Kernel-Based Nonlinear Signal Processing

José C. M. Bermudez, Paul Honeine, Jean-Yves Tourneret, Cédric Richard

Book Chapter
Kernel-Based Nonlinear Signal Processing

J. C. M. Bermudez, P. Honeine, J.-Y. Tourneret and C. Richard

June 24, 2014

Contents

Chapter 1

Kernel-Based Nonlinear Signal Processing

1.1 Chapter Summary

In this chapter we discuss kernel-based nonlinear signal processing. Sections 1.3 and 1.4 present a brief introduction to the theory of reproducing kernel Hilbert spaces (RKHS) and their application to nonparametric modeling of nonlinear functions. We give special emphasis to nonlinear function estimation and regression, and illustrate their relevance on several nonlinear signal processing applications. In Section 1.5 we explore online kernel-based function approximation. We focus on finite order modeling, sparsification techniques and their application to online learning. In Section 1.6 we discuss kernel-based online system identification, with emphasis on the popular kernel least mean squares algorithm (KLMS) and its properties. In Section 1.7 we present an introduction to Gaussian processes as applied to nonlinear regression, and we describe its application to spectral unmixing of hyperspectral images.

1.2 Introduction

An effective way to extend the scope of linear models to nonlinear processing is to map the input data $\boldsymbol{u}_i$ into a high-dimensional space using a nonlinear function $\boldsymbol{\varphi}(\cdot)$, and apply linear modeling techniques to the transformed data $\boldsymbol{\varphi}(\boldsymbol{u}_i)$. However, this strategy may fail when the image of $\boldsymbol{\varphi}(\cdot)$ lies in a very high, or even infinite, dimensional space. More recently, kernel-based methods have been proposed for applications in classification and regression problems. These methods exploit the central idea of this research area, known as the *kernel trick*, to evaluate inner products in the high dimensional space without the knowledge of the function $\boldsymbol{\varphi}(\cdot)$. Well-known examples can be found in [1,2]. This chapter discusses the application of kernel-based methods to solve nonlinear function estimation and regression problems.

1.3 Reproducing Kernel Hilbert Spaces (RKHS)

This section briefly reviews the main definitions and properties related to reproducing kernel Hilbert spaces [3] and Mercer kernels [4].

Let $\mathcal{H}$ denote a Hilbert space of real-valued functions $\psi(\cdot)$ on a compact $\mathcal{U} \subset \mathbb{R}^{\ell}$, and let $\langle \cdot\,, \cdot \rangle_{\mathcal{H}}$ be the inner product in $\mathcal{H}$. Suppose that the evaluation functional L_u defined by $L_u[\psi] \triangleq \psi(\boldsymbol{u})$ is linear with respect to $\psi(\cdot)$ and bounded, for all $\boldsymbol{u}$ in $\mathcal{U}$. By virtue of the Riesz representation theorem, there exists a unique positive definite function $\boldsymbol{u}_i \mapsto \kappa(\boldsymbol{u}_i, \boldsymbol{u}_j)$ in $\mathcal{H}$, denoted by $\kappa(\cdot, \boldsymbol{u}_j)$ and called *representer of evaluation* at $\boldsymbol{u}_j$, that satisfies [3]

$$\psi(\boldsymbol{u}_j) = \langle \psi(\cdot), \kappa(\cdot, \boldsymbol{u}_j) \rangle_{\mathcal{H}}, \quad \forall \psi \in \mathcal{H} \tag{1.1}$$

for every fixed $\boldsymbol{u}_j \in \mathcal{U}$. A proof of this may be found in [3]. Replacing $\psi(\cdot)$ by $\kappa(\cdot, \boldsymbol{u}_i)$ in (1.1) yields

$$\kappa(\boldsymbol{u}_j, \boldsymbol{u}_i) = \langle \kappa(\cdot, \boldsymbol{u}_i), \kappa(\cdot, \boldsymbol{u}_j) \rangle_{\mathcal{H}} \tag{1.2}$$

for all $\boldsymbol{u}_i,\ \boldsymbol{u}_j \in \mathcal{U}$. Equation (1.2) is the origin of the now generic term *reproducing kernel* to refer to $\kappa(\cdot\,, \cdot)$, which is also known as Mercer kernel. Note that $\mathcal{H}$ can be restricted to the span of $\{\kappa(\cdot, \boldsymbol{u}) : \boldsymbol{u} \in \mathcal{U}\}$ because, according to (1.1), nothing outside this set affects $\psi(\cdot)$ evaluated at any point of $\mathcal{U}$. Denoting by $\boldsymbol{\varphi}(\cdot)$ the map that assigns to each input $\boldsymbol{u}$ the kernel function $\kappa(\cdot, \boldsymbol{u})$, (1.2) implies that $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = \langle \boldsymbol{\varphi}(\boldsymbol{u}_i), \boldsymbol{\varphi}(\boldsymbol{u}_j) \rangle_{\mathcal{H}}$. The

kernel then evaluates the inner product of any pair of elements of $\mathcal{U}$ mapped to $\mathcal{H}$ without any explicit knowledge of either $\boldsymbol{\varphi}(\cdot)$ or $\mathcal{H}$. This key idea is known as the *kernel trick*. The kernel trick has been widely used to transform linear algorithms expressed only in terms of inner products into nonlinear ones. Examples are the nonlinear extensions to the principal component analysis [5] and the Fisher discriminant analysis [6,7]. Recent work has also been focused on kernel-based online prediction of time series [8–12].

Classic examples of kernels are the radially Gaussian kernel $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = \exp\bigl(-\|\boldsymbol{u}_i - \boldsymbol{u}_j\|^2/2\beta_0^2\bigr)$, with $\beta_0 \geqslant 0$ the kernel bandwidth, and the Laplacian kernel $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = \exp(-\|\boldsymbol{u}_i - \boldsymbol{u}_j\|/\beta_0)$. Another example which deserves attention in signal processing is the q-th degree polynomial kernel defined as $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = (\eta_0 + \boldsymbol{u}_i^t\, \boldsymbol{u}_j)^q$, with $\eta_0 \geqslant 0$ and $q \in \mathbb{N}^*$. The nonlinear function $\boldsymbol{\varphi}(\cdot)$ related to the latter transforms every observation $\boldsymbol{u}_i$ into a vector $\boldsymbol{\varphi}(\boldsymbol{u}_i)$, in which each component is proportional to a monomial of the form $(u_{i,1})^{k_1}(u_{i,2})^{k_2} \ldots (u_{i,p})^{k_p}$ for every set of exponents $(k_1, ..., k_p) \in \mathbb{N}^p$ satisfying $0 \leqslant \sum_{r=1}^{p} k_r \leqslant q$. For details, see [13, 14] and references therein. The models of interest then correspond to q-th degree Volterra series representations.

