Product Kernel Regularization Networks

Kudová Petra, Šámalová Terezie
Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, P.0. Box 5, 182 07 Prague 8, Czech Republic
E-mail: {petra, terka}@cs.cas.cz

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 prove uniqueness and existence of solution to the problem. We exploit the article by N. Aronszajn [1] on reproducing kernels and use his formulation of product of kernels and resulting kernel space to derive a new approximation schema – a Product Kernel Regularization Network. We present a concrete application of PKRN and compare it to classical Regularization Network and show that PKRN exhibit better approximation properties.

1 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 $\mathrm{comp}\{\sum_{i=1}^{n} a_i k_{x_i}; x_i \in \Omega, a_i \in \mathbb{R}\}$, where comp means completion of the set.

Next we will consider product of Reproducing Kernel Hilbert Spaces. For $i = 1, 2$ let F_i be a 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 1.1 *([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*

$$K((x_1,x_2),(y_1,y_2)) = K_1(x_1,y_1) K_2(x_2,y_2), \quad (1)$$

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

for proofs of the sketched properties we ask the reader to refer to [1] or to [12].

2 Learning from data as minimization of functionals

The task to find an optimal solution to the setting 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 ([5]). 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 remoteness 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 exhibit 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

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

where $\gamma \in \mathbb{R}^+$. (See section 2.1 for a more detailed construction.)

Now uniqueness of solution to such a problem comes clearly from strong quasiconvexity of the functional $\mathcal{F}$ composing of convex error part and strongly quasiconvex kernel part. To show existence of solution we need weak sequential lower semicontinuity of the functional which can be shown by computing second derivatives of the functional, for precise derivation see [11].

Derivation of the shape of the solution to the regularized minimization problem has been shown already in [5] but without taking advantage of RKHS, in [4], [8] and others known as Representer theorem, for the kernel case see [11]. 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

$$f_0(x) = \sum_{i=1}^{N} c_i k(x, u_i), \tag{3}$$

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

2.1 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 [5] a special stabilizer based on the Fourier Transform was proposed: $\Phi_G(f) = \int_{\mathbb{R}^d} \frac{|\hat{f}(s)|^2}{\hat{G}(s)} \mathrm{d}m_d(s)$, where $\hat{G} : \mathbb{R}^d \to \mathbb{R}_+$ is symmetric ($\hat{G}(s) = \hat{G}(-s)$) function tending to zero as $\|s\| \to \infty$ (the last holds for any $G \in \mathcal{L}_1$). That means $1/\hat{G}$ is a low-pass filter.

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

Let us define $g(x, y) = G(x - y) = \int_{\mathbb{R}^d} \hat{G}(t) e^{it.x} e^{-it.y} \mathrm{d}m_d(t)$. For $g \in \mathcal{S}(\mathbb{R}^{2d})$ symmetric positive definite we obtain an RKHS $\mathcal{H}$ (using the classical construction, see [4], [10],[13]). We put $\langle f, g\rangle_{\mathcal{H}} = \int_{\mathbb{R}^d} \frac{\hat{f}(s)\hat{g}^*(s)}{\hat{G}(s)} \mathrm{d}m_n(s)$ and obtain the norm $\|f\|_{\mathcal{H}}^2 = \int_{\mathbb{R}^d} \frac{|\hat{f}(s)|^2}{\hat{G}(s)} \mathrm{d}m_n(s)$, for $\mathcal{H} = \mathrm{comp\,span}\{G^\dagger(x, .), x \in \mathbb{R}^d\}$, where $\mathrm{comp}\{\ldots\}$ denotes completion of the set $\{\ldots\}$ and a^* means complex conjugate of a. It is easy to check the reproducing property of G on $\mathcal{H}$, that is $\langle f(x), G(x - y)\rangle_{\mathcal{H}} = f(y)$.

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 $\|f\|_k^2 = \int \frac{|\hat{f}|^2}{(1+s^2)^{-1}} = \|f\|_{\mathcal{L}_2}^2 + \|f'\|_{\mathcal{L}_2}^2$. So we see we obtain a Sobolev space W_2^1.

