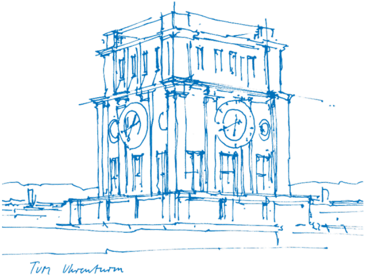
Algorithms for Scientific Computing

Hierarchical Methods
– Interpolation, Approximation and Classification –

Michael Bader
Technical University of Munich

Summer 2022



Part I

Approximation, Classification – Towards Data Mining

Recall Interpolation Problem

Interpolation problem:

- N ansatz functions: $g_k(x)$, $k = 0, \ldots, N-1$
- N supporting points: x_n, $n = 0, \ldots, N-1$
- N interpolation values: f_n, $n = 0, \ldots, N-1$
- find N coefficients c_k such that at all supporting points

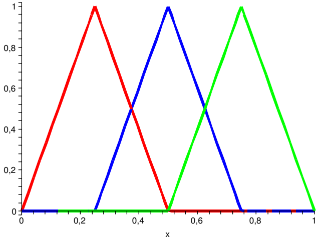
$$f_n = \sum_{k=0}^{N-1} c_k g_k(x_n)$$

How to choose the $g_k(x)$?

- polynomials $\phi_j(x) = x^j \rightsquigarrow$ numerics lecture …
- sine and cosine functions, or $\phi_k(x) = e^{ikx}$

  $\rightsquigarrow$ Discrete Fourier Transform: $f_n = \sum_{k=0}^{N-1} F_k e^{i2\pi nk/N}$
- now: piecewise linear functions

Piecewise Linear Interpolation



$$\varphi_k(x) := \begin{cases} \frac{1}{h_{k-1}}(x - x_{k-1}) & x_{k-1} < x < x_k \\ \frac{1}{h_k}(x_{k+1} - x) & x_k < x < x_{k+1} \\ 0 & \text{otherwise} \end{cases}$$

with $h_k := x_{k+1} - x_k$

Solution of the interpolation problem?

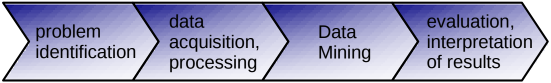
- ansatz function $\varphi_k(x) = 1$ at $x = x_k$
- ansatz function $\varphi_k(x) = 0$ at all $x_n \neq x_k$
- thus: $c_k = f_k$ for all k, because $f_n = \sum_{k=0}^{N-1} c_k \phi_k(x_n) = \cdots = c_n$

Too trivial? $\rightsquigarrow$ consider a slightly more complicated problem ...

Classification in Data Mining

Now for something completely different?

- We consider another application: classification in data mining (our contribution to “Big Data”)
- Aim is to extract new and (hopefully) useful information (out of data bases, etc.)



- We consider *predictive modelling* in data mining:
  - Forecast values on new, previously unseen data
  - Prediction based on given set of data points (*training data*)

Problem Setting: Binary Classification

Classification aims to

- assign a "correct" class label $k \in K$
- to all data points $\vec{x}$ in some d-dimensional feature space $\Omega = \mathbb{R}^d$
- based on set S of pre-classified data points for training

$$S := \{(\vec{x}_i, y_i) \in \Omega \times K\}_{i=1}^{m}$$

- Here: binary classification, for us $K := \{+1, -1\}$

Tasks (examples):

- Did a passenger of the Titanic survive?
  (dimensions: age, male/female, income, ...)?
- Is a bank customer credit-worthy?
  (dimensions: income, type of house, ...)?
- Will personalized advertising pay off for a certain person?
  (dimensions: interests, previous purchases, ...)

Classification

Many approaches exist:

- Decision trees
- Rule-based classifiers (decision rules)
- Instance-based classifiers (k-Nearest Neighbour, ...)
- Probabilistic (Bayes) classifiers
- Based on function representation (artificial neural networks, support vector machines, ...)

