Data Mining with Spare Grids

Seminar: Computational Aspects of Machine Learning

Sebastian Kreisel

October 13, 2015

Technische Universität München

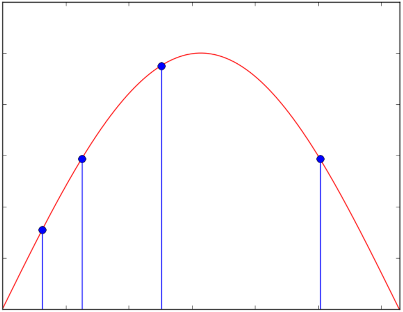
Overview

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

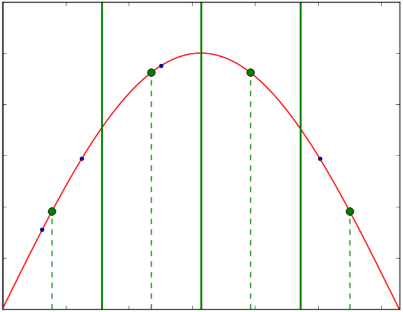
Motivation for Sparse Grids

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

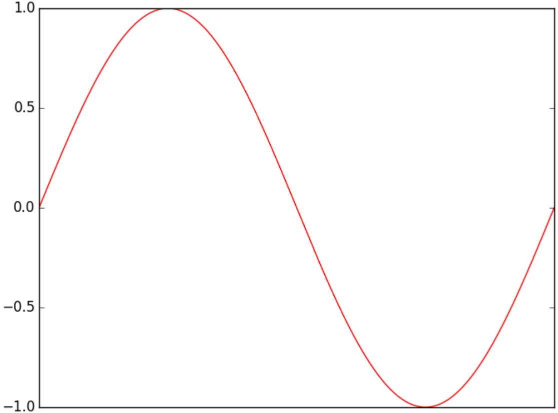
- Big datasets
- Easily/automatically classifiable data
- Medical, seismic, commercial data

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

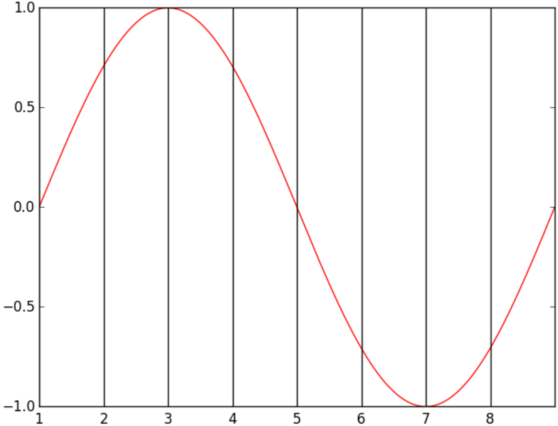
1. Function to discretize



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

Full Grid Discretization

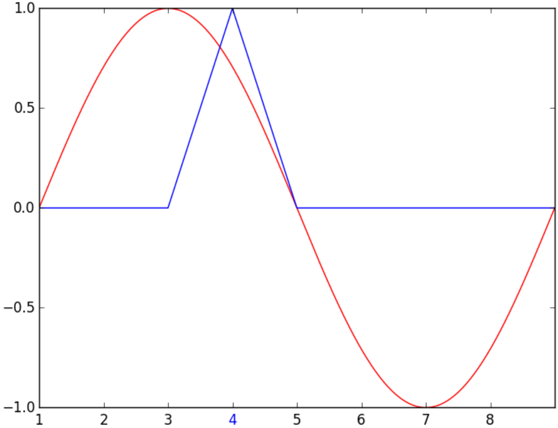
2. Full, regular grid



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

Full Grid Discretization

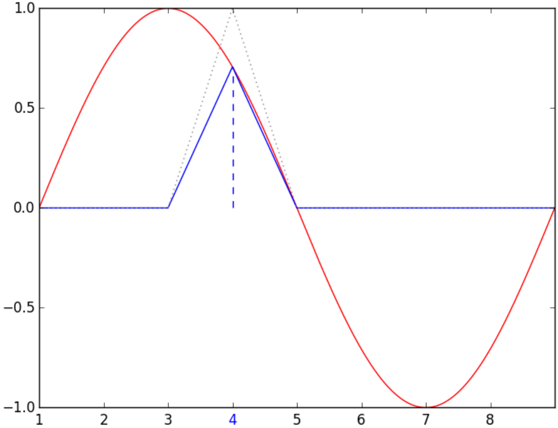
3. Basis function (standard hat function)



