Sum and Product Kernel Regularization Networks

Kudová, Petra
2005

Institute of Computer Science
Academy of Sciences of the Czech Republic

Sum and Product Kernel Regularization Networks

Kudová Petra, Šámalová Terezie

Technical report No. 935

May 2005

Pod Vodárenskou věží 2, 182 07 Prague 8, phone: (+4202) 6605 1111, fax: (+4202) 86 585 789,
e-mail:ics@cs.cas.cz

Institute of Computer Science
Academy of Sciences of the Czech Republic

Sum and Product Kernel Regularization Networks[1]

Kudová Petra, Šámalová Terezie[2]

Technical report No. 935

May 2005

Abstract:

We study approximation problems formulated as regularized minimization problems with kernel-based stabilizers. These approximation schemas exhibit easy derivation of solution to the problem, in the shape of linear combination of kernel functions (one-hidden layer feed-forward neural network schemas). We exploit the article by N. Aronszajn [1] on reproducing kernels and use his formulation of sum of kernels and product of kernels, and resulting kernel space to derive approximation schemas – Sum-Kernel Regularization Network and Product-Kernel Regularization Network. We present some concrete applications of the derived schemas, demonstrate their performance on experiments and compare them to classical solutions.

Keywords:
Neural networks, minimization of functionals, regularization theory, product of kernels, sum of kernels, experiments

[1]T. Šámalová was partially supported by the Program "Information Society" under project 1ET100300517, P. Kudová was partially supported by the HPC-EUROPA project (RII3-CT-2003-506079), with the support of the European Community - Research Infrastructure Action under the FP6 "Structuring the European Research Area" Programme, and by the Program "Information Socitey" under project 1ET100300414.

[2]Institute of Computer Science, Academy of Sciences of CR, Pod vodárenskou věží 2, P.0. Box 5, 182 07 Prague 8, Czech Republic
Charles University, Faculty of Mathematics and Physics, Ke Karlovu 3, 121 16 Prague 2, Czech Republic,
{kudova, terka}@cs.cas.cz

Section one is a brief overview of basic notations used, section two brings theory of reproducing kernel Hilbert spaces (RKHS). Further two sections show theory of sum and product of kernels. In section five learning from data and connection to RKHS theory is shown. In section six concrete minimization schemas including employment of sum and product kernels is shown. Section seven brings learning algorithm based on the theory and section eight experiments and comparisons to standard techniques. The part concerning product of kernels has already been presented in [5].

1 Preliminaries

A *normed linear space* W is any vector space over $\mathbb{R}$ or $\mathbb{C}$ with a *norm* $\|.\|$, where for all $x, y \in W$, $\lambda \in \mathbb{R}$ (or $\mathbb{C}$).

1. $\|x\| \geq 0$ and $\|x\| = 0$ only if $x = 0$
2. $\|\lambda x\| = |\lambda|\|x\|$, and
3. $\|x + y\| \leq \|x\| + \|y\|$.

Banach space $(B, \|.\|)$ is any normed linear space that is complete in its norm. A *Hilbert space* is a Banach space in which the norm is given by an inner product $\langle ., . \rangle$, that is $\|x\| = \langle x, x \rangle^{1/2}$.

Let d, k be positive integers, $\Omega \subseteq \mathbb{R}^d$. We $(C(\Omega), \|.\|_C)$ denote the space of continuous functions on Ω with the maximum norm. Next, C_k will denote all functions with continuous Fréchet derivative up to order k and C_∞ all infinitely differentiable functions. We say that $f \in C_\infty$ belongs to the *Schwartz space* $\mathcal{S}(\mathbb{R}^n)$ if $p \cdot D^\alpha f$ is a bounded function for any multiindex $\alpha = (\alpha_1, \ldots, \alpha_n)$ and any polynomial $p = \sum_i c_{\beta_i} x_1^{\beta_{i_1}} \ldots x_n^{\beta_{i_n}}$ on $\mathbb{R}^n$ (where $D^\alpha(f) = \left(\frac{\partial}{\partial x_1}\right)^{\alpha_1} \ldots \left(\frac{\partial}{\partial x_n}\right)^{\alpha_n}$). For convenience let us define (following [11]) the *normalized Lebesgue measure* m_d on $\mathbb{R}^d$ as $\mathrm{d}m_d(x) = (2\pi)^{-d/2}\mathrm{d}x$.

The Lebesgue space $(\mathcal{L}_p(\Omega), \|.\|_p)$ of functions on Ω with integrable p-th power will be renormed: $\|f\|_p = \left\{\int_\Omega |f|^p \mathrm{d}m_d\right\}^{1/p}$. This will simplify the use of *Fourier transform* $\hat{f}$ of the function $f \in \mathcal{L}^1(\mathbb{R}^d)$: $\hat{f}(t) = \int_{\mathbb{R}^d} f(x) e^{-it\cdot x} \mathrm{d}m_d$, where $t \in \mathbb{R}^d$ and $t \cdot x = t_1 x_1 + \cdots + t_d x_d$.

Let B be a Banach space, $\Omega \subset B$ and let $f : \Omega \times \Omega \to \mathbb{R}$ be a symmetric function (that is $f(x, y) = f(y, x)$). Then f is *positive definite* if for any $a_1, \ldots, a_n \in \mathbb{C}$ and $t_1, \ldots, t_n \in \Omega$

$$\sum_{i,j=1}^{n} \overline{a}_i a_j f(t_i, t_j) \geq 0,$$

where $\overline{a}$ is complex adjoint of a. We call the function *strictly positive definite* if it is positive definite and $\sum_{i,j=1}^n \overline{a}_i a_j f(t_i, t_j) = 0$, implies $a_i = a_j = 0 \ \forall i, j \in \{1, \ldots, n\}$.

Let V and W be vector spaces over the same field. Then $\mathcal{L} : W \to V$ is a *linear mapping* if and only if $\mathcal{L}(\lambda x + \mu y) = \lambda \mathcal{L}x + \mu \mathcal{L}y$ for all $x, y \in W$ and $\lambda, \mu \in F$ (where $F = \mathbb{R}$ or $\mathbb{C}$). If $V = W$, we call $\mathcal{L}$ an *operator*, if $V = F$ we call it a *linear form* or a *functional* on W.

For a functional $\mathcal{F} : X \to (-\infty, +\infty]$ we write $\operatorname{dom} \mathcal{F} = \{f \in X : \mathcal{F}(f) < +\infty\}$ and call this set the *domain* of $\mathcal{F}$. *Continuity* of $\mathcal{F}$ in $f \in \operatorname{dom} \mathcal{F}$ is defined as usual. A functional is *sequentially lower semicontinuous* if and only if the convergence of $\{f_n\}$ to f implies $\mathcal{F}(f) \leq \liminf_{n\to\infty} \mathcal{F}(f_n)$. Functional $\mathcal{F}$ is *weakly sequentially lower semicontinuous* if and only if $f_n \rightharpoonup f$ implies $\mathcal{F}(f) \leq \liminf_{n\to\infty} \mathcal{F}(f_n)$.

A functional $\mathcal{F}$ is *convex* on a convex set $E \subseteq \operatorname{dom} \mathcal{F}$ if for all $f, g \in E$ and all $\lambda \in [0, 1]$, $\mathcal{F}(\lambda f + (1-\lambda)g) \leq \lambda \mathcal{F}(f) + (1-\lambda)\mathcal{F}(g)$. Functional $\mathcal{F}$ is *(strongly) quasi-convex* if for all $f, g \in E, f \neq g$ it holds: $\mathcal{F}\left(\frac{1}{2}f + \frac{1}{2}g\right) \ (<) \leq \max\{\mathcal{F}(f), \mathcal{F}(g)\}$.

2 Reproducing Kernel Hilbert Spaces

Reproducing Kernel Hilbert Space (shortly RKHS) was defined by Aronszajn, 1950 ([1]) as Hilbert space $\mathcal{H}$ of functions (real or complex) defined over $\Omega \subset \mathbb{R}^d$ with the property, that for each $x \in \Omega$ the

evaluation functional on $\mathcal{H}$ given by $\mathcal{F}_x : f \mapsto f(x)$ is bounded. This implies existence of a positive definite symmetric function $k : \Omega \times \Omega \to \mathbb{R}$ (so called *reproducing kernel*) corresponding to $\mathcal{H}$ such that

1. for any $f \in \mathcal{H}$ and $y \in \Omega$ the following reproducing property holds $f(y) = \langle f(x), k(x,y) \rangle$, where $\langle ., . \rangle$ is scalar product in $\mathcal{H}$ and
2. for every $y \in \Omega$, the function $k_y(x) = k(x,y)$ is an element of $\mathcal{H}$.

Note that the reproducing kernel for $\mathcal{H}$ is unique. On the other hand, every positive definite symmetric function is a reproducing kernel for exactly one Hilbert space, that can be described as $\text{comp}\{\sum_{i=1}^n a_i k_{x_i}; x_i \in \Omega, a_i \in \mathbb{R}\}$, where comp means completion of the set. $\| \sum_{i=1}^n a_i k_{x_i} \|^2 = \sum_{i=1}^n \sum_{j=1}^n k(x_i, x_j) \overline{a_i} a_j$.

2.1 Proofs

All the proofs presented here have been sketched in [1].

