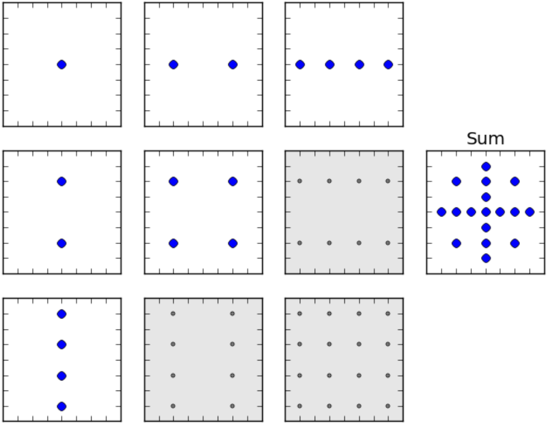
- Throwing away certain subspaces
- Finding those is a *a-priori* solvable optimization problem
- The resulting gird is now **sparse**

Profit

- Reducing the computational effort “a lot”
- Maintaining “high” accuracy

Sparse Grids – Basics

A *sparse* grid



Sparse Grids – Basics

Boundry and smoothness

- Boundries need special treatment
- The function needs to be sufficient smooth
  $D^2 f$ need to be bounded

Adaptivity

- *A-posteriori* modifications to better fit the function
- Picking a single gridpoint and adding level of detail around it
- Prone to overfitting and huge computational effort

Sparse Grids – Basics

Summary

- Hirachial basis through grouping gridpoints into levels
- Creating “subspaces” through combination of levels in dimensions
- Selecting and combining subspaces

To keep in mind

- Smoothness requirement for $f(x)$
- Boundry treatment
- Accuracy–effort trade-off
- Adaptivity options (*a-posteriori*)