1.4 Nonlinear Regression in an RKHS

The regression problem is an example of learning in which we want to predict the values of one or more continuous variables (*outputs*) as a function of a set of *inputs* that are measured or preset. Problems in which a collection of known input-output pairs is available for training the regression model are known as supervised learning problems. The regression is nonlinear when the function to be modeled is nonlinear.

A nonlinear regression problem with a scalar output variable is characterized by the expression

$$d_i = \psi(\boldsymbol{u}_i) + \eta_i, \qquad i = 1, \ldots, n \tag{1.3}$$

where d_i is the i-th output, $\boldsymbol{u}_i$ is the i-th input vector and η_i is the so-called noise sample, which in general represents the part of d_i that can not be modeled by $\psi(\cdot)$. Noise η_i is frequently modeled as a sample of a white Gaussian process.

Sometimes the mathematical form of $\psi(\cdot)$ is known or can be reasonably

inferred from practical or theoretical considerations, except for some parameters. In these cases, the objective is to estimate the unknown parameter values as precisely as possible [15–17]. However, in several practical problems the underlying process is complex and not well understood. In such cases, one possible approach is to use a family of functions that is flexible enough to approximate a sufficiently large variety of functional forms. The objective is then to estimate the parameters of the family of functions. Examples of popular families of functions are polynomial models and kernel expansions in RKHS. Another approach that has recently gained popularity in the signal processing community is to avoid the choice of specific functional models and imagine that the output samples are samples from a multivariate Gaussian distribution. The sequence of samples is modeled as a Gaussian process characterized by a covariance function in RKHS [18]. This chapter reviews recent solutions to the nonlinear regression problem in the RKHS.

To solve the nonlinear regression problem in the RKHS, let $\kappa : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ be a kernel, and let $\mathcal{H}$ be the RKHS associated with it. The problem is to determine a function $\psi(\cdot)$ of $\mathcal{H}$ that minimizes a cost function $\sum_{i=1}^{n} \mathcal{L}(d_i, \psi(\boldsymbol{u}_i))$, where $\mathcal{L}(d_i, \psi(\boldsymbol{u}_i))$ is an appropriate loss. Considering the least-squares approach for instance, the cost function is the sum of n squared errors between samples d_i of the desired response and the corresponding model output samples $\psi(\boldsymbol{u}_i) = \langle \psi(\cdot), \kappa(\cdot, \boldsymbol{u}_i) \rangle_{\mathcal{H}}$, namely,

$$\min_{\psi \in \mathcal{H}} \sum_{i=1}^{n} |d_i - \psi(\boldsymbol{u}_i)|^2. \tag{1.4}$$

While the solution to this problem is not unique, one often considers a regularized cost function

$$\min_{\psi \in \mathcal{H}} \sum_{i=1}^{n} |d_i - \psi(\boldsymbol{u}_i)|^2 + \epsilon \, \|\psi(\cdot)\|_{\mathcal{H}}^2 \tag{1.5}$$

where a regularization term is introduced in order to impose the uniqueness and smoothness of the solution. The tradeoff between training errors and smoothness of the solution is controlled by the tunable non-negative parameter ϵ. Taking the functional derivative of the above cost function with respect to $\psi(\cdot)$ [19] and nullifying it, leads to

$$-\sum_{i=1}^{n} \left(d_i - \psi(\boldsymbol{u}_i)\right) \kappa(\cdot, \boldsymbol{u}_i) + \epsilon \, \psi(\cdot) = 0$$

where we have used the linearity with respect to $\psi(\cdot)$ in $\psi(\boldsymbol{u}_i) = \langle \psi(\cdot), \kappa(\cdot, \boldsymbol{u}_i) \rangle_{\mathcal{H}}$, as well as the fact that $\psi(\cdot)$ is the derivative of the quadratic form $\frac{1}{2}\|\psi(\cdot)\|^2_{\mathcal{H}}$. Therefore, the function $\psi(\cdot)$ of $\mathcal{H}$ minimizing (1.4) can be written as a kernel expansion in terms of available data, namely

$$\psi(\cdot) = \sum_{i=j}^{n} \alpha_j \, \kappa(\cdot, \boldsymbol{u}_j). \tag{1.6}$$

By injecting this expression in the problem (1.4), we obtain the optimization problem $\min_{\boldsymbol{\alpha}} \ \|\boldsymbol{d} - \boldsymbol{K}\boldsymbol{\alpha}\|^2 + \epsilon \, \boldsymbol{\alpha}^\top \boldsymbol{K} \boldsymbol{\alpha}$, where $\boldsymbol{K}$ is the Gram matrix whose (i, j)-th entry is $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j)$ and $\boldsymbol{d}$ is the vector of desired output values whose i-th entry is d_i. By taking the derivative of this cost function with respect to $\boldsymbol{\alpha}$ and nullifying it, we get the $n \times n$ linear system of equations $(\boldsymbol{K}^\top \boldsymbol{K} + \epsilon \, \boldsymbol{K}) \, \boldsymbol{\alpha} = \boldsymbol{K}^\top \boldsymbol{d}$.

It is worth noting that the regularization term given in (1.5) is often dropped in adaptive learning. In this case, the uniqueness and regularity of the solution of (1.4) are controlled by other principles, such as the minimal disturbance principle (see Section 1.6).

1.5 Online Kernel-Based Function Approximation

1.5.1 Introduction

Online solution of the nonlinear regression problem formulated in the RHKS raises the question of how to process an increasing amount of observations and update the model (1.6) as new data is collected. Because the order of the model (1.6) is equal to the number n of available data $\boldsymbol{u_i}$, this approach cannot be considered for online applications. Hence, one needs to look for sparse solutions. One way to obtain a sparse solution is to consider fixed-size models of the form

$$\psi(\cdot) = \sum_{j=1}^{m} \alpha_j \, \kappa(\cdot, \boldsymbol{u}_{\omega_j}) \tag{1.7}$$

where the $\boldsymbol{u}_{\omega_1}, \ldots, \boldsymbol{u}_{\omega_m}$ form an m-element subset of the available input vectors.

Online applications, however, impose time-varying input signals. Hence, it is convenient to adapt the notation of (1.7) by denoting the input vector

at time n by $\boldsymbol{u}_n$, and by defining ω_j, $j = 1, \ldots, m$ to form a subset of $\mathcal{J}_n = \{1, 2, \ldots, n\}$ corresponding to the time indexes of the $m < n$ input vectors chosen to build the m-th order model. Under this notation, (1.7) becomes

$$\psi_n(\cdot) = \sum_{j=1}^{m} \alpha_{j,n}\, \kappa(\cdot, \boldsymbol{u}_{\omega_j}). \tag{1.8}$$

Once the fixed-order model (1.8) is adopted, the next problem is to choose an adequate set of m input vectors $\boldsymbol{u}_{\omega_j}$ to build the model. The m kernel functions $\kappa(\cdot, \boldsymbol{u}_{\omega_j})$ form the *dictionary* $\mathcal{D}$, and the selection of the elements to compose $\mathcal{D}$ must follow some sparsification strategy. In the following we briefly review existing sparsification rules that can be used to this end.