Lemma 2.1 *Let $\mathcal{L}(\Omega)$ be a real valued RKHS with k as kernel. Then $\mathcal{L}_C := \{f_1 + if_2; f_1, f_2 \in \mathcal{L}\}$ with $\|f_1 + if_2\|^2 = \|f_1\|^2 + \|f_x\|^2$ is a complex RKHS with the same k as kernel.*

Proof: $\mathcal{L}_C$ is clearly a Hilbert space. Evaluation functionals remain linear and bounded, i.e. $\mathcal{L}_C$ is RKHS. And for any $f \in \mathcal{L}$ it holds: $if(y) = \langle if(x), k(x,y) \rangle$. □

We see that it is sufficient to consider only complex RKHS.

Lemma 2.2 *Let $\mathcal{L}(\Omega)$ be a Hilbert space with a reproducing kernel k. Then k is unique.*

Proof: Suppose we have two reproducing kernels k, k' and $k \neq k'$. Then for some x, y we have $0 < \|k(x,y) - k'(x,y)\|^2 = \langle k - k'(x,y), k - k'(x,y) \rangle = \langle k - k'(x,y), k(x,y) \rangle - \langle k - k'(x,y), k'(x,y) \rangle = (k - k')(y,y) - (k - k')(y,y) = 0$, which is a contradiction. □

Lemma 2.3 *Let $\mathcal{L}(\Omega)$ be a Hilbert space with the property that all evaluation functionals $\mathcal{F}_x$ are linear and bounded. Then there exists a reproducing kernel satisfying properties (i) and (ii) that is also positive definite. On the other hand from (i) and (ii) we obtain linear bounded (continuous) evaluation functionals.*

Proof: $\mathcal{F}_y$ is a linear bounded (i.e. continuous) functional on Hilbert space $\mathcal{L}(\Omega)$. Thus by Fréchet-Riesz Theorem [7, p. 19] we have $a_y \in \mathcal{L}$ such that $\mathcal{F}_y(f) = \langle f(x), a_y(x) \rangle$. We put $a_y(x) = k(x,y)$ obtaining the reproducing kernel.

To check the desired properties (symmetry and positive definiteness) we use the reproducing property: $\sum_{i,j=1}^n \overline{a}_i a_j k(x_i, x_j) = \langle \sum_{j=1}^n a_j k(y, x_j), \sum_{i=1}^n a_i k(y, x_i) \rangle = \| \sum_{j=1}^n a_j k(y, x_j) \|^2 \geq 0$ and $k(x,y) = \langle k(z,y), k(z,x) \rangle = \overline{\langle k(z,x), k(z,y) \rangle} = \overline{k(y,x)}$.

To prove the last statement it is sufficient to observe, that: $|f(y)| = |\langle f(x), k(x,y) \rangle| \leq \|f\| \langle k(x,y), k(x,y) \rangle^{1/2} = \|f\| k(y,y)^{1/2}$. □

Lemma 2.4 *To every $k(x,y)$ satisfying the properties (i) and (ii) there corresponds one and only one Hilbert space $\mathcal{H}$ admitting k as a reproducing kernel.*

Proof: Let us take the class of all functions of the form $\sum \alpha_s k(x, y_s)$ with the norm $\| \sum_{s=1}^n \alpha_s k(x, y_s) \|^2 = \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j k(x_i, x_j)$. To complete the space we add limits of all Cauchy sequences (relative to the above norm which gives point-wise convergence). □

Theorem 2.5 *Let F be a linear class of functions with scalar product defined on Ω satisfying all the properties of a Hilbert space with the exception of completeness (an incomplete Hilbert space). The class can be completed if and only if*

1. *for every fixed $y \in \Omega$ the linear functional $\mathcal{F}_y(f)$ is bounded in F*
2. *for a Cauchy sequence $\{f_m\} \subset F$ the condition $f_m(y) \to 0$ for every y implies $\|f_m\| \to 0$.*

If the completion is possible, it is unique.

Proof: See [1, p. 347]. □

3 Sum of reproducing kernels

Sum of reproducing kernels was proposed and basic properties proved in [1]. We will consider sum of Reproducing Kernel Hilbert Spaces. For $i = 1, 2$ let F_i be an RKHS of functions on Ω, let K_i be the corresponding kernels and $\|.\|_i$ the corresponding norms. Consider the following space of all couples $\{f_1, f_2\}$ on Ω. $H = \{\{f_1, f_2\} \mid f_1 \in F_1, f_2 \in F_2\}$. The metric will be given by $\|\{f_1, f_2\}\|^2 = \|f_1\|_1^2 + \|f_2\|_2^2$.

Now we have to deal with duplicacies. Consider the class F_0 of all functions f belonging to $F_1 \cap F_2$. We define $H_0 := \{\{f, -f\}; f \in F_0\}$. It is a closed subspace of H and thus we can write $H = H_0 \oplus H'$, where H' is complementary subspace to H_0. Now to every element $\{f', f''\}$ of H there corresponds a function $f(x) = f'(x) + f''(x)$. This is a linear correspondence transforming H into a linear class of functions F. Elements of H_0 are transformed into zero functions and thus the correspondence between H' and F is one-to-one and has an inverse (for every $f \in F$ we obtain one $\{g'(f), g''(f)\}$). We define metric on F by

$$\|f\|^2 = \|\{g'(f), g''(f)\}\|^2 = \|g'(f)\|_1^2 + \|g''(f)\|_2^2.$$

Now we will show that to the class F with the above defined norm there corresponds a reproducing kernel $K = K_1 + K_2$.

Theorem 3.1 *([1]) Let F_i be RKHS and K_i and $\|.\|_i$ the corresponding kernels and norms. Let F be defined as above with norm $\|f\|^2 = \|\{g'(f), g''(f)\}\|^2 = \|g'(f)\|_1^2 + \|g''(f)\|_2^2$. Then*

$$K(x, y) = K_1(x, y) + K_2(x, y)$$

is kernel corresponding to F.

The claim holds also for F defined as class of all functions $f = f_1 + f_2$ with $f_i \in F_i$ and norm $\|f\|^2 = min\|f_1\|_1^2 + \|f_2\|_2^2$ minimum taken for all decompositions $f = f_1 + f_2$ with f_i in F_i.

3.1 Proofs

Lemma 3.2 *Let F_1, F_2 be RKHS. Define a space $F' := \{\{f_1, f_2\} \mid f_1 \in F_1, f_2 \in F_2\}$ and norm on it given by $\|\{f_1, f_2\}\|^2 = \|f_1\|_1^2 + \|f_2\|_2^2$. Then this norm is defined by scalar product.*

Proof: It is enough to prove that rectangular law is valid, i.e. we want to have $\|x+y\|^2 + \|x-y\|^2 = 2(\|x\|^2 + \|y\|^2)$ for all $x, y \in F'$. Let $x = \{f_1, f_2\}$, $y = \{g_1, g_2\}$, $f_1, g_1 \in F_1$, $f_2, g_2 \in F_2$. Now we just rewrite the rectangular law: $\|\{f_1, f_2\} + \{g_1, g_2\}\|^2 + \|\{f_1, f_2\} - \{g_1, g_2\}\|^2 =$
$\|\{f_1 + g_1, f_2 + g_2\}\|^2 + \|\{f_1 - g_1, f_2 - g_2\}\|^2$. Since F_1 and F_2 are Hilbert spaces and thus rectangular law is valid, we can continue using definition of the norm: $\|\{f_1+g_1, f_2+g_2\}\|^2+\|\{f_1-g_1, f_2-g_2\}\|^2 =$
$\|f_1+g_1\|_1^2+\|f_2+g_2\|_2^2+\|f_1-g_1\|_1^2+\|f_2-g_2\|_2^2 = 2(\|f_1\|_1^2+\|g_1\|_1^2)+2(\|f_2\|_2^2+\|g_2\|_2^2) = 2(|\{f_1, f_2\}\|^2+$
$\|\{g_1, g_2\}\|^2)$. □

Proof of theorem 3.1: First we see that $K(x, y) = K_1(x, y) + K_2(x, y)$ as a function of x for fixed y belongs to F, since it corresponds to element $\{K_1(x, y), K_2(x, y)\} \in H$.

Denote for fixed y $K'(x, y) = g'(K(x, y))$ and $K''(x, y) = g''(K(x, y))$. Thus we have $K'(x, y) + K''(x, y) = K(x, y) = K_1(x, y)+K_2(x, y)$. Hence we have $K'(x, y)-K_1(x, y) = -(K''(x, y)-K_2(x, y))$. So we see that $\{K'(x, y) - K_1(x, y)$,
$K''(x, y)-K_2(x, y)\}$ belongs to H_0. Now for any $f \in F$ we have $f(y) = f'(y)+f''(y) = \langle f'(x), K_1(x, y)\rangle_1 +$
$\langle f''(x), K_2(x, y)\rangle_2 = \langle\{f', f''\},\{K_1(x, y), K_2(x, y)\}\rangle = \langle\{f', f''\}$,
$\{K'(x, y), K''(x, y)\}\rangle + \langle\{f', f''\}, \{K_1(x, y) - K'(x, y), K_2(x, y) - K''(x, y)\}\rangle$.