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

Full Grid Discretization

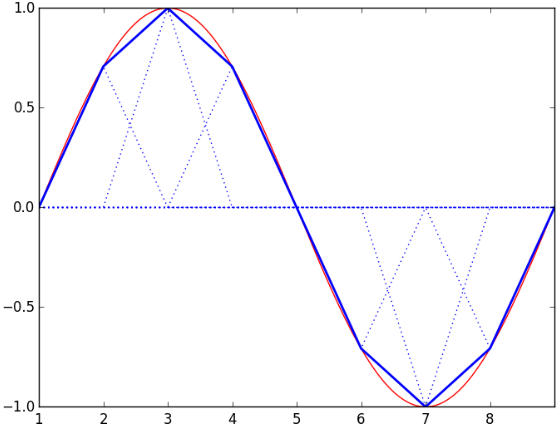
4. Coefficient α (Surplus)



$\alpha_4 \cdot \phi_4(x)$

Full Grid Discretization

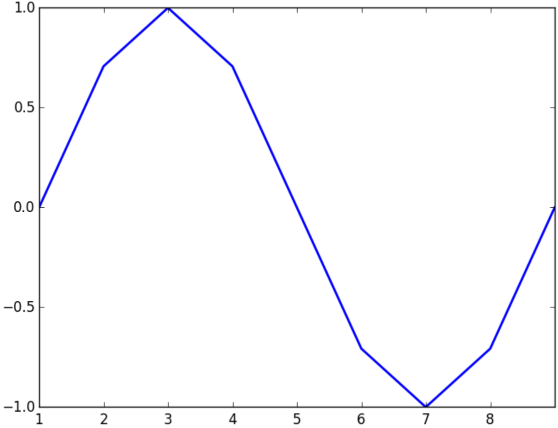
Sum over all basis functions



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

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

$d > 1$ dimensions

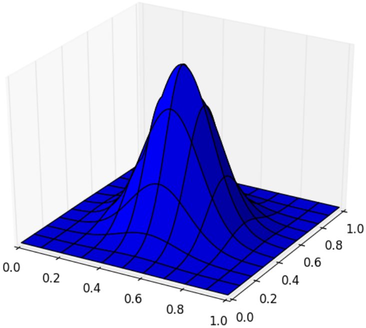
- Gridpoints as d–tuple, i.e. $(1, 3, 1)$
- Tensor product over one dimensional basis functions

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

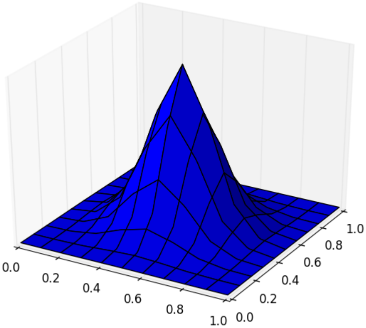
$$f(\vec{x}) \approx \hat{f}(\vec{x}) = \sum_i \alpha_i \phi_i(\vec{x})$$

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

- Gridpoints $i \in \{1, 2, \ldots, N\}^d$ defining $\phi_i(x)$
- For $d > 1$: **product** of 1D basis functions:

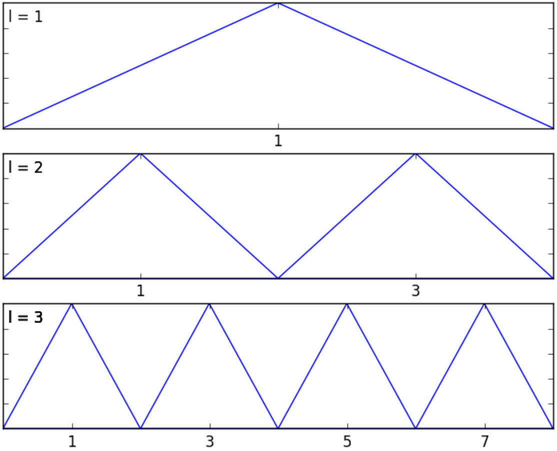
$$\phi_i(\vec{x}) = \prod_{j=1}^{d} \phi_{i,j}(x_j)$$

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

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

Sparse Grids – Basics

Hirachial Basis



Sparse Grids – Basics

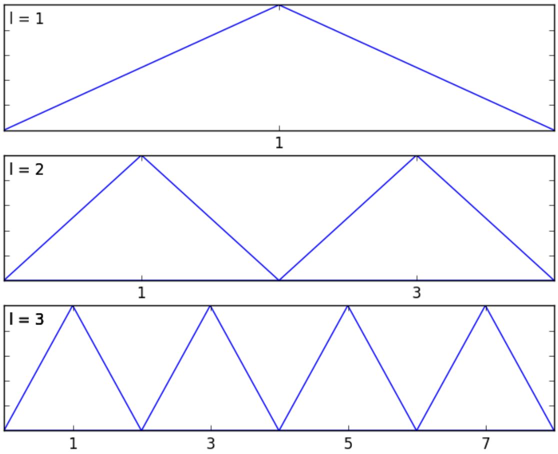
Hirachial basis (vs nodal basis)