Sparsification Rules

Discarding a kernel function from the model expansion (1.8) may degrade its performance. Sparsification rules aim at identifying kernel functions whose removal is expected to have negligible effect on the quality of the model. An extensive literature addressing this issue in batch and online modes exists, see, e.g., [20] and references therein. In particular, much attention has been recently focused on least-squares support vector machines since they suffer from the loss of sparsity due to the use of a quadratic loss function [21]. In batch mode, this problem was addressed using pruning [22, 23] and fixed-size approaches [21, 24, 25]. Truncation and approximation processes were considered for online scenarios [20].

The most informative sparsification criteria use approximate linear dependence conditions to evaluate whether the contribution of a candidate kernel function can be distributed over the elements of $\mathcal{D}$ by adjusting their multipliers. In [26], determination of the kernel function which is best approximated by the others is carried out by an eigendecomposition of the Gram matrix. This process is not appropriate for online applications, as its complexity at each time step is cubic in the size m of $\mathcal{D}$. In [10], the kernel function $\kappa(\cdot, \boldsymbol{u}_n)$ is inserted at time step n into $\mathcal{D}$ if the following condition is satisfied

$$\min_{\boldsymbol{\gamma}} \left\| \kappa(\cdot, \boldsymbol{u}_n) \;-\; \sum_{\omega_j \in \mathcal{J}_{n-1}} \gamma_j\, \kappa(\cdot, \boldsymbol{u}_{\omega_j}) \right\|_{\mathcal{H}}^2 \geqslant \nu \tag{1.9}$$

where κ is a unit-norm kernel[1], that is, $\kappa(\boldsymbol{u}_k, \boldsymbol{u}_k) = 1$ for all $\boldsymbol{u}_k$. The threshold ν determines the level of sparsity of the model. Note that condition (1.9) ensures the linear independence of the elements of the dictionary. A similar criterion is used in [8, 9], but in a different form. After updating the model parameters, a complementary pruning process is executed in [9] to limit the increase in the model order. It estimates the error induced in $\psi(\cdot)$ at time n by the removal of each kernel and discards those kernels found to have the smallest contribution. A major criticism that can be made of rule (1.9) is that it leads to elaborate and costly operations with quadratic complexity in the cardinality m of $\mathcal{D}$. In [8,9], the model reduction step is computationally more expensive than the parameter update step, the latter being a stochastic gradient descent with linear complexity in m. In [10], the authors focus their study on a parameter update step of the RLS type with quadratic complexity in m. To reduce the overall computational effort, the parameter update and the model reduction steps share intermediate calculation results. This excludes very useful and popular online regression techniques. In [27, 28] a coherence-based sparsification rule has been proposed which requires much less computational complexity than the rules discussed so far. According to this rule, kernel $\kappa(\cdot, \boldsymbol{u}_n)$ is inserted into the dictionary if

$$\max_{\omega_j} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_j})| \leqslant \varepsilon_0 \tag{1.10}$$

with ε_0 a parameter determining the dictionary coherence. It was shown in [28] that the dictionary dimension determined under rule (1.10) is always finite.

The next section presents a set of kernel-based adaptive algorithms and discusses their application to online time-series prediction

1.5.2 Algorithms for Online Time-Series Prediction

Consider the m-th order model at time step n, as given in (1.8), namely

$$\psi_n(\cdot) = \sum_{j=1}^{m} \alpha_{j,n}\, \kappa(\cdot, \boldsymbol{u}_{\omega_j})$$

where the m kernel functions $\kappa(\cdot, \boldsymbol{u}_{\omega_j})$ form a ε_0-coherent dictionary, obtained for instance by the coherence criterion (1.10). By injecting the above

[1]Replace $\kappa(\cdot, \boldsymbol{u}_k)$ with $\kappa(\cdot, \boldsymbol{u}_k)/\sqrt{\kappa(\boldsymbol{u}_k, \boldsymbol{u}_k)}$ in (1.9) if $\kappa(\cdot, \boldsymbol{u}_k)$ is not unit-norm.

model in the problem (1.4), we get the optimization problem $\min_{\boldsymbol{\alpha_n}} \|\boldsymbol{d} - \boldsymbol{H}\boldsymbol{\alpha_n}\|^2$, where $\boldsymbol{H}$ is the $n \times m$ matrix whose (i,j)-th entry is $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_{\omega_j})$. By taking the derivative of this cost function with respect to $\boldsymbol{\alpha_n}$ and nullifying it, we get the optimal solution

$$\boldsymbol{\alpha}_n = \left(\boldsymbol{H}^\top \boldsymbol{H}\right)^{-1} \boldsymbol{H}^\top \boldsymbol{d}$$

assuming $\boldsymbol{H}^\top \boldsymbol{H}$ nonsingular and where $\boldsymbol{\alpha}_n = [\alpha_{1,n}, \ldots, \alpha_{m,n}]^\top$. In the following, we provide adaptive techniques to estimate $\boldsymbol{\alpha}_n$ from its previous estimate, $\boldsymbol{\alpha}_{n-1}$.

We describe in details the Kernel Affine Projection (KAP) algorithm, and then we present several variants. Under the minimal disturbance principle, the estimate is determined at each time step by projecting the previous estimate, subject to solving an underdetermined least-squares problem. The latter is defined only on the p most recent inputs and outputs at each time step n, respectively $\{\boldsymbol{u}_n, \boldsymbol{u}_{n-1}, \ldots, \boldsymbol{u}_{n-p+1}\}$ and $\{d_n, d_{n-1}, \ldots, d_{n-p+1}\}$. In the following, $\boldsymbol{H}_n$ denotes the $p \times m$ matrix whose (i,j)-th entry is $\kappa(\boldsymbol{u}_{n-i+1}, \boldsymbol{u}_{\omega_j})$, namely,

$$\boldsymbol{H}_n = \begin{bmatrix} \boldsymbol{h}_n & \boldsymbol{h}_{n-1} & \ldots & \boldsymbol{h}_{n-p+1} \end{bmatrix}^\top$$

where

$$\boldsymbol{h}_{n-i+1} = \begin{bmatrix} \kappa(\boldsymbol{u}_{n-i+1}, \boldsymbol{u}_{\omega_1}) & \ldots & \kappa(\boldsymbol{u}_{n-i+1}, \boldsymbol{u}_{\omega_m}) \end{bmatrix}^\top, \quad i = 1, \ldots, p$$

and $\boldsymbol{d}_n$ is the column vector whose i-th entry is d_{n-i+1} for any $i = 1, \ldots, p$.