The last scalar product equals zero, since $\{f', f''\} \in H'$ and $\{K_1(x, y) - K'(x, y)$,
$K_2(x, y) - K''(x, y)\} \in H_0$. And the first part of the last expression equals to
$\langle\{f(x), K(x, y)\rangle$ which proves reproducing property of kernel $K(x, y)$.

To prove equivalence of both the definitions of norms we have to remember that $f(x)$ corresponds to $\{f_1, f_2\} \in H$ and also to $\{g'(f), g''(f)\} \in H'$. So we have $f = f_1 + f_2 = g'(f) + g''(f)$. It follows that $g'(f) - f_1 = -(g''(f) - f_2)$ and $\{f_1 - g'(x), f_2 - g''(x)\} \in H_0$. Thus

$$\|f_1\|_1^2 + \|f_2\|_2^2 = \|\{f_1, f_2\}\|^2 = \|\{f_1 - g'(f), f_2 - g''(f)\}\|^2 + \|\{g'(f), g''(f)\}\|^2.$$

This expression attains minimal value for $f_1 = g'(f)$ and $f_2 = g''(f)$ and its value is then $\|\{g'(f), g''(f)\}\|^2$ which is $\|f\|^2$ and proves equivalence of the two definitions. □

It is easy to extend this theorem to $K(x,y) = \sum_{i=1}^{n} K_i(x,y)$.

4 Product of reproducing kernels

Next we will consider product of Reproducing Kernel Hilbert Spaces. For $i = 1, 2$ let F_i be an RKHS of functions on Ω_i, let K_i be the corresponding kernel. Consider the following set of functions on $\Omega = \Omega_1 \times \Omega_2$ $F' = \{\sum_{i=1}^{n} f_{1,i}(x_1) f_{2,i}(x_2) \mid n \in \mathbb{N}, f_1 \in F_1, f_2 \in F_2\}$. Clearly, F' is a vector space, it is not complete though. For its completion, we first define a scalar product on F'. Let f, g be elements of F' expressed as $f(x_1,x_2) = \sum_{i=1}^{n} f_{1,i}(x_1) f_{2,i}(x_2)$, $g(x_1,x_2) = \sum_{j=1}^{m} g_{1,j}(x_1) g_{2,j}(x_2)$. We define $\langle f, g\rangle = \sum_{i=1}^{n}\sum_{j=1}^{m} \langle f_{1,i}, g_{1,j}\rangle_1 \langle f_{2,i}, g_{2,j}\rangle_2$, where $\langle\cdot,\cdot\rangle_i$ denotes the scalar product in F_i. It is a routine to check that this definition does not depend on the particular form in which f and g are expressed and that the properties of scalar product are satisfied. We define norm on F' by $\|f\| = \sqrt{\langle f, f\rangle}$. Finally, let F be the completion of F'. It can be shown ([1]) that the completion exists not only as an abstract Hilbert space but that F is in fact a space of functions on Ω. We call F the product of F_1 and F_2 and write $F = F_1 \otimes F_2$.

Theorem 4.1 *([1]) For $i = 1, 2$ let F_i be an RKHS on Ω_i with kernel K_i. Then the product $F = F_1 \otimes F_2$ on $\Omega_1 \times \Omega_2$ is an RKHS with kernel given by*

$$K((x_1,x_2),(y_1,y_2)) = K_1(x_1,y_1) K_2(x_2,y_2)\,, \tag{4.1}$$

where $x_1, y_1 \in \Omega_1$, $x_2, y_2 \in \Omega_2$.

5 Learning from data as minimization of functionals

The task to find an optimal solution to the problem of approximating a data set $z = \{(u_i, v_i)\}_{i=1}^{N} \subseteq \mathbb{R}^d \times \mathbb{R}$ by a function from a general function space X (minimizing error) is ill-posed. Thus we impose additional (regularization) conditions on the solution ([3]). These are typically things like a-priori knowledge, or some smoothness constraints. The solution f_0 has to minimize a functional $\mathcal{F} : \Omega \to \mathbb{R}$ that is composed of the error part and the "smoothness" part: $\mathcal{F}(f) = \mathcal{E}_z(f) + \gamma\Phi(f)$, where $\mathcal{E}_z$ is the error functional depending on the data $z = \{(u_i, v_i)\}_{i=1}^{N} \subseteq \mathbb{R}^d \times \mathbb{R}$ and penalizing distance from the data, Φ is the regularization part — the so called stabilizer — penalizing "remoteness from the global property" and γ is the regularization parameter giving the trade-off between the two terms of the functional to be minimized.

To prove existence and uniqueness of solution to such a problem we will use some results from mathematical analysis. Error part of our functional doesn't have sufficiently nice properties, so the regularization part has to do the job. We employ RKHS in such a way that we nicely and easily obtain existence, uniqueness and even form of the solution.

Let $\mathcal{H}$ be an RKHS over $\Omega \subseteq \mathbb{R}^d$ with kernel k and norm $\|.\|_k$. We construct the minimization functional composing of error part $\mathcal{E}_z(f)$ based on data $z = \{(u_i, v_i); i = 1, \ldots, N\} \subseteq \mathbb{R}^d \times \mathbb{R}$ and let the regularization part be $\Phi(f) = \|f\|_k^2$ forming

$$\mathcal{F}(f) = \mathcal{E}_z(f) + \gamma\|f\|_k^2, \tag{5.1}$$

where $\gamma \in \mathbb{R}^+$. (See Section 6 for a more detailed construction.)

Now uniqueness of solution to such a problem comes clearly from strong quasi-convexity of the functional $\mathcal{F}$ composing of convex error part and strongly quasi-convex kernel part. To show existence of solution we need weak sequential lower semi-continuity of the functional which can be shown by computing second derivatives of the functional, for precise derivation see [13].

Derivation of the shape of the solution to the regularized minimization problem has been shown already in [3] but without taking advantage of RKHS, in [2], [9] and others known as Representer theorem, for the kernel case see [13]. All the proofs are based on the idea that minimum of a function can exist in an interior point only if first derivative equals zero.

Employing this theorem we obtain solution to the kernel-based minimization problem in the form of

$$f_0(x) = \sum_{i=1}^{N} c_i k(x, u_i), \tag{5.2}$$

where u_i are the data points and $k(\cdot, \cdot)$ the corresponding kernel.

6 Concrete minimization functionals and RKHS

An error functional is usually of the form $\mathcal{E}_z(f) = \sum_{i=1}^{N} V(f(u_i), v_i)$. A typical example of the empirical error functional is the classical mean square error: $\mathcal{E}_z(f) = \frac{1}{N}\sum_{i=1}^{N}(f(u_i) - v_i)^2$.

In [3] a special stabilizer based on the Fourier Transform was proposed: $\Phi_K(f) = \int_{\mathbb{R}^d} \frac{|\hat{f}(s)|^2}{\hat{K}(s)} \mathrm{d}m_d(s)$, where $\hat{K} : \mathbb{R}^d \to \mathbb{R}_+$ is symmetric ($\hat{K}(s) = \hat{K}(-s)$) function tending to zero as $\|s\| \to \infty$ (the last holds for any $K \in \mathcal{L}_1$). That means $1/\hat{K}$ is a low-pass filter.

Thus the functional $\mathcal{F}_K$ to be minimized is of the form: $\mathcal{F}_K(f) = \mathcal{E}_z(f) + \Phi_K(f) = \frac{1}{N}\sum_{i=1}^{N}(f(u_i) - v_i)^2 + \gamma \int_{\mathbb{R}^d} \frac{|\hat{f}(s)|^2}{\hat{K}(s)} \mathrm{d}m_d(s)$, where $\gamma \in \mathbb{R}^+$. Now we show how to build an RKHS corresponding to the regularization part of our functional:

Let us define $k(x, y) = K(x - y) = \int_{\mathbb{R}^d} \hat{K}(t) e^{it.x} e^{-it.y} \mathrm{d}m_d(t)$. For $k \in \mathcal{S}(\mathbb{R}^{2d})$ symmetric positive definite we obtain an RKHS $\mathcal{H}$ (using the classical construction, see [2], [12],[14]). We put $\langle f, g\rangle_{\mathcal{H}} = \int_{\mathbb{R}^d} \frac{\hat{f}(s)\hat{g}^*(s)}{\hat{K}(s)} \mathrm{d}m_n(s)$ and obtain the norm $\|f\|^2_{\mathcal{H}} = \int_{\mathbb{R}^d} \frac{|\hat{f}(s)|^2}{\hat{K}(s)} \mathrm{d}m_n(s)$, for $\mathcal{H} = \mathrm{comp}\,\mathrm{span}\{k(x, .), x \in \mathbb{R}^d\}$, where comp denotes completion of the set and a^* means complex conjugate of a. It is easy to check the reproducing property of K on $\mathcal{H}$, that is $\langle f(x), K(x - y)\rangle_{\mathcal{H}} = f(y)$.

Special types of reproducing kernels and following RKHS are the well known Gaussian kernel $K_1(x, y) = e^{-\|x-y\|^2}$ with Fourier transform $\hat{K_1}(s) = e^{-\frac{\|s\|^2}{2}}$ or in one dimension kernel $K_2(x, y) = e^{-|x-y|}$ with Fourier transform $\hat{K_2}(s) = (1 + s^2)^{-1}$. The norm for this RKHS is of the form $\|f\|_K = \int \frac{|\hat{f}|^2}{(1+s^2)^{-1}} = \|f\|^2_{\mathcal{L}_2} + \|f'\|^2_{\mathcal{L}_2}$. So we see we obtain a Sobolev space W_2^1.