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

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

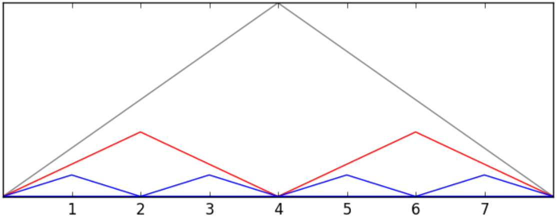
Sparse Grids – Basics

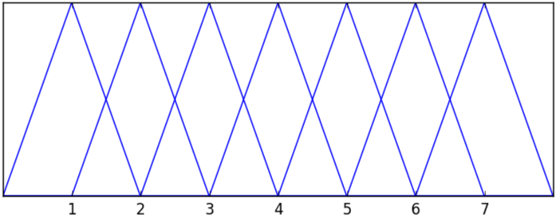
Hirachial Basis



Sparse Grids – Basics

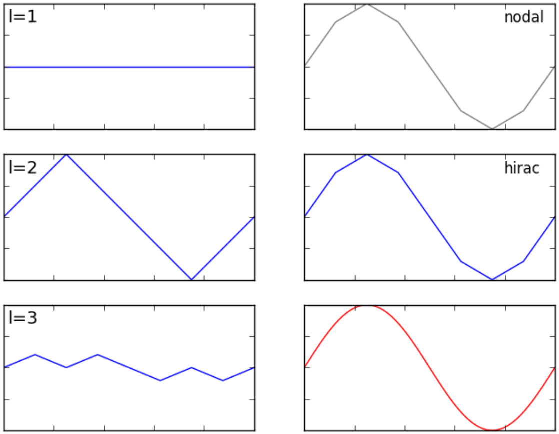
Hirachial vs. nodal basis





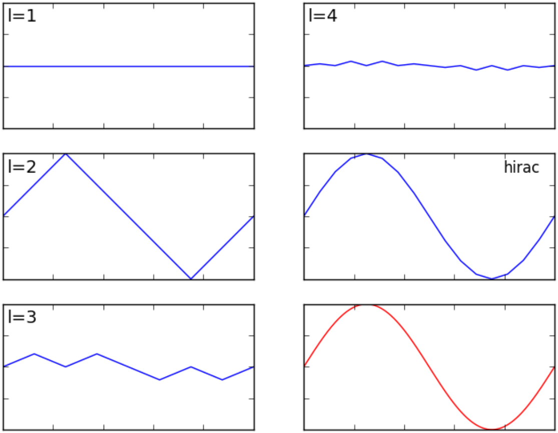
Sparse Grids – Basics

Full grid discretization: Hirachial basis



Sparse Grids – Basics

Full grid discretization: Hirachial basis



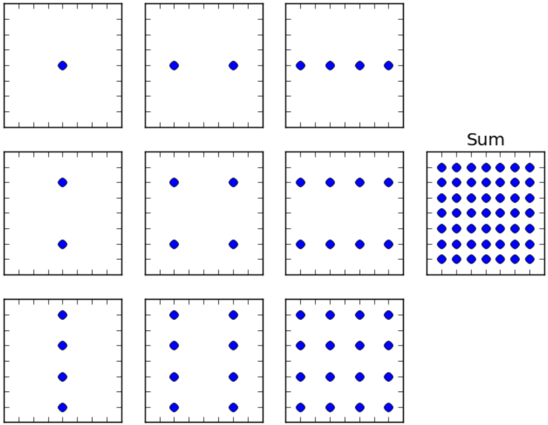
Sparse Grids – Basics

Basis function subspaces

- Combination of levels and dimensions
- Notion of hirachial subspaces
- Defined by the levels of detail in all dimensions $(l_x, l_y, \ldots)$

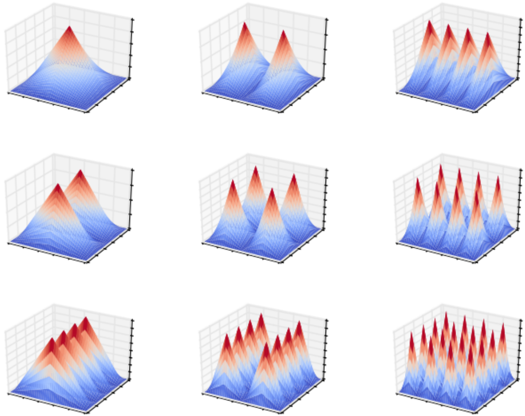
Sparse Grids – Basics

Hirachical gridpoints



Sparse Grids – Basics

Hirachical subspaces



Sparse Grids – Basics

Sparse grid – Changes

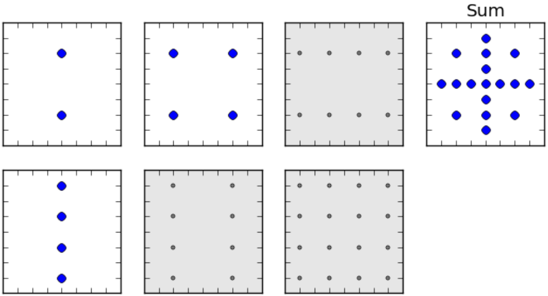
- Throwing away certain subspaces
- Finding those is a *a-priori* solvable optimization problem
- The resulting gird is now **sparse**

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
  $D^2 f$ needs 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*)

Sparse Grids – Data Mining

Machine learning tasks

- Classification
- Regression

Least squares

$$\hat{c} = \arg\min_{f} \left(\frac{1}{N} \sum_{i=1}^{N} (y_i - f(x_i))^2 \; + \; \lambda ||\nabla f|| \right)$$

Sparse Grids – Data Mining

Sparse grid setting

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

Least squares: Sparse gird discretized

$$\hat{c} = \arg\min_{\alpha} \left(\frac{1}{N} \sum_{i=1}^{N} (y_i - \sum_j \alpha_j \phi_j(x_i))^2 \ + \ \lambda \sum_j \alpha_j^2 \right)$$

Sparse Grids – Data Mining

Matrix formulation

$$(\frac{1}{N}BB^T + \lambda C)\alpha = \frac{1}{N}By$$

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

Sparse Grids – Data Mining

Observations

- $BB^T \in \mathbb{R}^{M \times M}$ where $M =$ number of gridpoints
- Scales only linear in $N =$ number of datapoints