The affine projection problem is defined at time step n with the following constrained optimization problem:

$$\min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha} - \boldsymbol{\alpha}_{n-1}\|^2 \qquad \text{subject to} \qquad \boldsymbol{d}_n = \boldsymbol{H}_n \boldsymbol{\alpha}. \tag{1.11}$$

In other words, $\boldsymbol{\alpha}_n$ is obtained by projecting $\boldsymbol{\alpha}_{n-1}$ onto the intersection of the p manifolds defined, for $i = 1, 2, \ldots, p$, by

$$\left\{\boldsymbol{\alpha} : \boldsymbol{h}_{n-i+1}^\top \boldsymbol{\alpha} - d_{n-i+1} = 0\right\}, \qquad i = 1, \ldots, p.$$

At time step n, upon arrival of a new sample $\boldsymbol{u}_n$, one of the following alternatives holds. If $\kappa(\cdot, \boldsymbol{u}_n)$ violates the coherence-based condition (1.10), the dictionary remains unaltered. On the other hand, if the condition (1.10) is satisfied, $\kappa(\cdot, \boldsymbol{u}_n)$ is inserted into the dictionary where it is denoted by

$\kappa(\cdot, \boldsymbol{u}_{\omega_{m+1}})$. The matrix $\boldsymbol{H}_n$ is augmented by appending to $\boldsymbol{H}_{n-1}$ the column vector $[\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_{m+1}}) \;\; \kappa(\boldsymbol{u}_{n-1}, \boldsymbol{u}_{\omega_{m+1}}) \;\; \ldots \;\; \kappa(\boldsymbol{u}_{n-p+1}, \boldsymbol{u}_{\omega_{m+1}})]^\top$. One more entry is also added to the vector $\boldsymbol{\alpha}_n$. The corresponding updating rules, depending on the coherence-based sparsification rule, are detailed in the following.

First case: $\max\limits_{j=1,\ldots,m} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_j})| > \varepsilon_0$

In this case, $\kappa(\cdot, \boldsymbol{u}_n)$ is not appended to the dictionary, since it can be reasonably well represented by the kernel functions already in the dictionary. The solution to the constrained optimization problem (1.11) is determined by minimizing the Lagrangian function

$$\|\boldsymbol{\alpha} - \boldsymbol{\alpha}_{n-1}\|^2 + \boldsymbol{\lambda}^\top (\boldsymbol{d}_n - \boldsymbol{H}_n \boldsymbol{\alpha}) \tag{1.12}$$

where $\boldsymbol{\lambda}$ denotes the vector of Lagrange multipliers. Differentiating this expression with respect to $\boldsymbol{\alpha}$, setting it to zero and solving for $\boldsymbol{\alpha} = \boldsymbol{\alpha_n}$, and making $\boldsymbol{d}_n - \boldsymbol{H}_n \boldsymbol{\alpha_n} = \boldsymbol{0}$ to determine $\boldsymbol{\lambda}$, we get the following equations:

$$2\,(\boldsymbol{\alpha}_n - \boldsymbol{\alpha}_{n-1}) = \boldsymbol{H}_n^\top \boldsymbol{\lambda} \quad \text{and} \quad \boldsymbol{\lambda} = 2\,(\boldsymbol{H}_n \boldsymbol{H}_n^\top)^{-1} (\boldsymbol{d}_n - \boldsymbol{H}_n \boldsymbol{\alpha}_{n-1}). \tag{1.13}$$

In order to ensure the nonsingularity of the matrix to be inverted in (1.13), one may use the regularized version, with $(\boldsymbol{H}_n \boldsymbol{H}_n^\top + \epsilon\, \mathbf{I})$. Solving (1.13) for $\boldsymbol{\alpha}_n$, we obtain the following recursive update rule:

$$\boldsymbol{\alpha}_n = \boldsymbol{\alpha}_{n-1} + \eta\, \boldsymbol{H}_n^\top (\boldsymbol{H}_n \boldsymbol{H}_n^\top + \epsilon\, \mathbf{I})^{-1} (\boldsymbol{d}_n - \boldsymbol{H}_n \boldsymbol{\alpha}_{n-1}) \tag{1.14}$$

where we have introduced the step-size control parameter η. This update rule requires inverting the usually small $p \times p$ matrix $(\boldsymbol{H}_n \boldsymbol{H}_n^\top + \epsilon \boldsymbol{I})$.

Second case: $\max\limits_{j=1,\ldots,m} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_j})| \leqslant \varepsilon_0$

In this case, the kernel function $\kappa(\cdot, \boldsymbol{u}_n)$ is included in the dictionary since it cannot be efficiently represented by the elements already in the dictionary. Hence, it is denoted by $\kappa(\cdot, \boldsymbol{u}_{\omega_{m+1}})$. The model order m in (1.7) is increased by one, and $\boldsymbol{H}_n$ is updated to a $p \times (m+1)$ matrix. To accommodate the new entry α_{m+1} in $\boldsymbol{\alpha}_n$, the optimization problem (1.11) is rewritten as

$$\min_{\boldsymbol{\alpha}} \; \|\boldsymbol{\alpha}_{1,\ldots,m} - \boldsymbol{\alpha}_{n-1}\|^2 + \alpha_{m+1}^2 \qquad \text{subject to} \quad \boldsymbol{d}_n = \boldsymbol{H}_n \boldsymbol{\alpha} \tag{1.15}$$

where $\boldsymbol{\alpha}_{1,\ldots,m}$ denotes the first m entries of the vector $\boldsymbol{\alpha}$ and $\boldsymbol{H}_n$ has been increased by one column as described before. Note that the $(m+1)$-th entry

of $\boldsymbol{\alpha}$, namely α_{m+1}, acts as a regularizing term in the objective function. By following the same steps as in the derivation of (1.14), we get the following update rule

$$\boldsymbol{\alpha}_n = \begin{bmatrix} \boldsymbol{\alpha}_{n-1} \\ 0 \end{bmatrix} + \eta\, \boldsymbol{H}_n^\top (\boldsymbol{H}_n \boldsymbol{H}_n^\top + \epsilon \boldsymbol{I})^{-1} \left(\boldsymbol{d}_n - \boldsymbol{H}_n \begin{bmatrix} \boldsymbol{\alpha}_{n-1} \\ 0 \end{bmatrix} \right). \tag{1.16}$$

Next, we investigate the instantaneous approximations for the gradient vectors, in order to derive the KNLMS and the KLMS algorithms.

