Greedy Kernel Techniques with Applications to Machine Learning

Robert Schaback

Jochen Werner

Göttingen University

Institut für Numerische und Angewandte Mathematik

http://www.num.math.uni-goettingen.de/schaback

Overview

1. Historical Remarks: From RBF to Kernels
2. Kernel Techniques
3. Simplification by Relaxation
4. Online Learning and Greedy Methods

Development of Kernel Techniques

1. Radial Basis Functions (with a little help from Will)
2. Computer–Aided Design
3. Meshless Methods for PDE Solving in Engineering
4. Learning with Kernels

Reconstruction of Functions

Problem: Find $u\ :\ \Omega \to I\!R$

Given: Data

Discrete scattered data	$(x_j, u(x_j)),\ \ 1 \le j \le N$
PDE data	$\left\{ \begin{array}{ll} (x, \Delta u(x)) & x \in \Omega \\ (y, u(y)) & y \in \partial\Omega \end{array} \right\}$
General functionals	$(\lambda, \lambda(u)),\ \ \lambda \in \Lambda =$ set of functionals

Learning

Problem: Find $u\ :\ \Omega \ni$ Stimulus $\mapsto$ Response

Given: Training data

$$(x_j, u(x_j)),\ \ 1 \le j \le N$$

Why Kernels?

Data $(x_j, u(x_j)) \in \Omega \times I\!R,\ 1 \le j \le N$

General linear reconstruction:

$$\tilde{u}(x) := \sum_{j=1}^{n} \underbrace{L_j(x)}_{=?} u(x_j)$$

Error estimate

$$\begin{aligned}
|u(x) - \tilde{u}(x)|^2 &= \left| u(x) - \sum_{j=1}^{n} L_j(x) u(x_j) \right|^2 \\
&= \left| \left(\delta_x - \sum_{j=1}^{n} L_j(x) \delta_{x_j} \right) (u) \right|^2 \\
&\le \underbrace{\left\| \delta_x - \sum_{j=1}^{n} L_j(x) \delta_{x_j} \right\|_{H^*}^2}_{\text{minimize wrt. } L_j(x)} \|u\|_H^2 \\
\underbrace{(\delta_x, \delta_{x_k})_{H^*}}_{=:K(x,x_k)} &= \sum_{j=1}^{n} L_j^*(x) \underbrace{(\delta_{x_j}, \delta_{x_k})_{H^*}}_{=:K(x_j,x_k)},\ 1 \le k \le N
\end{aligned}$$

Optimal L_j^* is Lagrange interpolant
on span $\{K(x, x_k)\ :\ 1 \le k \le N\}$

Necessary: $u \in$ RKHS H

Generalization: Arbitrary data functionals $\lambda \in H^*$
(meshless collocation methods)

Generalized Data

Data $(\lambda_j, \lambda_j(u)) \in H^* \times I\!R,\ 1 \leq j \leq N$

General linear reconstruction:

$$\tilde{u} := \sum_{j=1}^{n} \underbrace{L_j}_{=?} \lambda_j u$$

Error estimate with test functional $\mu \in H^*$

$$\begin{aligned}
|\mu(u) - \mu(\tilde{u})|^2 &= \left| \mu(u) - \sum_{j=1}^{n} \mu(L_j)\lambda_j(u) \right|^2 \\
&= \left| \left(\mu - \sum_{j=1}^{n} \mu(L_j)\lambda_j \right)(u) \right|^2 \\
&\leq \underbrace{\left\| \mu - \sum_{j=1}^{n} \mu(L_j)\lambda_j \right\|_{H^*}^2}_{\text{minimize wrt. } \mu(L_j)} \|u\|_H^2 \\
\underbrace{(\mu, \lambda_k)_{H^*}}_{=:K(\mu,\lambda_k)} &= \sum_{j=1}^{n} \mu(L_j^*) \underbrace{(\lambda_j, \lambda_k)_{H^*}}_{=:K(\lambda_j,\lambda_k)},\ 1 \leq k \leq N
\end{aligned}$$

Optimal L_j^* is Lagrange interpolant
on span $\{\text{Riesz representer of } \lambda_k\ :\ 1 \leq k \leq N\}$

Necessary: $u \in$ RKHS H, $\lambda_j \in H^*$

Second Optimality Property

Data $(x_j, u(x_j)) \in \Omega \times I\!R,\ 1 \leq j \leq N$

Optimal linear reconstruction:

$$u^*(x) := \sum_{j=1}^{n} L_j^*(x) u(x_j)$$

Minimization of norm under all other interpolants:

$$\|u^*\|_H = \min \left\{ \|v\|_H \ : \ \begin{array}{l} v \in H \\ v(x_j) = u(x_j),\ 1 \leq j \leq N \end{array} \right\}$$