6.1 Minimization of Sum of Kernels

Here we will consider a more sophisticated example of kernels – sum of kernels introduced in Section 3. We will consider two cases. First suppose that a-priori knowledge or analysis of data suggests to look for solution as a sum of two functions (for example data is generated from function influenced by two sources differing in frequency). We will use a kernel summed of two parts (employing Theorem 3.1) corresponding to high and low frequencies, in the easiest case two Gaussians of different widths:

$$K(x, y) = K_1(x, y) + K_2(x, y) = e^{-(\frac{\|x-y\|}{d_1})^2} + e^{-(\frac{\|x-y\|}{d_2})^2}$$

Regularized minimization schema in this case will be of the form:

$$\mathcal{F}_K(f) = \frac{1}{N}\sum_{i=1}^{N}(f(u_i) - v_i)^2 + \gamma \int_{\mathbb{R}^d} \frac{|\hat{f}(s)|^2}{\widehat{(K_1 + K_2)}(s)} \mathrm{d}m_n(s). \tag{6.1}$$

Since we operate in an RKHS we can employ Representer theorem (5.2) and obtain solution in the form of

$$f_0(x) = \sum_{i=1}^{N} c_i \left(e^{-(\frac{\|x-u_i\|}{d_1})^2} + e^{-(\frac{\|x-u_i\|}{d_2})^2} \right). \tag{6.2}$$

Another conceivable task would be to approximate data with different distribution in the input space. Here again sum of kernels might be helpful if we define one kernel for let's say positive axis and one kernel for negative one. If our data isn't distributed so nicely it is always possible to do some pre- and post-processing to achieve this.

First, consider the following lemma from [1].

Lemma 6.1 *Let F be an RKHS of real-valued functions on Ω with K as kernel. Then function K_A defined by*

$$K_A(x,y) = \begin{cases} K(x,y) & \text{if } x,y \in A, \\ 0 & \text{otherwise;} \end{cases}$$

is a kernel for a space $F_A = \{f_A, f \in F\}$, where $f_A(x) = f(x)$ if $x \in A$ and $f_A(x) = 0$ otherwise.

We may use this lemma for different sets A. Then we can apply Theorem 3.1 for kernels gained in this way. Consequently, our kernels may look as follows:

Choose constants $d_1, \ldots, d_s > 0$ and partition $\mathbb{R}$ to sets $A_1, \ldots, A_s$. Then define

$$K(x,y) = \begin{cases} e^{-(\frac{\|x-y\|}{d_i})^2} & x,y \in A_i \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

Again, we may use Representer theorem to obtain a form in which we will expect the solution.

6.2 Minimization of Product Kernels

Now we will consider the product of kernels introduced in Section 4. Suppose that a-priori knowledge of our data suggests to look for the solution as a member of product of two function spaces. In one dimension the data may be clustered faraway thus being suitable for approximation via narrow Gaussian kernels. In the other dimension the data is smooth, hence we will use broader Gaussian kernel. Employing theorem 4.1 we obtain a kernel for the product space of the form:

$K((x_1,x_2),(y_1,y_2)) = k_1(x_1,y_1)\cdot k_2(x_2,y_2) = e^{-(\frac{\|x_1-y_1\|}{d_1})^2} \cdot e^{-(\frac{\|x_2-y_2\|}{d_2})^2}$, where $x_1, y_1 \in \Omega_1$, $x_2, y_2 \in \Omega_2$.

Regularized minimization schema in this case is of the form:

$$\mathcal{F}_K(f) = \frac{1}{N}\sum_{i=1}^{N}(f(u_i)-v_i)^2 + \gamma \int_{\mathbb{R}^d} \frac{|\hat{f}(s)|^2}{\widehat{k_1 k_2}(s)} \mathrm{d}m_n(s). \tag{6.4}$$

Taking advantage of this being an RKHS we have the form of the solution to such a type of minimization:

$$f_0(x_1,x_2) = \sum_{i=1}^{N} c_i e^{-(\frac{\|x_1-u_{i,1}\|}{d_1})^2} \cdot e^{-(\frac{\|x_2-u_{i,2}\|}{d_2})^2}. \tag{6.5}$$

Generally, it is possible to combine different types of kernels, for example for heterogeneous data, where individual attributes are of different types and different kernels are suitable for them.

Approximation schemas of this type exhibit so far nicer approximation properties since it can be better fitted to special types of data.

7 Learning algorithm

Now we present a learning algorithms based on the theoretical results from the previous sections, first for approximation with sum kernels and then for approximation with product kernels.

For the case of sum kernels, we assume that we have a data set $\{u^i, v^i\}_{i=1}^N$, where $u^i \in \mathbb{R}^n$, $v^i \in \mathbb{R}$ and N is a number of data samples. If we fit this data set using the regularization schema 6.1, then the solution can be represented by a feed-forward neural network with one hidden layer of N *sum units* and a linear output layer (see fig. 7.1a).

By a *sum unit* (see fig 7.1b) we mean a unit with n real inputs and one real output. It consists of two positive definite kernel functions $K_1(c,\cdot)$, $K_2(c,\cdot)$, both evaluating the same input vector. The output of the sum unit is computed as the sum $K_1(c,x) + K_2(c,x)$.

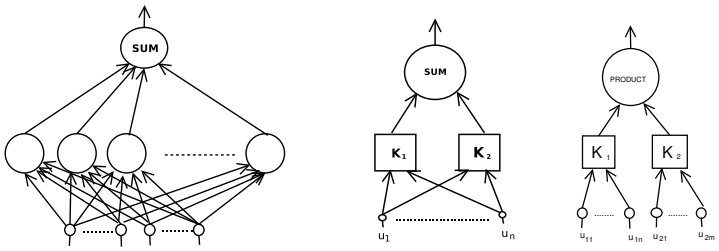

Figure 7.1: a) Regularization Network, b) Sum Unit, c) Product Unit.

Input: Data set $\{u^i, v^i\}_{i=1}^N \subseteq \mathbb{R}^n \times \mathbb{R}$
Output: Sum Kernel Regularization Network.

1. Set the centers of kernels:

$$\forall i \in \{1, \ldots, N\} : c^i \leftarrow u^i$$

2. Compute the values of weights $w_1, \ldots, w_N$:

$$(N\gamma I + K)w = v, \tag{7.1}$$

where I is the identity matrix,
$K_{i,j} = K_1(c^i, u^j) + K(c^i, u^j)$,
and $v = (v^1, \ldots, v^N)$, $\gamma > 0$.

Figure 7.2: Learning algorithm for Sum Kernel Regularization Network.

The network then evaluates the function

$$f(x) = \sum_{i=1}^{N} w_i (K_1(c^i, x) + K_2(c^i, x)),$$

and we refer to it as Sum Kernel Regularization Network (SKRN).

Let us call the parameters c^i *centers* and the coefficients of the linear combination w_i *weights*.

The learning algorithm of Sum Kernel Regularization network is sketched at Fig. 7.2. It is derived from Tikhonov regularization and for the case of Regularization Network it was described in [9]. For the discussion of learning algorithms of Regularization Networks see also [4].

The algorithm is quite simple, setting the centers of kernels to the data points given by the training set and evaluating the values of output weights by solving linear system of equations. Its drawback is the presence of parameter γ, that must be estimated in advance (cross-validation is usually used).

For the case of Product Kernels, we assume that we have a data set $\{u_1^i, u_2^i, v^i\}_{i=1}^N$, where $u_1^i \in \mathbb{R}^n$, $u_2^i \in \mathbb{R}^m$, $v^i \in \mathbb{R}$ and N is a number of data samples. We will fit these data set using the regularization scheme 6.4.

Again, the solution is represented by a feed-forward neural network with one hidden layer, now consisting of N *product units*, and a linear output layer.

By a *product unit* (see fig 7.1c) we mean an unit with $(n+m)$ real inputs and one real output. It consists of two positive definite kernel functions $K_1(c_1, \cdot)$, $K_2(c_2, \cdot)$, one evaluating the first n inputs and one evaluating the other m inputs. The output of the product unit is computed as the product $K_1(c_1, x_1) \cdot K_2(c_2, x_2)$.

The network then evaluates the function

$$f(x_1, x_2) = \sum_{i=1}^{N} w_i K_1(c_1^i, x_1) \cdot K_2(c_2^i, x_2),$$

Input: Data set $\{u_1^i, u_2^i, v^i\}_{i=1}^N \subseteq \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}$
Output: Product Kernel Regularization Network.

1. Set the centers of kernels:

$$\forall i \in \{1, \ldots, N\}: \quad c_1^i \leftarrow u_1^i$$
$$c_2^i \leftarrow u_2^i$$

2. Compute the values of weights $w_1, \ldots, w_N$:

$$(N\gamma I + K)w = v,$$

where I is the identity matrix,
$K_{i,j} = K_1(c_1^i, u_1^j) \cdot K_2(c_2^i, u_2^j)$,
and $v = (v^1, \ldots, v^N)$, $\gamma > 0$.

Figure 7.3: Learning algorithm for Product Kernel Regularization Network.

and we refer to such network as Product Kernel Regularization Network (PKRN).

