Algorithms of Scientific Computing

Hierarchical Methods and Sparse Grids

Michael Bader

Technische Universität München

Summer Term 2012

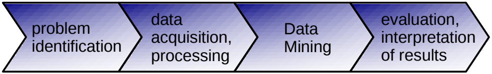
Part VI

More on Sparse Grids: Numerical Classification in Data Mining

Classification in Data Mining

Now for something completely different?

- We consider one more application: classification in data mining
- Aim is extraction of new and (hopefully) useful information out of data bases



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

Binary Classification

Classification problem

- Classification aims to
  - assign a “correct” class label $k \in K$
  - to all data points $\vec{x}$ in some d-dimensional feature space Ω
  - 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:
  - Is person male or female (dimensions: shoe size and body height)?
  - Is customer of bank credit-worthy (dimensions: income, type of house, . . .)?
  - Will direct mailing pay out (dimensions: interests, . . .)?
  - . . .

Classification

Classical approaches

- Decision trees
- Rule-based classifiers (decision rules)
- Instance-based classifiers (k-NN, ...)
- Probabilistic (Bayes) classifiers
- Based on function representation (ANN, SVM, ...)

Problem

- Depend all at least quadratically on size of training set (think of classification based on comparisons of data points)
- Approach based on discretization of Ω would allow linear training time
- But: curse of dimensionality $\Rightarrow$ sparse grids!

Sparse Grid Classification

- 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 randomly sampling of unknown function f disturbed by noise
- Reconstruct piecewise d-linear sparse grid approximation u of f:

$$f_N(\vec{x}) = \sum_{i=1}^{N} v_i \phi_i(\vec{x})$$

- To determine class at new data location $\vec{x}$:
  - Compute $f_N(\vec{x})$
  - Predict class $+1$, if $f_N(\vec{x}) \geq 0$; otherwise -1

Sparse Grid Classification

- Solve regularized least squares problem

$$f_N \stackrel{!}{=} \underset{f_N \in V_N}{\arg\min} \left(\frac{1}{m} \sum_{i=1}^{m} (y_i - f_N(\vec{x}_i))^2 + \lambda ||\nabla f_N||_{L_2}^2 \right)$$

- Aims:
  - Be close to training data: minimize quadratical error
  - Prevent overfitting: minimize gradient to avoid oszillations due to noise in training data
  - Parameter λ to stir trade-off
- Derive system of linear equations:
  - We plug-in f_N
  - And minimize by setting each first derivative $\partial / \partial v_i$ to zero

From Minimization to System of Linear Equations

⇒ N linear equations for N unknowns

$$\left(\frac{1}{m}BB^T + \lambda C\right)\vec{v} = \frac{1}{m}B\vec{y}\,,$$

- With matrices C and B

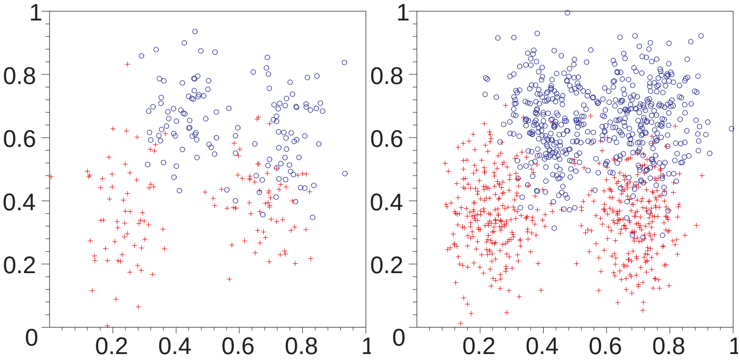
$$(C)_{ij} = \langle \nabla\phi_i(\vec{x}), \nabla\phi_j(\vec{x})\rangle_{L_2}, \qquad (B)_{ij} = \phi_i(\vec{x}_j)$$

- B here is simple, can be applied to vector in $\mathcal{O}(Nm)$ (and even better)
- C is a well-known matrix in the field of solving partial differential equations (Poisson problem, e.g.): $\mathcal{O}(N)$

⇒ Solve linear system iteratively to determine hierarchical surplusses v_i and thus classifier f_N

Example 1 – Ripley Data Set

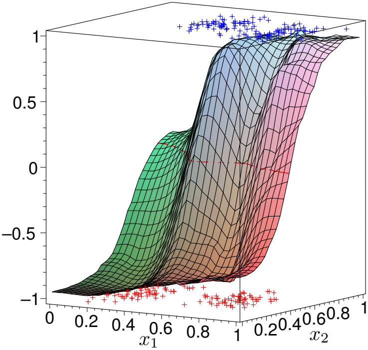
- Artificial, 2*d* data set
- 250 points for training, 1000 to test on

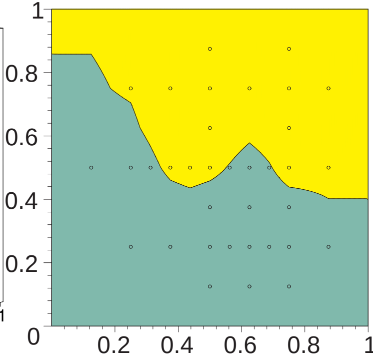


- Constructed to contain 8% of noise

Example 1 – Ripley Data Set (2)

- Compute adaptive sparse grid classifier
- Result can look as follows



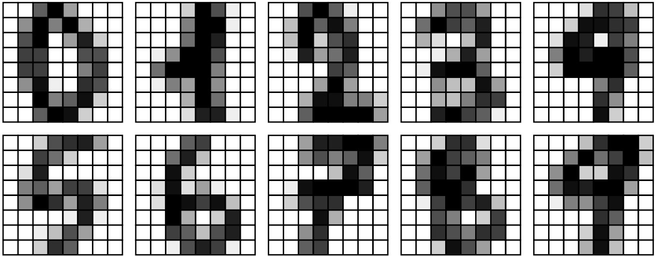


- Best accuracy: 91.5% on test data (max. 92%)
- Suitable treatment of boundary needed

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)



Example 2 – Optidigits (2)

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

$\Rightarrow$ Best accuracy: 97.7% correctly classified

Summary

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