Learning with Kernels

Training data $(x_j, u(x_j)) \in \Omega \times I\!R,\ 1 \le j \le N$

Kernel Trick:

$$\begin{array}{rcl} \Phi & & \Omega \to H = \textbf{feature space} \\ \Phi(x) & := & K(x,\cdot) \in H \\ K(x,y) & = & (K(x,\cdot), K(y,\cdot))_H \\ & = & (\Phi(x), \Phi(y))_H \\ u^*(x) & = & \sum_{j=1}^{N} L_j^* u(x_j) \\ & = & \sum_{j=1}^{N} \alpha_j^* K(x, x_j) \end{array}$$

Optimal L_j^* is Lagrange interpolant
on span $\{K(x, x_k)\ :\ 1 \le k \le N\}$

Minimization of norm
under all other interpolants:

$$\|u^*\|_H = \min \left\{ \|v\|_H\ :\ \begin{array}{l} v \in H \\ v(x_j) = u(x_j),\ 1 \le j \le N \end{array} \right\}$$

For Classification:

Minimization of $\|v\|_H$ is maximization
of separation margin

Simplification by Relaxation

Unrelaxed:

$$\begin{aligned} \|u^*\|_H &= \min \left\{ \|v\|_H \;:\; \begin{array}{l} v \in H \\ v(x_j) = u(x_j),\ 1 \le j \le N \end{array} \right\} \\ u^*(x) &= \sum_{j=1}^{N} L_j^* u(x_j) \\ &= \sum_{j=1}^{N} \alpha_j^* K(x, x_j) \qquad \textbf{full}\text{ sum!} \end{aligned}$$

Relaxed: Given $\epsilon \ge 0$

$$\begin{aligned} \|u^*\|_H &= \min \left\{ \|v\|_H \;:\; \begin{array}{l} v \in H \\ |v(x_j) - u(x_j)| \le \epsilon,\ 1 \le j \le N \end{array} \right\} \\ u^*(x) &= \sum_{\substack{j=1 \\ |v(x_j)-u(x_j)|=\epsilon}}^{N} \alpha_j^* K(x, x_j) \qquad \textbf{reduced}\text{ sum!} \end{aligned}$$

Reason: Kuhn–Tucker conditions for linear constraints

Support Vectors: $x_j^{\pm}$ with $v(x_j^{\pm}) - u(x_j^{\pm}) = \pm\epsilon$

Quadratic problem: Minimize $\|v\|_H^2 = \sum_{j,k=1}^{N} \alpha_j \alpha_k K(x_k, x_j)$

Open problem: Bound on # of support vectors

Online Learning

Given $\epsilon \geq 0$. Current "knowledge":

$$u^*(x) = \sum_{\substack{j=1 \\ |v(x_j)-u(x_j)|=\epsilon}}^{\mathbf{n}} \alpha_j^* K(x, x_j) \text{ **reduced** sum!}$$

Iteration of Online Learning:

1. Wait for new training sample $(x, u(x))$
2. If $|u(x) - u^*(x)| \leq \epsilon$ do nothing.
3. If $|u(x) - u^*(x)| > \epsilon$ set $x_{n+1} := x$ and update u^* to u^{**} with (**Learning by new blunders**)

$$|u(x_j) - u^{**}(x_j)| \leq \epsilon, \quad 1 \leq j \leq n+1.$$

Theorem $\|u^*\|_H < \|u^{**}\|_H$ (knowledge gain)

The increase can be quantified and bounded below.

Theorem If Ω is compact, and if the presented samples satisfy

$$h_k := \sup_{y \in \Omega} \min_{1 \leq j \leq k} \|y - x_j\|_2 \to 0 \text{ for } k \to \infty$$

the algorithm performs only a finite number of steps.

Snape Teaching

Theorem If Ω is compact, and if the teacher always poses the hardest possible problem, i.e.

$$|u(x_{n+1}) - u^*(x_{n+1})| = \|u - u^*\|_{\infty,\Omega}$$

then the algorithm performs only a finite number of steps.