The learning algorithm of PKRN is similar to the learning algorithm of SKRN, and is sketched at Fig. 7.3.

In general, both the sum unit and the product unit can consist of arbitrary number (more than two) of kernel functions. The application of given theory and algorithms on these cases is strait-forward.

8 Experiments

The goal of our experiments was to demonstrate the performance of proposed Product Kernel Regularization Network and Sum Kernel Regularization Network, and compare them with classical Regularization Network with Gaussian kernels.

8.1 Methodology

For the comparison we have chosen the Proben1 data repository (see [10]) containing both approximation and classification tasks. The short description of Proben1 tasks is listed in table 8.1. Each task is present in three variants, three different partitioning into training and testing data. We refer to this variants with suffix 1,2, or 3 (e.g. *cancer1, cancer2, cancer3*).

In addition we applied the Product Kernel Regularization Network on the real-life task, the prediction of the flow rate on the Czech river Ploučnice.

Gaussian kernels were used in all experiments. For each experiment, we first estimated the explicit parameters of the learning algorithms, namely the regularization parameter γ and the width(s) of Gaussians. Parameters with the lowest cross-validation error on the training set are chosen and used to learn the network on the whole training set. Than the error on the testing set is evaluated, as a measure of a real performance of the resulting network.

We always evaluated the normalized error:

$$E = 100\frac{1}{N}\sum_{i=1}^{N} ||v^i - f(u^i)||^2,$$

where N is number of examples and f is the network output.

For the parameter search we used k-fold cross-validation. It means that we divided the training set TrS to k-folds $T_1, \ldots, T_k$ of an approximately same size, such as $\bigcup_{i=1}^k T_i = TrS$ and $T_i \bigcap T_j = 0, i \neq j$, and for each i run the learning algorithm 7.2 (resp. 7.3) on the data set $TrS_i = \bigcup_{j \neq i} T_j$. To the network obtained by learning on data TrS_i we refer as f^i. Then the cross-validation error can be computed as

Task name	n	m	N_{train}	N_{test}	**Type**
cancer	9	2	525	174	class
card	51	2	518	172	class
diabetes	8	2	576	192	class
flare	24	3	800	266	approx
glass	9	6	161	53	class
heartac	35	1	228	75	approx
hearta	35	1	690	230	approx
heartc	35	2	228	75	class
heart	35	2	690	230	class
horse	58	3	273	91	class
soybean	82	19	513	170	class

Table 8.1: Overview of Proben1 tasks. Number of inputs (n), number of outputs (m), number of samples in training and testing sets (N_{train},N_{test}). Type of task: approximation or classification.

$$E_{cross} = \frac{1}{k}\sum_{i=1}^{k} E_i \text{ ,where } E_i = \frac{1}{|T_i|}\sum_{(u_j,v_j)\in T_i} ||v^j - f^i(u^j)||^2.$$

Grid search is used to find the parameters with the lowest cross-validation error. We start with a coarse grid of parameters $[\gamma, w_1, w_2]_j$ (sometimes more than two widths are needed, in case of classical RN only one width is needed). We evaluate the cross-validation error for each tuple and create a finer grid around the point with the lowest cross-validation error. This process is repeated until the cross-validation error stops decreasing.

The standard numerical library LAPACK [6] was used for linear system solving.

8.2 Regularization Networks

First of all, we applied the classical Regularization Network with Gaussian kernels.

Table 8.2 summarizes the results obtained on data tasks from Proben1. For each task errors on the training and testing set are listed, together with the parameters' values found by the grid search and number of evaluations (runs of the basic algorithm) needed to find these values.

8.3 Sum Kernel Regularization Networks

Another experiments were performed to demonstrate the behavior of SKRN. We have tried both two types of Sum Kernels, the former is a sum of two Gaussians (see 6.2), the letter is a sum of Gaussians with non-zero output only on particular subsets of an input space (see 6.3). We refer to them SKRN$_A$ and SKRN$_B$, respectively.

In the case of SKRN$_B$, the data sets were divided into two or three disjunct subsets, on each of them was active one kernel. This enables us to replace large linear systems by two (resp. three) small ones (for individual subsets) that can be solved separately, possibly in parallel. This *Divide et Impera* strategy may help in cases, where the solution of the whole linear system is too expensive due to the large size of data set.

The table 8.3 shows the results obtained on Proben1 with SKRN$_A$, again errors on training and testing sets, winning values for parameters and number of evaluations are listed.

This experiment showed an interesting behavior on several data sets. The error on the training set is almost a zero (rounded to zero) and still the generalization ability of the network is good, i.e. the error on testing set is not high. This is caused by the fact, that the chosen kernel consists of two Gaussians, one being very narrow (see fig. 8.1). The diagonal in matrix K from 7.1 is dominant and so regularization member is not needed, precisely γ is near to zero.

Task	E_{train}	E_{test}	γ	b	evaluations
cancer1	2.28	1.75	0.30×10^{-3}	1.57	96
cancer2	1.86	3.01	0.20×10^{-3}	1.60	76
cancer3	2.11	2.79	0.55×10^{-3}	1.53	97
card1	8.75	10.01	2.03×10^{-3}	4.02	126
card2	7.55	12.53	1.29×10^{-3}	4.18	101
card3	6.52	12.35	0.33×10^{-3}	4.28	113
diabetes1	13.97	16.02	1.32×10^{-3}	1.04	117
diabetes2	14.00	16.77	2.16×10^{-3}	1.01	101
diabetes3	13.69	16.01	0.23×10^{-3}	1.47	112
flare1	0.36	0.55	11.52×10^{-3}	3.37	76
flare2	0.42	0.28	3.96×10^{-3}	3.24	60
flare3	0.38	0.35	3.16×10^{-3}	2.65	36
glass1	3.37	6.99	2.54×10^{-3}	0.31	165
glass2	4.32	7.93	2.20×10^{-3}	0.52	137
glass3	3.96	7.25	2.75×10^{-3}	0.38	72
heart1	9.61	13.66	1.97×10^{-3}	2.70	57
heart2	9.33	13.83	1.97×10^{-3}	2.70	57
heart3	9.23	15.99	1.06×10^{-3}	3.76	117
hearta1	3.42	4.38	0.45×10^{-3}	4.64	134
hearta2	3.54	4.07	0.54×10^{-3}	4.64	134
hearta3	3.44	4.43	0.58×10^{-3}	4.57	122
heartac1	4.22	2.76	0.98×10^{-3}	8.20	496
heartac2	3.50	3.86	0.66×10^{-3}	6.71	342
heartac3	3.36	5.01	0.84×10^{-3}	7.35	405
heartc1	9.99	16.07	0.56×10^{-3}	11.81	900
heartc2	12.70	6.13	0.31×10^{-3}	11.81	917
heartc3	8.79	12.68	2.10×10^{-3}	3.24	121
horse1	7.35	11.90	3.58×10^{-3}	3.40	121
horse2	7.97	15.14	4.09×10^{-3}	3.81	117
horse3	4.26	13.61	2.16×10^{-3}	2.85	85
soybean1	0.12	0.66	0.08×10^{-3}	3.24	57
soybean2	0.24	0.50	0.18×10^{-3}	3.51	85
soybean3	0.23	0.58	0.15×10^{-3}	3.67	89

Table 8.2: Results obtained by Regularization Network with Gaussian kernels on data tasks from Proben1 repository. Winning values for parameters γ and b and number of evaluations needed for their estimation are listed.

Task	E_{train}	E_{test}	γ	b_1	b_2	evaluations
cancer1	0.00	1.77	0.00×10^{-3}	2.116	0.248	664
cancer2	0.00	2.96	0.00×10^{-3}	2.063	0.220	624
cancer3	0.00	2.73	0.00×10^{-3}	1.953	0.152	292
card1	8.81	10.03	4.19×10^{-3}	4.079	4.061	472
card2	0.00	12.54	0.00×10^{-3}	3.542	0.038	376
card3	6.55	12.32	1.32×10^{-3}	4.178	3.145	387
diabetes1	14.01	16.00	2.74×10^{-3}	3.237	0.881	239
diabetes2	13.78	16.80	3.67×10^{-3}	3.419	0.801	520
diabetes3	13.69	15.95	0.39×10^{-3}	3.096	1.306	399
flare1	0.35	0.54	13.59×10^{-3}	2.833	2.833	200
flare2	0.44	0.26	31.62×10^{-3}	2.650	2.650	164
flare3	0.42	0.33	31.62×10^{-3}	2.650	2.650	164
glass1	2.35	6.15	3.00×10^{-3}	0.206	0.985	439
glass2	1.09	6.97	2.37×10^{-3}	0.140	0.764	699
glass3	3.04	6.29	4.29×10^{-3}	0.229	1.013	724
heart1	0.00	13.91	0.02×10^{-3}	3.143	0.091	600
heart2	0.00	13.82	0.00×10^{-3}	2.968	0.000	260
heart3	0.00	15.94	0.00×10^{-3}	3.102	0.000	324
hearta1	0.00	4.37	0.00×10^{-3}	3.412	0.044	532
hearta2	3.51	4.06	1.05×10^{-3}	4.572	4.572	478
hearta3	0.00	4.49	0.00×10^{-3}	3.470	0.038	372
heartac1	0.00	3.26	0.00×10^{-3}	4.572	0.232	483
heartac2	0.00	3.85	0.00×10^{-3}	3.748	0.085	500
heartac3	3.36	5.01	1.68×10^{-3}	7.349	7.349	1588
heartc1	0.00	15.69	0.00×10^{-3}	4.644	0.115	470
heartc2	0.00	6.33	0.00×10^{-3}	4.334	0.096	680
heartc3	0.00	12.38	0.00×10^{-3}	2.923	0.067	760
horse1	0.20	11.90	0.26×10^{-3}	3.873	0.000	408
horse2	2.84	15.11	3.12×10^{-3}	5.281	1.715	768
horse3	0.18	14.13	0.00×10^{-3}	3.774	1.050	328
soybean1	0.11	0.66	0.15×10^{-3}	4.043	2.725	367
soybean2	0.25	0.53	0.80×10^{-3}	2.833	2.833	175
soybean3	0.22	0.57	0.28×10^{-3}	4.043	3.469	367

Table 8.3: Results obtained by Regularization Network with sum kernels (2 Gaussians) on data tasks from Proben1 repository. Winning values for parameters γ and b_1, b_2 and number of evaluations needed for their estimation are listed.