As a more general example we will consider the product of kernels introduced in section 1. Suppose that apriori knowledge of our data suggests to look for the solution as a member of product of two functional spaces. In one dimension the data may be clustered thus being suitable for approximation via Gaussian kernels. In the other dimension we have only information on smoothness of the data, hence we will use kernel resulting in Sobolev norm. Employing theorem 1.1 we obtain a kernel for the product space of the form: $K((x_1, x_2), (y_1, y_2)) = k_1(x_1, y_1) \cdot k_2(x_2 y_2) = e^{-\|x_1 - y_1\|^2} \cdot e^{-|x_2 - y_2|}$, 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:

$$\mathcal{F}_K(f) = \frac{1}{N}\sum_{i=1}^{N}(f(u_i) - v_i)^2 + \gamma \int_{\mathbb{R}^d} \frac{|\hat{f}(s)|^2}{\hat{k_1}\hat{k_2}(s)} \mathrm{d}m_n(s). \tag{4}$$

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

$$f_0(x_1, x_2) = \sum_{i=1}^{N} c_i e^{-\|x_1 - u_{i,1}\|^2} \cdot e^{-|x_2 - u_{i,2}|}. \tag{5}$$

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

3 Learning algorithm

Now we present a learning algorithm based on the theoretical results from the previous sections. We as-

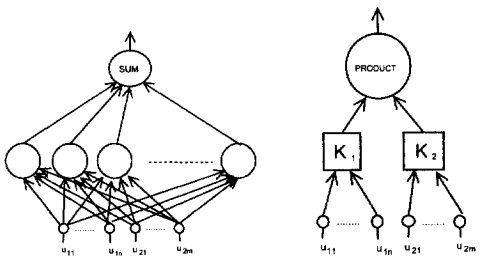

Fig. 1. a) Product Kernel Regularization Network b) Product Unit

sume that we have a data set $\{\vec{u_1}^i, \vec{u_2}^i, v^i\}_{i=1}^N$, where $\vec{u_1}^i \in \mathbb{R}^n$, $\vec{u_2}^i \in \mathbb{R}^m$, $v^i \in \mathbb{R}$ and N is a number of data samples. We will fit these data set using *Product Kernel Regularization Network* (PKRN) derived from the regularization schema 4.

It is a feed-forward neural network with one hidden layer of N *product units* and a linear output layer (see Fig. 1a). By a *product unit* (see Fig. 1b) we mean a unit with $(n + m)$ real inputs and one real output. It consists of two positive definite kernel functions $K_1(\vec{c_1}, \cdot)$, $K_2(\vec{c_2}, \cdot)$, one evaluating the first n inputs and one evaluating the other m inputs, the output of the product unit is computed as the product $K_1(\vec{c_1}, \vec{u_1}) \cdot K_2(\vec{c_2}, \vec{u_2})$.

The network then evaluates the function

$$f(\vec{u_1}, \vec{u_2}) = \sum_{i=1}^{k} w_i K_1(\vec{c_1}^i, \vec{u_1}) \cdot K_2(\vec{c_2}^i, \vec{u_2}), \qquad (6)$$

where the parameters $\vec{c_1}^i$ and $\vec{c_2}^i$ are called *centers* and the coefficients of the linear combination w_i *weights*.

The learning algorithm for PKRN is sketched at Fig. 2. It is derived from Tikhonov regularization and for the case of Regularization Network was described in [8]. See also [6].

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. Parameter γ must be estimated in advance (cross-validation is usually used).

4 Experiments

We tested the performance of proposed Product Kernel Regularization Network on several experiments, including both benchmark and real life problems.

We always use two disjunct data sets, one for training and one for evaluating the error of the result, and compute the normalized error:

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

1. Set the centers of kernels:

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

2. Compute the values of weigths $w_1, \ldots, w_k$:

$$(k\gamma I + K)\vec{w} = \vec{v},$$

where I is the identity matrix,
$K_{i,j} = K_1(\vec{c_1}^i, \vec{u_1}^j) \cdot K_2(\vec{c_2}^i, \vec{u_2}^j)$,
and $\vec{v} = (v_1, \ldots, v_k)$, $\gamma > 0$.

Fig. 2. Learning algorithm for Product Kernel Regularization Network.

Table 1. Error values for PKRN and RN on Proben1 data sets.

	PKRN		**RN**	
	E_{train}	E_{test}	E_{train}	E_{test}
cancer1	2.739	**1.816**	2.658	1.875
cancer2	2.152	3.516	2.279	**3.199**
cancer3	2.374	**2.798**	2.348	2.873
glass1	6.141	8.590	4.899	**8.033**
glass2	5.269	**8.202**	4.570	8.317
glass3	3.691	**7.411**	4.837	7.691

$$E = 100\frac{1}{N}\sum_{i=1}^{N} ||v^i - f(\vec{u_1}^i, \vec{u_2}^i)||^2, \qquad (7)$$

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

We have used the Gaussian function $e^{-\left(\frac{\|c-x\|}{b}\right)^2}$ for both kernel functions (K_1 and K_2), but the kernels differ in the width b of the Gaussian functions. All parameters γ, b_1 and b_2 were estimated by cross-validation. LAPACK library [7] was used for linear system solving.