Drawback: Number of memorized samples grows

Goal: Forgetting well–learned samples

Forgetful Students

Given $\epsilon > \delta \geq 0$. Current “knowledge”:

$$u^*(x) = \sum_{\substack{j=1 \\ |v(x_j)-u(x_j)|=\delta}}^{\mathbf{n}} \alpha_j^* K(x, x_j) \text{ \textbf{reduced} sum!}$$

Iteration of Forgetful Online Learning:

1. Wait for new training sample $(x, u(x))$
2. If $|u(x) - u^*(x)| \leq \epsilon$ do nothing.
3. If $|u(x) - u^*(x)| > \epsilon$ set $x_{n+1} := x$ and update u^* to u^{**} with (**Learning by new blunders**)

$$|u(x_j) - u^{**}(x_j)| \leq \delta, \quad 1 \leq j \leq n+1.$$

4. Discard old samples with $< \delta$ above.
   (**Forget well–managed samples**)

Theorem Similar results as in the “memorizing” case.

Implementation

Given $\epsilon \geq 0$, training data $(x_j, u(x_j)),\ 1 \leq j \leq n$.

Quadratic Optimization Problem

$$\text{Minimize} \quad \sum_{j,k=1}^{n} \alpha_j \alpha_k K(x_j, x_k)$$

$$-\epsilon \leq \sum_{j=1}^{n} \alpha_j K(x_j, x_k) - u(x_k) \leq \epsilon, \quad 1 \leq k \leq n$$

Required:
Fast update method based on active sets

Problem:
Quadratic objective function

Current workaround:
Linear systems for interpolation,
Greedy Techniques

Example: Previous algorithm with $\delta = 0$.