Instantaneous approximations — The KNLMS algorithm

By considering an instantaneous approximation with $p = 1$, the affine projection algorithm described above leads to the kernel normalized least mean square (KNLMS) algorithm. At each time step n, the algorithm described above enforces a null *a posteriori* error, namely $d_n = \boldsymbol{h}_n^\top \boldsymbol{\alpha}_n$, where $\boldsymbol{h}_n$ is the column vector whose i-th entry is $\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_i})$. The update rules (1.14) and (1.16) reduce to

1. If $\max_j |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_j})| > \varepsilon_0$: Let $\boldsymbol{h}_n = [\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_1}) \ldots \kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})]^\top$, then

$$\boldsymbol{\alpha}_n = \boldsymbol{\alpha}_{n-1} + \frac{\eta}{\epsilon + \|\boldsymbol{h}_n\|^2} \left(d_n - \boldsymbol{h}_n^\top \boldsymbol{\alpha}_{n-1}\right) \boldsymbol{h}_n. \tag{1.17}$$

2. If $\max_j |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_j})| \leqslant \varepsilon_0$: Let $\boldsymbol{h}_n = [\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_1}) \ldots \kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_{m+1}})]^\top$, then

$$\boldsymbol{\alpha}_n = \begin{bmatrix} \boldsymbol{\alpha}_{n-1} \\ 0 \end{bmatrix} + \frac{\eta}{\epsilon + \|\boldsymbol{h}_n\|^2} \left(d_n - \boldsymbol{h}_n^\top \begin{bmatrix} \boldsymbol{\alpha}_{n-1} \\ 0 \end{bmatrix} \right) \boldsymbol{h}_n. \tag{1.18}$$

Instantaneous approximations — The KLMS algorithm

One of the easiest ways to derive an adaptive rule is to consider the stochastic gradient of (1.4), which leads to the so-called kernel least mean square (KLMS) algorithm. Since the *a priori* error at time step n is $d_n = \boldsymbol{h}_n^\top \boldsymbol{\alpha}_{n-1}$, we get the same steps as in the KNLMS algorithm without the step size normalization, and the update rules (1.17) and (1.18) become, respectively,

$$\boldsymbol{\alpha}_n = \boldsymbol{\alpha}_{n-1} + \eta \left(d_n - \boldsymbol{h}_n^\top \boldsymbol{\alpha}_{n-1}\right) \boldsymbol{h}_n$$

and

$$\boldsymbol{\alpha}_n = \begin{bmatrix} \boldsymbol{\alpha}_{n-1} \\ 0 \end{bmatrix} + \eta \left(d_n - \boldsymbol{h}_n^\top \begin{bmatrix} \boldsymbol{\alpha}_{n-1} \\ 0 \end{bmatrix} \right) \boldsymbol{h}_n.$$

1.5.3 Example 1

Consider the "Henon map" defined by the nonlinear system

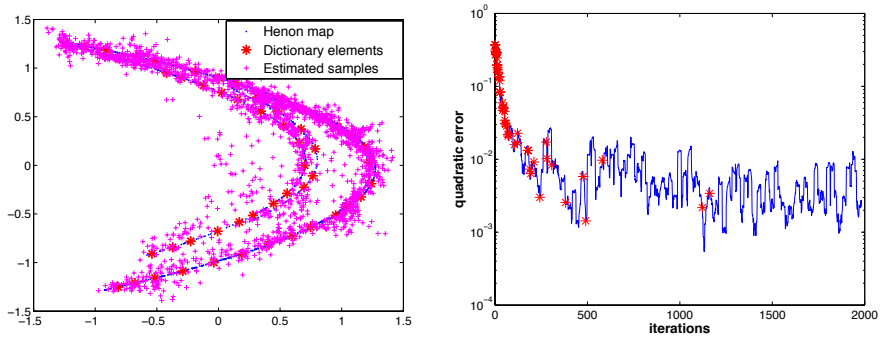
$$\begin{cases} d_n = 1 - \gamma_1 \, d_{n-1}^2 + \gamma_2 \, d_{n-2}; \\ d_0 = -0.3; \\ d_1 = 0. \end{cases}$$

For $\gamma_1 = 1.4$ and $\gamma_2 = 0.3$, the corresponding time-series has a chaotic behavior. The investigated model takes the form $d_n = \psi(d_{n-1}, d_{n-2})$. The length of the time series was 2000 samples. We considered the Gaussian kernel, and the value of its bandwidth was set to $\beta_0 = 0.35$. The choice of the threshold $\varepsilon_0 = 0.6$ led to a dictionary with 52 entries. The relevance of the proposed online kernel-based function approximation is illustrated in Fig. 1.1. Figure 1.1(a) shows the evolution of the pairs (d_n, d_{n-1}) with n corresponding to the actual Henon map, and those predicted using the kernel-based nonlinear model with coefficients $\boldsymbol{\alpha}_n$ adapted using the KAP algorithm with $\eta = 1$. Figure 1.1(b) shows the evolution of the quadratic error.

1.6 Online Nonlinear System Identification

The adaptive algorithms presented above can be employed in any application requiring the estimation of parameters associated with a time series. One particularly important application is nonlinear system identification. The block diagram of a kernel-based adaptive system identification problem is shown in Fig. 1.2. Here, $\mathcal{U}$ is a compact subspace of $\mathbb{R}^\ell$, $\kappa \colon \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ is a reproducing kernel, $(\mathcal{H}, \langle \cdot, \cdot \rangle_{\mathcal{H}})$ is the induced RKHS with its inner product and z_n is a zero-mean additive noise uncorrelated with any other signal.

It is well known that a nonlinear adaptive filtering problem with input signal in $\mathcal{U}$ can be solved using a linear adaptive filter [29]. The linear adaptive filter input is a nonlinear mapping of $\mathcal{U}$ to an Hilbert space $\mathcal{H}$ possessing



(a) Evolution of the estimated map. (b) Evolution of the quadratic error.

Figure 1.1: Illustration of the Henon map in two-dimensions (d_n, d_{n-1}), and the evolution of the quadratic error. Elements of the dictionary are illustrated by red stars.

a reproducing kernel. As discussed in Section 1.5.1, the order of the linear adaptive filter can be finite if a proper input sparsification rule is employed [28], even if the dimensionality of the transformed input in $\mathcal{H}$ is infinite as in the case of a Gaussian kernel.