The table 1 compares the resulting errors of PKRN and Regularization Networks (RN) on data sets selected from Proben1 [9] benchmark repository.

The applicability of PKRN on real life problems is demonstrated on the prediction of the flow rate on the Czech river Ploucnice. 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 $f: \mathbb{R} \times \mathbb{R} \to \mathbb{R}$.

We have three different data sets for this task – called pl1, pl1s and pl2, each containing 1000 training samples

and 367 testing samples. The results obtained on these data sets by PKRN are listed in the table 2.

Table 2. Error values of PKRN on training and testing sets of the Ploucnice data.

	pl1	pl1s	pl2
E_{train}	0.057180	0.057215	0.109477
E_{test}	0.048332	0.048475	0.097608

The table 3 shows that the PKRN overperformes the so called *conservative predictor*. Conservative prediction is a predictor saying that the value will be the same as it was yesterday, and in spite of its simplicity it is very successful on some tasks, including this one.

The prediction on the testing set made by PKRN is displayed at Fig. 3.

Table 3. Comparison of errors obtained by PKRN and conservative predictor (CP).

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

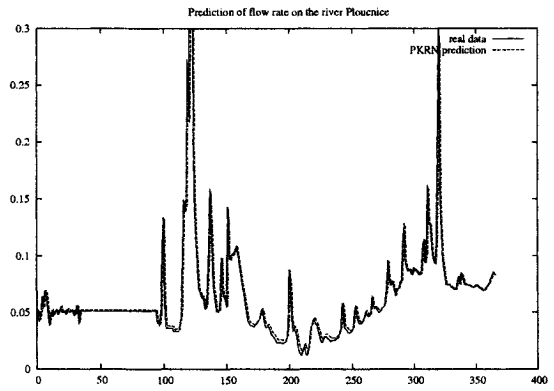

Fig. 3. Prediction of the flow rate on the river Ploucnice by Product Kernel Regularization Network.

5 Conclusion

We have shown how to employ RKHS in approximation theory and stressed advantages of this approach. Inspired by the article [1] we introduce kernel-product based approximation and derive the shape of Product Kernel Regularization Networks (PKRN).

We tested the performance of proposed PKRN on benchmark tasks from Proben1 repository and showed that its result are comparable to standard variant of Regularization Network. We demonstrated the applicability of PKRN on prediction of river flow rate, which is a real-life task.

We expect our algorithm to be useful particularly in situations where some prior knowledge of the character of data is available in the sense that we can expect that for some groups of inputs different kernel functions are suitable.

References

[1] Aronszajn N. (1950) *Theory of Reproducing Kernels*. Transactions of the AMS, **68**, 3, pp. 337-404.

[2] Cucker F., Smale S. (2001). *On the Mathematical Foundations of Learning*. Bulletin of the American Mathematical Society **39**, 1-49.

[3] Daniel J. W. (1971). *The Approximate Minimization of Functionals*. Prentice-Hall, Inc.

[4] Girossi F. (1998). *An Equivalence between Sparse Approximation and Support Vector Machines*. Neural Computation **10**, 1455-1480, MIT. (A.I. Memo No. 1606, MIT, 1997)

[5] Girossi F., Jones M., Poggio T. (1995). *Regularization Theory and Neural Networks Architectures*. Neural Computation, **7**, 219-269.

[6] Kudová P. (2004). *Kernel Based Regularization Networks and RBF Networks*. Doktorandský Den 2004, Paseky nad Jizerou.

[7] Lapack library. `http://www.netlib.org/lapack/`.

[8] Poggio T., Smale S. (2003). *The Mathematics of Learning: Delaing with Data*. Notices of the AMS **50**, 5, 536-544.

[9] Prechelt, L. (1994) *PROBEN1 – A Set of Benchmarks and Benchmarking Rules for Neural Network Training Algorithms*. Universitaet Karlsruhe, 21/94.

[10] Schölkopf B., Smola A. J. (2002). *Learning with Kernels*. MIT Press, Cambridge, Massachusetts.

[11] Šidlofová T. (2004). *Existence and Uniqueness of Minimization Problems with Fourier Based Stabilizers*. Compstat 2004, Prague.

[12] Šidlofová T. (2004). *Kernel Based Regularization and Neural Networks*. Doktorandský Den 2004, Paseky nad Jizerou.

[13] Wahba G. (1990). *Spline Models for Observational Data*. Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia.

This work was supported by GA ČR grant 201/05/0557 and the National Research Program Information Society project T100300414.