Here: Discretization-Based Classification:

- Approach based on discretization of Ω (i.e., approximate S by a function)
- Strength: allows linear training time w.r.t. size of training set (in contrast to many/most others – classification based on comparisons of data point, e.g., implies quadratic growth with size of training set)
- Roadblock: curse of dimensionality $\Rightarrow$ possible solution: sparse grids! (to be discussed ...)

Classification using d-Dimensional Functions

- Given: training set (normalized)

$$S := \left\{ (\vec{x}_i, y_i) \in [0,1]^d \times \{+1, -1\} \right\}_{i=1}^{m}$$

- Assume: training data obtained by random sampling of unknown function f (possibly disturbed by noise)
- Find approximation f_N of f:

$$f(\vec{x}) \approx f_N(\vec{x}) = \sum_{j=1}^{N} v_j \phi_j(\vec{x})$$

$\rightsquigarrow$ following our "coefficients and basis functions" approach

- To determine classification of a new data point $\vec{x}$:
  - Compute $f_N(\vec{x})$
  - Classify as $+1$, if $f_N(\vec{x}) \geq 0$; otherwise -1

First: Classification in 1D

- Given: training set (normalized)

$$S := \{(\vec{x}_i, y_i) \in [0,1] \times \{+1,-1\}\}_{i=1}^{m}$$

- Find approximation f_N of f:

$$f(x) \approx f_N(x) = \sum_{j=1}^{N} v_j \phi_j(x)$$

- Classical approach: minimize quadratic error

$$\sum_{i=1}^{m} \left(f_N(x_i) - y_i\right)^2 \stackrel{!}{=} \min \quad \Leftrightarrow \quad \sum_{i=1}^{m} \left(\sum_{j=1}^{N} v_j \phi_j(x_i) - y_i \right)^2 \stackrel{!}{=} \min$$

- Remember solution via “least squares”: $G^T G v = G^T y$
  where $G_{ij} = \phi_j(x_i)$

Classification in 1D – Least Squares Solution

- minimize quadratic error $\rightsquigarrow$ find values v_j that minimize

$$\sum_{i=1}^{m}\left(\sum_{j=1}^{N} v_j \phi_j(x_i) - y_i\right)^2 \quad \text{or} \quad \sum_{i=1}^{m}\left(\sum_{j=1}^{N} G_{ij} v_j - y_i\right)^2$$

- approach: set all partial derivatives $\frac{\partial}{\partial v_k}$ to zero

$$\frac{\partial}{\partial v_k}\left(\sum_{i=1}^{m}\left(\sum_{j=1}^{N} G_{ij} v_j - y_i\right)^2\right) = \sum_{i=1}^{m} \frac{\partial}{\partial v_k}\left(\sum_{j=1}^{N} G_{ij} v_j - y_i\right)^2 = 0$$

$$\Leftrightarrow \sum_{i=1}^{m} 2\left(\sum_{j=1}^{N} G_{ij} v_j - y_i\right) G_{ik} = 2\sum_{i=1}^{m}\left(\sum_{j=1}^{N} G_{ik} G_{ij} v_j - G_{ik} y_i\right) = 0$$

$$\Leftrightarrow \sum_{i=1}^{m}\sum_{j=1}^{N} G_{ik} G_{ij} v_j - \sum_{i=1}^{m} G_{ik} y_i = \underbrace{\sum_{i=1}^{m} G_{ik} \underbrace{\sum_{j=1}^{N} G_{ij} v_j}_{=(Gv)_i}}_{=(G^T G v)_k} - \underbrace{\sum_{i=1}^{m} G_{ik} y_i}_{=(G^T y)_k} = 0$$

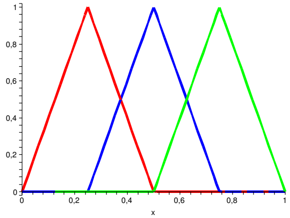
Classification in 1D – Ansatz Functions?