A proper design of an adaptive algorithm to solve the nonlinear system identification problem depicted in Fig. 1.2, however, requires a good understanding of the algorithm properties in such an application. However, very few analyses related to the behavior of kernel-based adaptive algorithms are available in the literature. A statistical analysis of the behavior of the KLMS adaptive algorithm using a fixed order dictionary $\mathcal{D}$ and a Gaussian kernel for system identification has been presented in [30]. Other aspects of the KLMS behavior have been studied in [30–33]. In the analysis, the environment is assumed stationary, meaning that $\psi(\boldsymbol{u}_n)$ is stationary for $\boldsymbol{u}_n$ stationary. This assumption is satisfied by several nonlinear systems used to model practical situations, such as memoryless, Wiener and Hammerstein systems. System inputs are assumed to be zero-mean, independent and identically distributed Gaussian $(\ell \times 1)$ vectors $\boldsymbol{u}_n$ so that $E\{\boldsymbol{u}_{n-i}\, \boldsymbol{u}_{n-j}^\top\} = \boldsymbol{0}$ for $i \neq j$. The components of the input vector $\boldsymbol{u}_n$ can, however, be correlated. Let $\boldsymbol{R}_{uu} = E\{\boldsymbol{u}_n\, \boldsymbol{u}_n^\top\}$ denote their autocorrelation matrix.

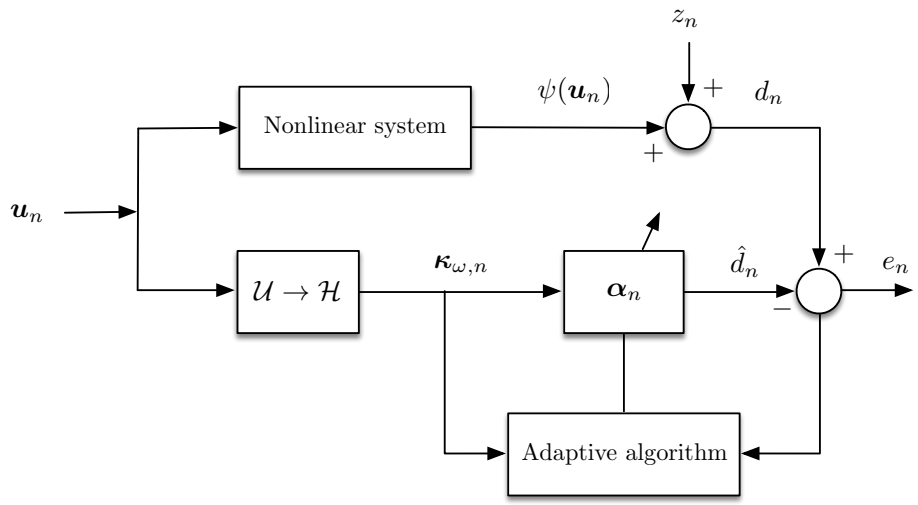


Figure 1.2: Kernel-based adaptive system identification.

For a dictionary of size m, let $\boldsymbol{\kappa}_{\omega,n}$ be the vector of kernels at time $n > m$,[2] that is,

$$\boldsymbol{\kappa}_{\omega,n} = [\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_1}), \ldots, \kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})]^\top \tag{1.19}$$

where $\boldsymbol{u}_{\omega_i}$ is the i-th element of the dictionary, with $\boldsymbol{u}_{\omega_i} \neq \boldsymbol{u}_n$ for $i = 1, \ldots, m$. Here we consider that the vectors $\boldsymbol{u}_{\omega_i}$, $i = 1, \ldots, m$ may change at each iteration following some dictionary updating schedule. The only limitation imposed in the following analysis is that $\boldsymbol{u}_{\omega_i,n} \neq \boldsymbol{u}_{\omega_j,n}$ for $i \neq j$ so that the dictionary vectors which are arguments of different entries of $\boldsymbol{\kappa}_{\omega,n}$ are statistically independent. Note that this framework does not allow the user to pre-tune the dictionary to improve performance. In [34] we proposed a theoretical analysis of the KLMS algorithm with Gaussian kernel that considers the dictionary as part of the filter parameters to be set.

To keep the notation simple, however, we will not show explicitly the dependence of ω_i on n and represent $\boldsymbol{u}_{\omega_i,n}$ as $\boldsymbol{u}_{\omega_i}$ for all i.

From Fig. 1.2 and model (1.8), the estimated system output is

$$\hat{d}(n) = \boldsymbol{\alpha}_n^\top \boldsymbol{\kappa}_{\omega,n}. \tag{1.20}$$

[2]If the dictionary size m is adapted online, we assume that n is sufficiently large so that the size m does not increase anymore.

The corresponding estimation error is defined as

$$e_n = d_n - \hat{d}_n. \tag{1.21}$$

Squaring both sides of (1.21) and taking the expected value leads to the MSE

$$J_{\text{mse},n} = E\{e_n^2\} = E\{d_n^2\} - 2\boldsymbol{p}_{\kappa d}^\top \boldsymbol{\alpha}_n + \boldsymbol{\alpha}_n^\top \boldsymbol{R}_{\kappa\kappa} \boldsymbol{\alpha}_n \tag{1.22}$$

where $\boldsymbol{R}_{\kappa\kappa} = E\{\boldsymbol{\kappa}_{\omega,n} \boldsymbol{\kappa}_{\omega,n}^\top\}$ is the correlation matrix of the kernelized input, and $\boldsymbol{p}_{\kappa d} = E\{d_n \boldsymbol{\kappa}_{\omega,n}\}$ is the cross-correlation vector between $\boldsymbol{\kappa}_{\omega,n}$ and d_n. It is shown in [30] that $\boldsymbol{R}_{\kappa\kappa}$ is positive definite. Thus, the optimum weight vector is given by

$$\boldsymbol{\alpha}_{\text{opt}} = \boldsymbol{R}_{\kappa\kappa}^{-1} \boldsymbol{p}_{\kappa d}. \tag{1.23}$$

The optimal estimation error is

$$e_{0,n} = d_n - \boldsymbol{\kappa}_{\omega,n}^\top \boldsymbol{\alpha}_{\mathbf{opt}} \tag{1.24}$$

and the corresponding minimum MSE is

$$J_{\min} = E\{d_n^2\} - \boldsymbol{p}_{\kappa d}^\top \boldsymbol{R}_{\kappa\kappa}^{-1} \boldsymbol{p}_{\kappa d}. \tag{1.25}$$

These are the well-known expressions of the Wiener solution and minimum MSE, where the input signal vector has been replaced by the kernelized input vector. Determining the optimum $\boldsymbol{\alpha}_{\text{opt}}$ requires the determination of the covariance matrix $\boldsymbol{R}_{\kappa\kappa}$, given the statistical properties of $\boldsymbol{u}_n$ and the reproducing kernel.