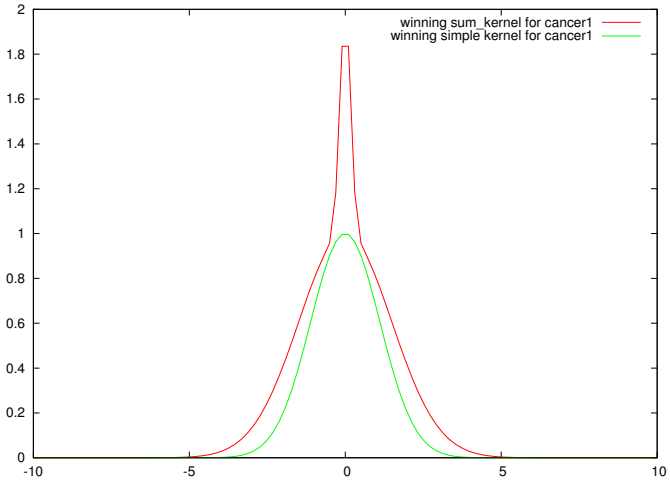

Figure 8.1: Kernels found by parameter search for cancer1 data set.

The condition numbers of matrices K and $K + N\gamma I$ (see Fig.7.2,7.3) are displayed on Fig. 8.2. On some tasks the condition number is really lower for the case of Sum Kernels, but we can not see any significant trend.

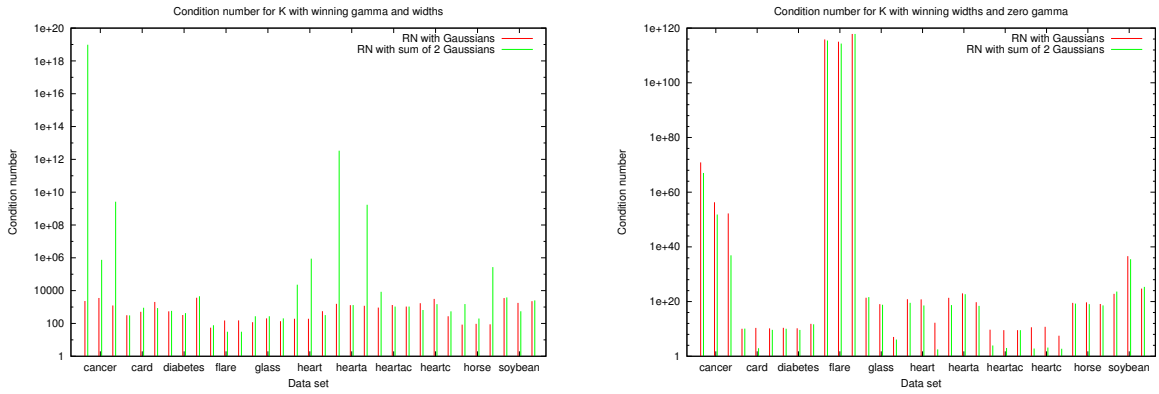

Figure 8.2: Condition numbers of matrix $K + N\gamma I$ (left) and K (right) for Gaussian Kernels and Sum Kernels.

The results obtain by SKRN$_B$ are listed in table 8.4. The *Divide et Impera* strategy significantly decreases the time requirements. See Fig. 8.3 for the comparison of time needed by classical RN and SKRN$_B$.

8.4 Product Kernel Regularization Network

The PKRN should be used in situations, where we have some knowledge about a character of data, especially when different attributes are of different types or different properties.

This is not the case of Proben1 tasks, so the application of PKRN on these tasks is in this sense blind and the partitioning of inputs to several groups was chosen randomly. The table 8.5 summarizes the results obtained by PKRN on Proben1 tasks, including training and testing errors, winning parameters and number of evaluations.

Task	E_{train}	E_{test}	γ_1	γ_2	γ_3	b_1	b_2	b_3
cancer1	2.11	1.93	0.00×10^{-3}	0.76×10^{-3}	0.87×10^{-3}	1.05	1.60	2.83
cancer2	1.68	3.37	0.00×10^{-3}	0.55×10^{-3}	0.69×10^{-3}	0.86	1.65	3.01
cancer3	1.68	2.95	0.00×10^{-3}	0.55×10^{-3}	0.02×10^{-3}	1.18	1.44	6.26
card1	8.55	10.58	0.56×10^{-3}	4.78×10^{-3}	-	10.76	2.87	-
card2	7.22	13.03	0.91×10^{-3}	3.15×10^{-3}	-	8.20	2.93	-
card3	6.22	12.86	3.29×10^{-3}	1.80×10^{-3}	-	3.80	2.43	-
diabetes1	12.92	16.66	3.00×10^{-3}	2.42×10^{-3}	9.48×10^{-3}	0.68	0.87	0.85
diabetes2	13.64	17.33	4.23×10^{-3}	4.66×10^{-3}	5.99×10^{-3}	1.18	1.23	0.74
diabetes3	12.85	16.34	1.91×10^{-3}	2.33×10^{-3}	1.51×10^{-3}	0.89	1.42	1.01
flare1	0.35	0.59	316.23×10^{-3}	73.56×10^{-3}	8.26×10^{-3}	1.30	2.83	4.04
flare2	0.41	0.28	1.32×10^{-3}	25.25×10^{-3}	7.32×10^{-3}	4.18	2.70	3.24
flare3	0.38	0.34	2.02×10^{-3}	25.25×10^{-3}	3.96×10^{-3}	4.57	3.24	3.24
glass1	2.56	6.78	3.57×10^{-3}	2.85×10^{-3}	-	0.28	0.47	-
glass2	3.27	7.29	5.39×10^{-3}	3.41×10^{-3}	-	0.35	0.67	-
glass3	3.48	6.44	4.31×10^{-3}	7.02×10^{-3}	-	0.33	0.53	-
heart1	9.51	13.79	2.39×10^{-3}	1.92×10^{-3}	-	2.89	4.20	-
heart2	8.52	14.31	1.80×10^{-3}	1.69×10^{-3}	-	2.46	4.08	-
heart3	8.30	16.75	1.72×10^{-3}	2.30×10^{-3}	-	2.60	4.18	-
hearta1	3.20	4.45	0.73×10^{-3}	0.18×10^{-3}	-	3.77	8.20	-
hearta2	3.17	4.34	2.05×10^{-3}	0.98×10^{-3}	-	2.87	4.64	-
hearta3	3.37	4.40	4.32×10^{-3}	0.47×10^{-3}	-	2.83	6.39	-
heartac1	3.68	3.37	6.82×10^{-3}	2.05×10^{-3}	-	2.91	8.20	-
heartac2	2.99	3.97	1.11×10^{-3}	1.10×10^{-3}	-	3.77	8.20	-
heartac3	3.14	5.13	1.69×10^{-3}	4.21×10^{-3}	-	4.57	6.55	-
heartc1	6.50	16.07	0.00×10^{-3}	0.47×10^{-3}	-	1.74	14.61	-
heartc2	11.06	6.69	4.16×10^{-3}	0.41×10^{-3}	-	3.39	14.61	-
heartc3	9.91	11.74	3.41×10^{-3}	3.27×10^{-3}	-	3.25	8.20	-
horse1	7.66	12.62	11.75×10^{-3}	5.21×10^{-3}	-	3.74	2.70	-
horse2	6.84	15.70	7.09×10^{-3}	4.78×10^{-3}	-	3.88	3.92	-
horse3	8.56	15.24	12.28×10^{-3}	4.33×10^{-3}	-	4.57	3.95	-
soybean1	0.12	0.64	0.09×10^{-3}	0.36×10^{-3}	0.25×10^{-3}	4.18	4.18	4.18
soybean2	0.19	0.54	0.47×10^{-3}	0.25×10^{-3}	0.37×10^{-3}	3.68	4.64	4.57
soybean3	0.15	0.72	0.00×10^{-3}	0.20×10^{-3}	0.30×10^{-3}	3.10	5.92	3.77

Table 8.4: Results obtained by Regularization Network with sum kernels (Gaussians active on different subsets of input space) on data tasks from Proben1 repository. Winning values for parameters γ and width for each Gaussian and number of evaluations needed for their estimation are listed. Because learning algorithm was run separately on different subset of input space, also different values of γ were used.