How to choose the $\phi_j(x)$?

- polynomials $\phi_j(x) = x^j \rightsquigarrow$ numerics lecture ...
- sine and cosine functions, or $\phi_j(x) = e^{ijx}$
  $\rightsquigarrow$ Discrete Fourier Transform! (see tutorials)
- new idea: piecewise linear functions

$$\varphi_j(x) := \begin{cases} \frac{1}{h}(x - \xi_{j-1}) & \xi_{j-1} < x < \xi_j \\ \frac{1}{h}(\xi_{j+1} - x) & \xi_j < x < \xi_{j+1} \\ 0 & \text{otherwise} \end{cases}$$

with grid points $\xi_j := j \cdot h$



How difficult is it to compute the solution?

Classification with Piecewise Linear Functions

Structure of the matrix G, where $G_{ij} = \varphi_j(x_i)$:

- assume N_j data points per interval $[\xi_{j-1}, \xi_j]$
- these N_j data points generate N_j non-zeros in column $G_{*,j-1}$ and N_j non-zeros in column $G_{*,j}$
- Example structure (3 basis functions, 4 intervals, 10 data points):

$$G^T := \left(\begin{array}{cc|ccc|cccc|c} * & * & * & * & * & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & * & * & * & * & * & * & * & 0 \\ 0 & 0 & 0 & 0 & 0 & * & * & * & * & * \end{array}\right)$$

- $G^T G$ is thus a tridiagonal matrix
- system $G^T G v = G^T y$ therefore easy to solve

What problems do you expect?

Classification with Piecewise Linear Functions

Possible Problems:

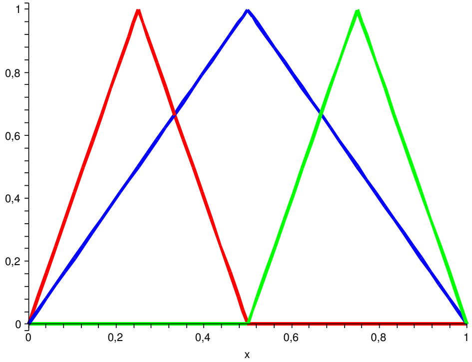
- How to chose resolution h?
  $\rightarrow$ too fine: intervals might be empty (and undetermined)
  $\rightarrow$ too coarse: bad approximation of signal
  $\rightarrow$ too fine: over-approximation of noisy signal
- What if data points are not equally distributed?
  $\rightarrow$ fine resolution required at clustered data points
  $\rightarrow$ coarse resolution required in “empty” regions

Requirements and Possible Approaches:

- Adaptive placement of grid points
  $\rightarrow$ invest grid points where the data points are clustered
- “Regularization” for noisy data
  $\rightarrow$ avoid over-approximation via additional requirements
  $\rightarrow$ require “smooth” approximation and/or limit gradients, e.g.

Hierarchical Basis

- piecewise linear functions with multi-level resolution:



- coarse-level functions (with wide support) $\rightarrow$ never "too fine"
- also "never too coarse"? $\rightarrow$ interaction of fine and coarse?

Define Hierarchical Basis

- consider mesh size $h_n = 2^{-n}$ and grid points $x_{n,i} = i \cdot h_n$
- Define "mother of all hat functions"

$$\phi(x) := \max\{1 - |x|, 0\}$$

- nodal basis then $\Phi_n := \{\phi_{n,i}, 0 \leq i \leq 2^n\}$ with

$$\phi_{n,i}(x) := \phi\left(\frac{x - x_{n,i}}{h_n}\right)$$

- hierarchical basis combines $\widehat{\Phi}_n := \{\phi_{n,i}, i = 1, 3, \ldots, 2^n - 1\}$ (only odd indices) and defines basis as

$$\Psi_n := \bigcup_{l=1}^{n} \widehat{\Phi}_l$$

- hierarchical basis still represents all piecewise linear functions: $\operatorname{span}(\Phi_n) = \operatorname{span}(\Psi_n)$