To evaluate $\boldsymbol{R}_{\kappa\kappa}$, we note that its entries are given by

$$[\boldsymbol{R}_{\kappa\kappa}]_{ij} = \begin{cases} E\{\kappa^2(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_i})\}, & i = j \\ E\{\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_i})\, \kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_j})\}, & i \neq j \end{cases} \tag{1.26}$$

with $1 \leqslant i, j \leqslant m$. Note that $\boldsymbol{R}_{\kappa\kappa}$ remains time-invariant even if the dictionary is updated at each iteration, as $\boldsymbol{u}_n$ is stationary and $\boldsymbol{u}_{\omega_i}$ and $\boldsymbol{u}_{\omega_j}$ are statistically independent for $i \neq j$. Also, we assume in the following the use of the Gaussian kernel $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = \exp\bigl(-\|\boldsymbol{u}_i - \boldsymbol{u}_j\|^2 / 2\beta_0^2\bigr)$.

Let us introduce the following notations

$$\begin{aligned} &\|\boldsymbol{u}_n - \boldsymbol{u}_{\omega_i}\|^2 = \boldsymbol{y}_2^\top \boldsymbol{Q}_2 \boldsymbol{y}_2 \\ &\|\boldsymbol{u}_n - \boldsymbol{u}_{\omega_i}\|^2 + \|\boldsymbol{u}_n - \boldsymbol{u}_{\omega_j}\|^2 = \boldsymbol{y}_3^\top \boldsymbol{Q}_3 \boldsymbol{y}_3, \quad i \neq j \end{aligned} \tag{1.27}$$

where $\|\cdot\|$ is the ℓ_2 norm and

$$\begin{aligned} \boldsymbol{y}_2 &= \left(\boldsymbol{u}_n^\top \boldsymbol{u}_{\omega_i}^\top\right)^\top \\ \boldsymbol{y}_3 &= \left(\boldsymbol{u}_n^\top \boldsymbol{u}_{\omega_i}^\top \boldsymbol{u}_{\omega_j}^\top\right)^\top \end{aligned} \tag{1.28}$$

and

$$\boldsymbol{Q}_2 = \begin{pmatrix} \boldsymbol{I} & -\boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{I} \end{pmatrix} \qquad \boldsymbol{Q}_3 = \begin{pmatrix} 2\boldsymbol{I} & -\boldsymbol{I} & -\boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{I} & \boldsymbol{O} \\ -\boldsymbol{I} & \boldsymbol{O} & \boldsymbol{I} \end{pmatrix} \tag{1.29}$$

where $\boldsymbol{I}$ is the $(\ell \times \ell)$ identity matrix and $\boldsymbol{O}$ is the $(\ell \times \ell)$ null matrix. From [35, p. 100], we know that the moment generating function of a quadratic form $\xi = \boldsymbol{y}^\top \boldsymbol{Q}\, \boldsymbol{y}$, where $\boldsymbol{y}$ is a zero-mean Gaussian vector with covariance matrix $\boldsymbol{R}_y$, is given by

$$\psi_\xi(s) = E\{e^{s\xi}\} = \det\{\boldsymbol{I} - 2\, s\, \boldsymbol{Q}\, \boldsymbol{R}_y\}^{-1/2}. \tag{1.30}$$

Making $s = -1/(2\beta_0^2)$ in (1.30), we find that the (i,j)-th element of $\boldsymbol{R}_{\kappa\kappa}$ is given by

$$[\boldsymbol{R}_{\kappa\kappa}]_{ij} = \begin{cases} \boldsymbol{r}_{\rm md} = \det\left\{\boldsymbol{I}_2 + 2\, \boldsymbol{Q}_2\, \boldsymbol{R}_2/\beta_0^2\right\}^{-1/2}, & i = j \\ \boldsymbol{r}_{\rm od} = \det\left\{\boldsymbol{I}_3 + \boldsymbol{Q}_3\, \boldsymbol{R}_3/\beta_0^2\right\}^{-1/2}, & i \neq j \end{cases} \tag{1.31}$$

with $1 \leqslant i,\, j \leqslant M$. The main diagonal entries $[\boldsymbol{R}_{\kappa\kappa}]_{ii}$ are all equal to $\boldsymbol{r}_{\rm md}$ and the off-diagonal entries $[\boldsymbol{R}_{\kappa\kappa}]_{ij}$ are all equal to $\boldsymbol{r}_{\rm od}$ because $\boldsymbol{u}_{\omega_i}$ and $\boldsymbol{u}_{\omega_j}$ are i.i.d. In (1.31), $\boldsymbol{R}_q$ is the $(q\ell \times q\ell)$ correlation matrix of vector $\boldsymbol{y}_q$, $\boldsymbol{I}_q$ is the $(q\ell \times q\ell)$ identity matrix, and $\det\{\cdot\}$ denotes the determinant of a matrix. Finally, note that matrix $\boldsymbol{R}_q$ is block-diagonal with $\boldsymbol{R}_{uu}$ along its diagonal.

The analysis in [30] uses the following statistical assumptions for feasibility:

A1: $\boldsymbol{\kappa}_{\omega,n}\boldsymbol{\kappa}_{\omega,n}^\top$ is statistically independent of $\boldsymbol{v}_n$. This assumption is justified in detail in [36] and has been successfully employed in several adaptive filter analyses. It is called here for further reference "modified independence assumption" (MIA). This assumption has been shown in [36] to be less restrictive than the classical independence assumption [37].

A2: The finite-order model provides a close enough approximation to the infinite-order model with minimum MSE, so that $E[e_{0,n}] \approx 0$.

A3: $e_{0,n}$ and $\boldsymbol{\kappa}_{\omega,n}\,\boldsymbol{\kappa}_{\omega,n}^{\top}$ are uncorrelated. This assumption is also supported by the arguments supporting the MIA (**A1**) [36].

The following are the main results of the analysis presented in [30]. The reader is directed to [30] for more details. Defining the weight error vector $\boldsymbol{v}_n = \boldsymbol{\alpha}_n - \boldsymbol{\alpha}_{\mathbf{opt}}$ leads to the KLMS weight-error vector update equation

$$\boldsymbol{v}_{n+1} = \boldsymbol{v}_n + \eta\, e_n\, \boldsymbol{\kappa}_{\omega,n} \tag{1.32}$$

and the estimation error can be written as

$$e_n = d_n - \boldsymbol{\kappa}_{\omega,n}^{\top}\boldsymbol{v}_n - \boldsymbol{\kappa}_{\omega,n}^{\top}\boldsymbol{\alpha}_{\mathrm{opt}}. \tag{1.33}$$

The analysis in [30] shows that the mean behavior of the adaptive weights is given by

$$E\{\boldsymbol{v}_{n+1}\} = (\boldsymbol{I} - \eta\, \boldsymbol{R}_{\kappa\kappa})\, E\{\boldsymbol{v}_n\}. \tag{1.34}$$