Task	inputs	E_{train}	E_{test}	γ	b_1	b_2	b_3	evals
cancer1	(3-3-3)	3.39	2.94	0.29×10^{-3}	2.01	0.98	0.98	1732
cancer1	(3-6)	2.68	1.81	0.35×10^{-3}	0.77	4.18	–	396
cancer1	(6-3)	2.68	1.81	0.35×10^{-3}	4.18	0.77	–	396
cancer2	(3-3-3)	2.68	4.92	0.29×10^{-3}	1.83	0.77	0.93	1664
cancer2	(3-6)	2.07	3.61	0.57×10^{-3}	0.58	3.77	–	412
cancer2	(6-3)	2.07	3.61	0.57×10^{-3}	3.77	0.58	–	412
cancer3	(3-3-3)	3.39	3.65	0.40×10^{-3}	1.61	1.02	1.89	1285
cancer3	(3-6)	2.28	2.81	0.29×10^{-3}	0.89	2.60	–	415
cancer3	(6-3)	2.28	2.81	0.29×10^{-3}	2.60	0.89	–	415
card1	(5-1-45)	9.22	9.99	1.49×10^{-3}	8.20	8.20	4.46	6618
card2	(5-1-45)	7.96	12.90	1.70×10^{-3}	8.20	8.20	3.83	6925
card3	(5-1-45)	6.94	12.23	1.38×10^{-3}	8.20	8.20	2.61	6930
diabetes1	(3-2-3)	16.44	16.75	0.15×10^{-3}	2.72	2.47	2.47	944
diabetes2	(3-2-3)	15.87	18.14	0.23×10^{-3}	2.83	1.05	2.33	773
diabetes3	(3-2-3)	16.31	16.62	0.36×10^{-3}	2.11	1.80	1.70	1200
flare1	(18-2-4)	0.36	0.54	13.66×10^{-3}	2.97	1.05	3.24	1008
flare2	(18-2-4)	0.42	0.28	3.16×10^{-3}	2.65	1.28	2.65	756
flare3	(18-2-4)	0.40	0.34	3.96×10^{-3}	3.24	0.00	3.24	1092
glass1	(3-3-3)	2.33	8.52	0.54×10^{-3}	0.19	0.20	0.15	1568
glass1	(3-6)	2.64	7.31	1.50×10^{-3}	0.20	0.30	–	567
glass2	(3-3-3)	4.53	8.64	2.86×10^{-3}	0.44	0.22	0.28	2048
glass2	(3-6)	2.55	7.46	1.58×10^{-3}	0.20	0.33	–	667
glass3	(3-3-3)	4.98	8.26	4.00×10^{-3}	0.26	0.37	0.22	2540
glass3	(3-6)	3.31	7.26	3.15×10^{-3}	0.58	0.21	–	424
heart1	(1-34)	9.56	13.67	1.32×10^{-3}	1.46	3.08	–	472
heart2	(1-34)	9.43	13.86	1.56×10^{-3}	2.64	2.95	–	503
heart3	(1-34)	9.15	16.06	0.58×10^{-3}	1.08	4.57	–	487
hearta1	(1-17-17)	4.63	5.96	1.03×10^{-3}	1.68	4.57	4.57	1948
hearta1	(1-34)	3.47	4.39	0.19×10^{-3}	2.18	6.39	–	1175
hearta2	(1-17-17)	4.58	5.24	1.11×10^{-3}	0.68	3.77	3.67	1440
hearta2	(1-34)	3.28	4.29	0.48×10^{-3}	0.78	4.57	–	476
hearta3	(1-17-17)	4.60	5.57	1.32×10^{-3}	1.23	3.91	3.91	1621
hearta3	(1-34)	3.40	4.44	0.45×10^{-3}	4.64	4.64	–	514
heartac1	(1-17-17)	4.84	5.85	1.80×10^{-3}	4.18	2.56	2.23	1425
heartac1	(1-34)	4.22	2.76	0.99×10^{-3}	8.20	8.20	–	1944
heartac2	(1-17-17)	5.43	6.66	2.01×10^{-3}	2.05	6.55	6.55	4768
heartac2	(1-34)	3.49	3.87	0.63×10^{-3}	6.71	6.71	–	1346
heartac3	(1-17-17)	5.11	6.52	1.30×10^{-3}	4.19	4.09	4.28	2369
heartac3	(1-34)	3.26	5.18	7.36×10^{-3}	2.83	2.83	–	200
heartc1	(1-34)	10.00	16.08	0.57×10^{-3}	11.15	11.70	–	3528
heartc2	(1-34)	12.37	6.29	0.28×10^{-3}	1.87	11.11	–	3375
heartc3	(1-34)	8.71	12.65	2.39×10^{-3}	4.57	3.07	–	496
horse1	(16-16-16)	14.83	13.44	8.21×10^{-3}	4.64	4.64	4.64	2020
horse1	(29-29)	14.25	12.45	6.92×10^{-3}	7.51	7.20	–	1644
horse2	(16-16-16)	13.09	17.33	8.33×10^{-3}	4.13	3.57	3.56	2140
horse2	(29-29)	12.24	15.97	3.88×10^{-3}	6.71	6.71	–	1332
horse3	(16-16-16)	7.96	18.31	1.94×10^{-3}	2.65	2.17	2.02	2236
horse3	(29-29)	9.63	15.88	3.58×10^{-3}	3.77	3.51	–	479
soybean1	(41-41)	0.13	0.86	0.11×10^{-3}	3.67	3.47	–	351
soybean2	(41-41)	0.23	0.71	0.18×10^{-3}	3.90	3.48	–	463
soybean3	(41-41)	0.21	0.78	0.15×10^{-3}	3.77	4.04	–	367

Table 8.5: Results obtained by PKRN on data tasks from Proben1 repository. Winning values for parameters γ and widths, and number of evaluations needed for their estimation are listed. The numbers $(x_1 - x_2 - x_3)$ describe the partitioning of inputs (in some cases we tried more possibilities).

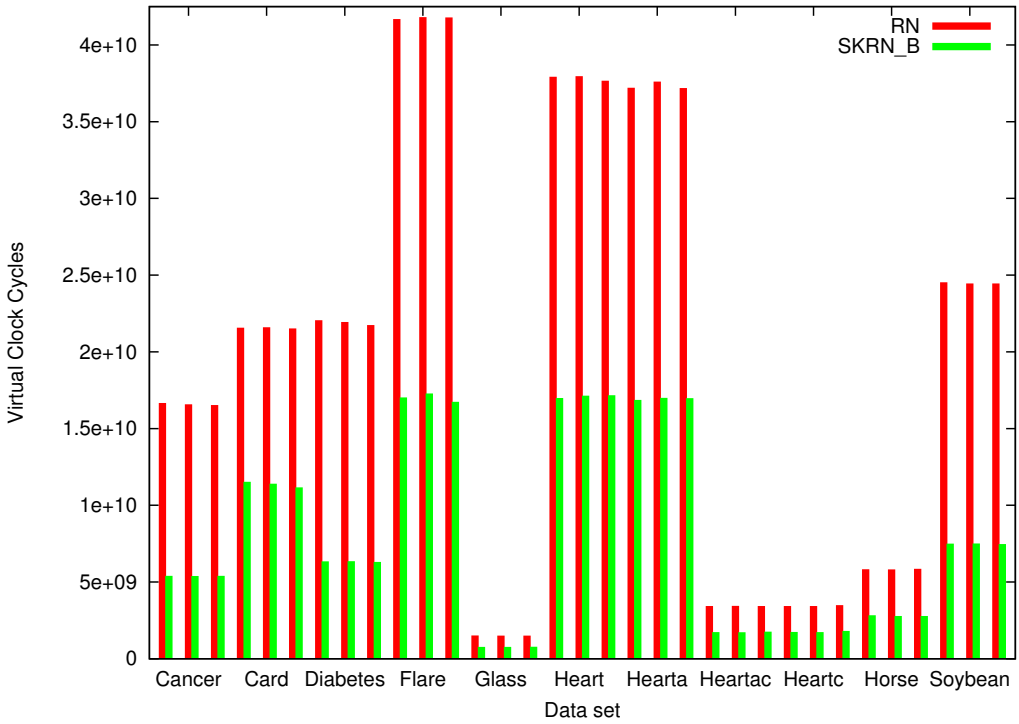

Figure 8.3: Time (in clock cycles [8]) needed for on run of learning algorithm for RN and SKRN$_B$.

8.5 Comparison of RN, PKRN and SKRN

The table 8.6 brings a comparison of errors achieved by RN, PKRN, SKRN$_A$ and SKRN$_B$. We can see that all types of Regularization Networks achieved comparable results in terms of errors on testing set.

However the networks of type SKRN$_A$ achieved almost zero error on the training set on many tasks, preserving the generalization ability. In the addition, they achieved the lowest error on test set in majority of cases.

The networks of type SKRN$_B$ exhibit the worst performance in most cases. This slight increase of the error is the cost for the important decrease of time requirements (see 8.3).

8.6 Prediction of the flow rate

The applicability of PKRN on real life problems is demonstrated on the prediction of the flow rate on the Czech river Ploučnice. Our goal is to predict the current flow rate from the flow rate and total rainfall from the previous date , i.e. we are approximating function $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. Data set contains 1000 training samples and 367 testing samples.