Classification with Hierarchical Basis Functions

Structure of the matrix G, where $G_{ij} = \varphi_j(x_i)$,
where $\{\varphi_1, \varphi_2, \varphi_3\} = \{\phi_{2,1}, \phi_{1,1}, \phi_{2,3}\}$:

- again assume N_j data points per interval $[\xi_{j-1}, \xi_j]$
- Example structure then (again with 3 basis functions, 4 intervals, 10 data points):

$$G^T := \left(\begin{array}{cc|ccc|cccc|c} * & * & * & * & * & 0 & 0 & 0 & 0 & 0 \\ * & * & * & * & * & * & * & * & * & * \\ 0 & 0 & 0 & 0 & 0 & * & * & * & * & * \end{array} \right)$$

- $G^T G$ no longer tridiagonal $\rightarrow$ expect a denser matrix
- how difficult is it to solve $G^T G v = G^T y$?
  $\rightarrow$ later in the lecture
- efficiency of this approach for data mining?
  $\rightarrow$ requires hierarchical basis in high dimensions

The Curse of Dimensionality

- Recall: d-dimensional training set

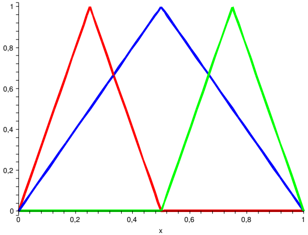
$$S := \left\{(\vec{x}_i, y_i) \in [0,1]^d \times \{+1,-1\}\right\}_{i=1}^{m}$$

- How many grid points necessary for classification?
- Nodal basis in d dimensions:
  n grid points per dimension, thus n^d **grid points**
  $\rightarrow$ **curse of dimensionality**
- How to build hierarchical basis in d dimensions?
- How to build **adaptive** hierarchical approximations?
  Preferably in d dimensions?
- Can we beat the curse of dimensionality using a hierarchical basis approach? $\rightsquigarrow$ **“sparse grids”**

Part II

Exercises

Interpolation with Hierarchical Basis



Solve interpolation problem:

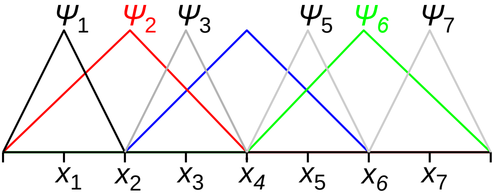
- given f_n and regular grids points x_n
- using **hierarchical basis** $\psi_k(x)$
- requiring: $f_n = \sum_{k=1}^{N} c_k \psi_k(x_n)$

Questions:

1. set up linear system of equations to obtain c_k for $n = 3$
2. derive formulas for the c_k (depending on the f_n)
3. repeat questions 1 and 2 for $n = 7$
4. how does this extend to $n = 2^L - 1$?

Interpolation with Partly-Hierarchical Basis

Consider new (partly hierarchical) basis:



Again, solve interpolation problem:

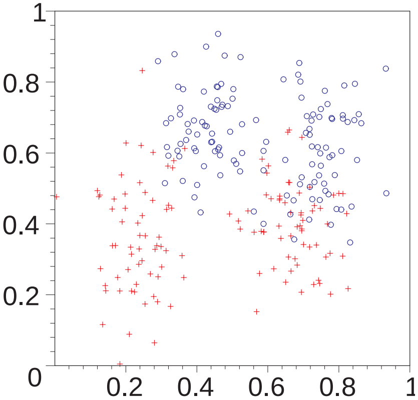
1. set up linear system of equations to obtain c_k for $n = 7$
2. how does this extend to $n = 2^L - 1$?
3. how does this extend to solving the interpolation problem for a hierarchical basis?