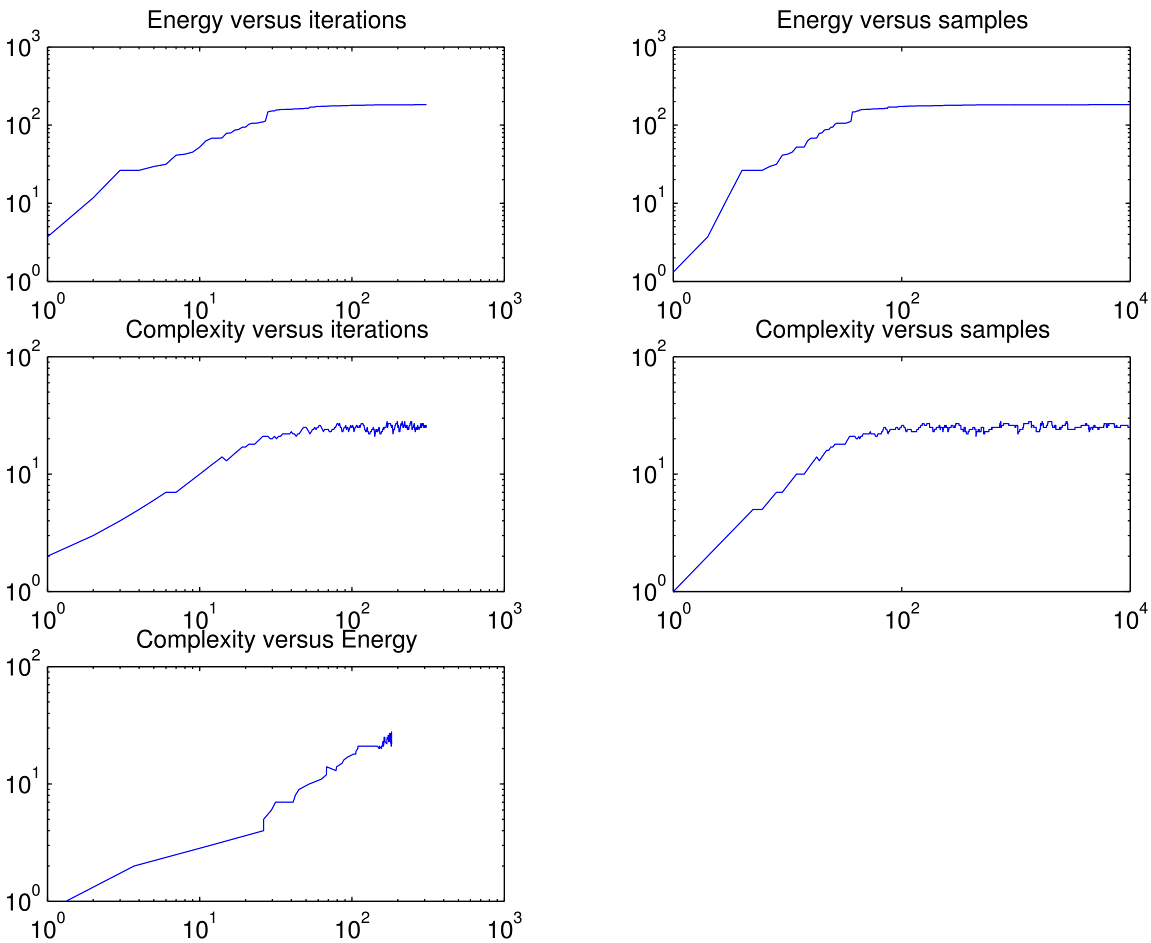
Learning the "peaks" Function

Online learning, random samples, $\epsilon = 0.01$

After some 100 samples: starts to discard

After some 400 learning steps: nothing to learn

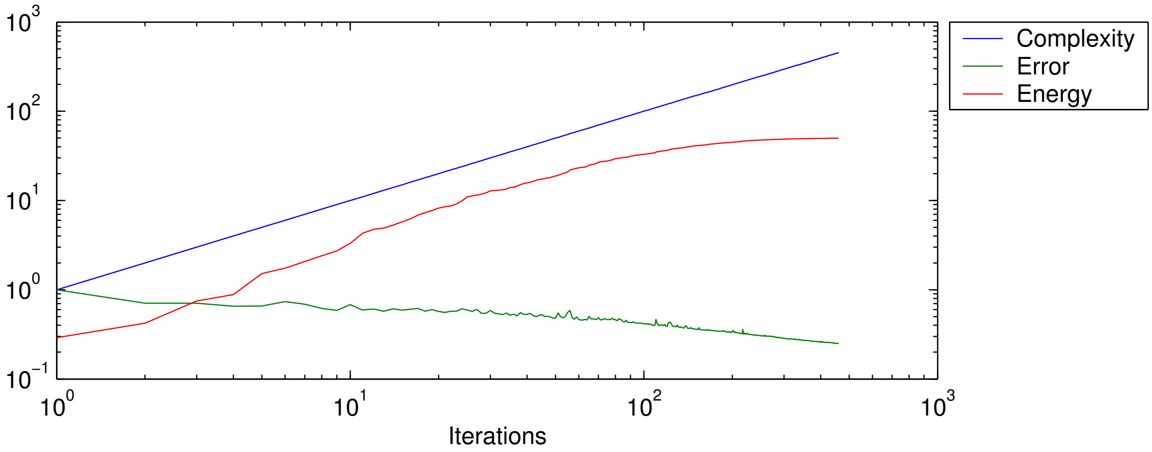
Final complexity: about 35 support vectors

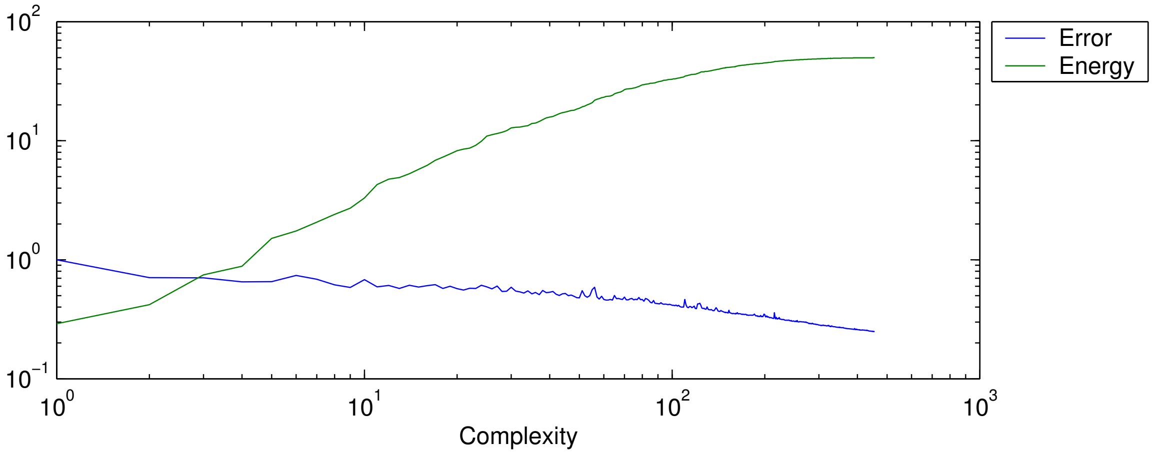


in

Boston Census Data

22784 training samples with 16 variables describing input data for estimating the price of houses from the 1990 US census. Method: Greedy with Snape teaching





in

Open Problems

Find non–quadratic forgetful method

Find fast quadratic update method

Prove upper bounds for number of support vectors

Understand bias–variance tradeoff

Use learning techniques for solving PDEs