This task is challenging, since very low error values can be achieved using so called *conservative prediction*. Conservative predictor (CP) is a predictor saying that the value will be the same as it was yesterday. In spite of its simplicity, it is very successful on this task, and so the neural networks tend to learn this conservative prediction.

We applied PKRN on this task, applying one Gaussian kernel on the flow rate and another Gaussian kernel on the total rainfall. In spite of being very close to conservative prediction, it over-performs the CP.

The table 8.7 shows the errors achieved by the CP and the PKRN. The prediction of the flow rate on the testing set made by PKRN is displayed at Fig. 8.4.

	RN		$SKRN_A$		PKRN		$SKRN_B$	
Task	E_{train}	E_{test}	E_{train}	E_{test}	E_{train}	E_{test}	E_{train}	E_{test}
cancer1	2.28	**1.75**	0.00	1.77	2.68	1.81	2.11	1.93
cancer2	1.86	3.01	0.00	**2.96**	2.07	3.61	1.68	3.37
cancer3	2.11	2.79	0.00	**2.73**	2.28	2.81	1.68	2.95
card1	8.75	**10.01**	8.81	10.03	9.22	9.99	8.55	10.58
card2	7.55	**12.53**	0.00	12.54	7.96	12.90	7.22	13.03
card3	6.52	12.35	6.55	12.32	6.94	**12.23**	6.22	12.86
diabetes1	13.97	16.02	14.01	**16.00**	16.44	16.75	12.92	16.66
diabetes2	14.00	**16.77**	13.78	16.80	15.87	18.14	13.64	17.33
diabetes3	13.69	16.01	13.69	**15.95**	16.31	16.62	12.85	16.34
flare1	0.36	0.55	0.35	**0.54**	0.36	**0.54**	0.35	0.59
flare2	0.42	0.28	0.44	**0.26**	0.42	0.28	0.41	0.28
flare3	0.38	0.35	0.42	**0.33**	0.40	0.35	0.38	0.34
glass1	3.37	6.99	2.35	**6.15**	2.64	7.31	2.56	6.78
glass2	4.32	7.93	1.09	**6.97**	2.55	7.46	3.27	7.29
glass3	3.96	7.25	3.04	**6.29**	3.31	7.26	3.48	6.44
heart1	9.61	**13.66**	0.00	13.91	9.56	13.67	9.51	13.79
heart2	9.33	13.83	0.00	**13.82**	9.43	13.86	8.52	14.31
heart3	9.23	15.99	0.00	**15.94**	9.15	16.06	8.30	16.75
hearta1	3.42	4.38	0.00	**4.37**	3.47	4.39	3.20	4.45
hearta2	3.54	4.07	3.51	**4.06**	3.28	4.29	3.17	4.34
hearta3	3.44	4.43	0.00	4.49	3.40	4.44	3.37	**4.40**
heartac1	4.22	**2.76**	0.00	3.26	4.22	**2.76**	3.68	3.37
heartac2	3.50	3.86	0.00	**3.85**	3.49	3.87	2.99	3.97
heartac3	3.36	**5.01**	3.36	**5.01**	3.26	5.18	3.14	5.13
heartc1	9.99	16.07	0.00	**15.69**	10.00	16.08	6.50	16.07
heartc2	12.70	**6.13**	0.00	6.33	12.37	6.29	11.06	6.69
heartc3	8.79	12.68	0.00	12.38	8.71	12.65	9.91	**11.74**
horse1	7.35	**11.90**	0.20	**11.90**	14.25	12.45	7.66	12.62
horse2	7.97	15.14	2.84	**15.11**	12.24	15.97	6.84	15.70
horse3	4.26	**13.61**	0.18	14.13	9.63	15.88	8.56	15.24
soybean1	0.12	0.66	0.11	0.66	0.13	0.86	0.12	**0.64**
soybean2	0.24	**0.50**	0.25	0.53	0.23	0.71	0.19	0.54
soybean3	0.23	0.58	0.22	**0.57**	0.21	0.78	0.15	0.72

Table 8.6: Comparisons of errors on training and testing set for RN with Gaussian kernels and SKRN and PKRN.

	PKRN	**CP**
E_{train}	0.057	0.093
E_{test}	0.048	0.054

Table 8.7: Comparison of errors obtained by PKRN and conservative predictor

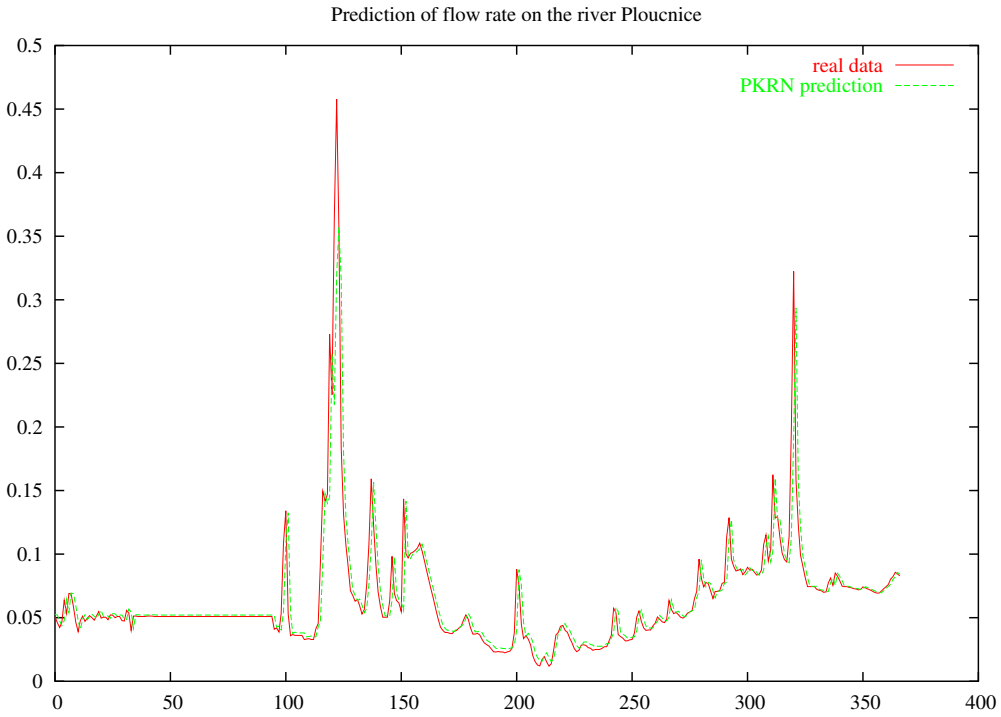

Figure 8.4: Prediction of the flow rate on the river Ploucnice by Product Kernel Regularization Network.

9 Conclusion

We have shown how to employ RKHS in approximation theory and stressed advantages of this approach. Inspired by the article [1] we introduced kernel-product and sum of kernels based approximation and derived the shape of Sum and Product Kernel Regularization Networks.

We compared proposed PKRN and SKRN to classical Regularization Network on benchmarks. All methods gave comparable results, though our SKRN achieved lowest errors in most cases. We also demonstrated how SKRN can be used to decrease the time requirements for larger data sets. In addition, we demonstrated the performance of PKRN on prediction of a river flow rate and showed that it performs better than CP.

We showed that our algorithms are vital alternative to classical RNs. We can benefit from them in situations where some knowledge of the character of data is available or if we can expect that for some groups of inputs different kernel functions are suitable.

Bibliography

[1] N. Aronszajn. Theory of reproducing kernels. *Transactions of the AMS*, 68:337–404, 1950.

[2] F. Girosi. An equivalence between sparse approximation and support vector machines. Technical report, Massachutesetts Institute of Technology, 1997. A.I. Memo No. 1606.

[3] F. Girosi, M. Jones, and T. Poggio. Regularization theory and Neural Networks architectures. *Neural Computation*, 2:219–269, 7 1995.

[4] P. Kudová. Kernel based regularization networks and RBF networks. In *Doktorandský Den 2004, Paseky nad Jizerou*, 2004.

[5] P. Kudová and Šámalová T. Product kernel regularization networks. In *Proceedings of ICANNGA, 2005*. Springer-Verlag, 2205.

[6] LAPACK. Linear algebra package, `http://www.netlib.org/lapack/`.

[7] J. Lukeš. *Zápisky z funkcionální analýzy*. Karolinum, UK Praha, 2002.

[8] PAPI. Performance application programming interface, `http://icl.cs.utk.edu/papi/`.

[9] T. Poggio and S. Smale. The mathematics of learning: Dealing with data. *Notices of the AMS*, 50:536–544, 5 2003.

[10] L. Prechelt. PROBEN1 – a set of benchmarks and benchmarking rules for neural network training algorithms. Technical Report 21/94, Universitaet Karlsruhe, 9 1994.

[11] W. Rudin. *Functional Analysis. 2nd Edition*. McGraw-Hill, NY., 1991.

[12] B. Schoelkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, Massachusetts, 2002.

[13] T. Šidlofová. Existence and uniqueness of minimization problems with fourier based stabilizers. In *Proceedings of Compstat, Prague*, 2004.

[14] G. Wahba. Spline models for observational data. *Series in Applied Mathematics*, 59, 1990.