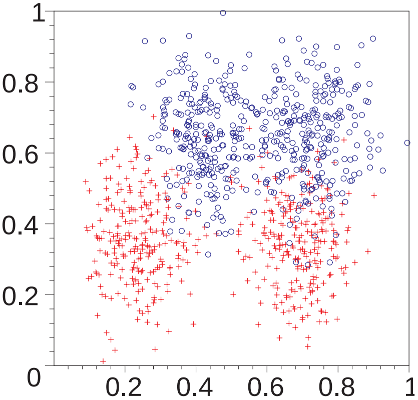
Part III

Sparse Grid Classification (Outlook)

Example 1 – Ripley Data Set

- Artificial, 2*d* data set, frequently used as a benchmark (mixture of Gaussian distributions plus noise)
- 250 points for training, 1000 to test on

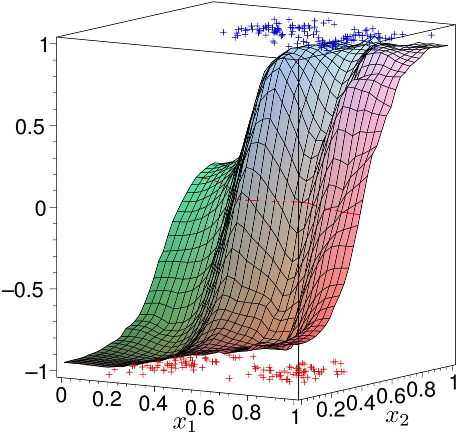


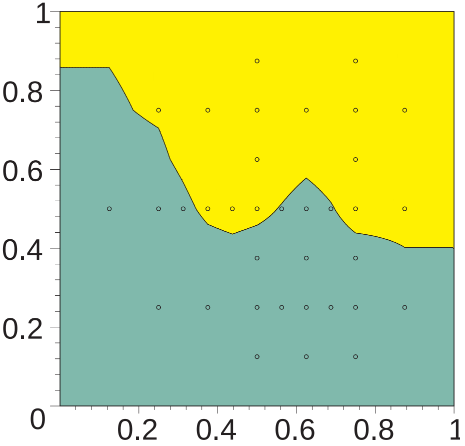


- Constructed to contain 8% of noise

Outlook – Ripley Data Set (Using Sparse Grids)

- Compute adaptive sparse grid classifier, e.g.:



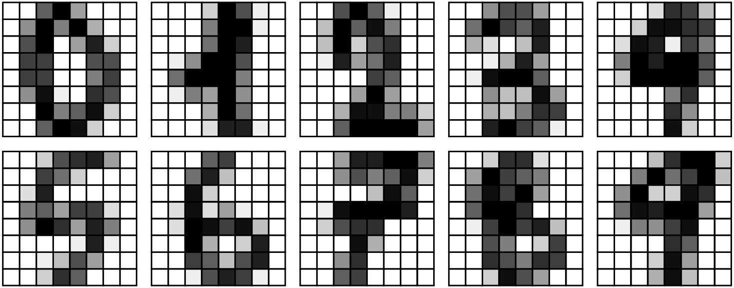


- Best accuracy: 91.5% on test data (max. 92%)
- Suitable treatment of boundary needed

Example 2 – Optidigits

Now for something really high-dimensional. . .

- Optical recognition of handwritten digits:
  Classify images of handwritten digits
- 64-dimensional data set of gray-values (0,1,. . . ,16)



Outlook – Optidigits (Using Sparse Grids)

- Construct ten different binary classifiers (one digit ($+1$) against all others (-1))
- Take the one with highest prediction (function value)

$\Rightarrow$ Best accuracy: 97.7% correctly classified

Summary

- Even high-dimensional problems (“real problems” are often not that high-dimensional) can be successfully solved
- Typically requires to adapt sparse grids to given problem
  - What to do with the boundary (3 points per dim. equiv. to $3^d \rightsquigarrow$ already really large for $d = 64$)?
  - Adaptive refinement!
  - Consider dependency of algorithms in d, N, and m (not only exponential parts can hurt!)
  - ...