For the analysis of the MSE behavior, we note that using (1.33) and the MIA (**A1**), the second-order moments of the weights are related to the MSE through [37]

$$J_{\mathrm{ms}}(n) = J_{\min} + \mathrm{trace}\{\boldsymbol{R}_{\kappa\kappa}\, \boldsymbol{C}_{v,n}\} \tag{1.35}$$

where $\boldsymbol{C}_{v,n} = E\{\boldsymbol{v}_n\, \boldsymbol{v}_n^{\top}\}$ is the autocorrelation matrix of $\boldsymbol{v}_n$ and $J_{\min} = E\{e_{0,n}^2\}$ is the minimum MSE. The study of the MSE behavior (1.35) requires a model for $\boldsymbol{C}_{v,n}$. This model is highly affected by the transformation imposed on the input signal $\boldsymbol{u}_n$ by the kernel. An analytical model for the behavior of $\boldsymbol{C}_{v,n}$ is derived in [30] and is given by

$$\boldsymbol{C}_{v,n+1} \approx \boldsymbol{C}_{v,n} - \eta\,(\boldsymbol{R}_{\kappa\kappa}\, \boldsymbol{C}_{v,n} + \boldsymbol{C}_{v,n}\, \boldsymbol{R}_{\kappa\kappa}) + \eta^2\, \boldsymbol{T}_n + \eta^2\, \boldsymbol{R}_{\kappa\kappa}\, J_{\min} \tag{1.36a}$$

with

$$\boldsymbol{T}_n = E\{\boldsymbol{\kappa}_n\, \boldsymbol{\kappa}_n^{\top}\, \boldsymbol{v}_n\, \boldsymbol{v}_n^{\top}\, \boldsymbol{\kappa}_n\, \boldsymbol{\kappa}_n^{\top}\}. \tag{1.36b}$$

The moments in (1.36b) are evaluated in [30], yielding the following recursive

expressions for the entries of the autocorrelation matrix $\boldsymbol{C}_{v,n}$:

$$\begin{aligned}[\boldsymbol{C}_{v,n+1}]_{ii} =&(1-2\eta\boldsymbol{r}_{\rm md}+\eta^2\mu_1)\,[\boldsymbol{C}_{v,n}]_{ii}+\eta^2\mu_3\sum_{\substack{\ell=1\\ \ell\neq i}}^{M}[\boldsymbol{C}_{v,n}]_{\ell\ell}\\ &+(2\eta^2\mu_2-2\eta\boldsymbol{r}_{\rm od})\sum_{\substack{\ell=1\\ \ell\neq i}}^{M}[\boldsymbol{C}_{v,n}]_{i\ell}+\eta^2\mu_4\sum_{\substack{\ell=1\\ \ell\neq i}}^{M}\sum_{\substack{p=1\\ p\neq\{i,\ell\}}}^{M}[\boldsymbol{C}_{v,n}]_{\ell p}\\ &+\eta^2\,\boldsymbol{r}_{\rm md}\,\boldsymbol{J}_{\min}\end{aligned} \tag{1.37}$$

and, for $j\neq i$,

$$\begin{aligned}[\boldsymbol{C}_{v,n+1}]_{ij} =&(1-2\eta\boldsymbol{r}_{\rm md}+2\eta^2\mu_3)\,[\boldsymbol{C}_{v,n}]_{ij}+\eta^2\mu_4\sum_{\substack{\ell=1\\ \ell\neq\{i,j\}}}^{m}[\boldsymbol{C}_{v,n}]_{\ell\ell}\\ &+(\eta^2\mu_2-\eta\boldsymbol{r}_{\rm od})([\boldsymbol{C}_{v,n}]_{ii}+[\boldsymbol{C}_{v,n}]_{jj})\\ &+(2\eta^2\mu_4-\eta\boldsymbol{r}_{\rm od})\sum_{\substack{\ell=1\\ \ell\neq\{i,j\}}}^{m}([\boldsymbol{C}_{v,n}]_{i\ell}+[\boldsymbol{C}_{v,n}]_{j\ell})\\ &+\eta^2\mu_5\sum_{\substack{\ell=1\\ \ell\neq\{i,j\}}}^{m}\sum_{\substack{p=1\\ p\neq\{i,j,\ell\}}}^{m}[\boldsymbol{C}_{v,n}]_{\ell p}+\eta^2\,\boldsymbol{r}_{\rm od}\,\boldsymbol{J}_{\min}\end{aligned} \tag{1.38}$$

where $\boldsymbol{r}_{\rm md}=[\boldsymbol{R}_{\kappa\kappa}]_{ii}$ and $\boldsymbol{r}_{\rm od}=[\boldsymbol{R}_{\kappa\kappa}]_{ij}$ with $j\neq i$, as defined in (1.31), $\mu_1=\left[\det\{\boldsymbol{I}_2+4\,\boldsymbol{Q}_2\,\boldsymbol{R}_2/\beta_0^2\}\right]^{-1/2}$, $\mu_2=[\det\{\boldsymbol{I}_3+\boldsymbol{Q}_{3'}\,\boldsymbol{R}_3/\beta_0^2\}]^{-1/2}$, $\mu_3=[\det\{\boldsymbol{I}_3+2\,\boldsymbol{Q}_3\,\boldsymbol{R}_3/\beta_0^2\}]^{-1/2}$, $\mu_4=[\det\{\boldsymbol{I}_4+\boldsymbol{Q}_4\,\boldsymbol{R}_4/\beta_0^2\}]^{-1/2}$ and $\mu_5=[\det\{\boldsymbol{I}_5+\boldsymbol{Q}_5\,\boldsymbol{R}_5/\beta_0^2\}]^{-1/2}$ with $\boldsymbol{Q}_2$ and $\boldsymbol{Q}_3$ defined in (1.29) and

$$\boldsymbol{Q}_{3'}=\begin{pmatrix}4\boldsymbol{I}&-3\boldsymbol{I}&-\boldsymbol{I}\\-3\boldsymbol{I}&3\boldsymbol{I}&\boldsymbol{O}\\-\boldsymbol{I}&\boldsymbol{O}&\boldsymbol{I}\end{pmatrix}. \tag{1.39a}$$

$$\boldsymbol{Q}_4=\begin{pmatrix}4\boldsymbol{I}&-2\boldsymbol{I}&-\boldsymbol{I}&-\boldsymbol{I}\\-2\boldsymbol{I}&2\boldsymbol{I}&\boldsymbol{O}&\boldsymbol{O}\\-\boldsymbol{I}&\boldsymbol{O}&\boldsymbol{I}&\boldsymbol{O}\\-\boldsymbol{I}&\boldsymbol{O}&\boldsymbol{O}&\boldsymbol{I}\end{pmatrix}. \tag{1.39b}$$

$$\boldsymbol{Q}_5=\begin{pmatrix}4\boldsymbol{I}&-\boldsymbol{I}&-\boldsymbol{I}&-\boldsymbol{I}&-\boldsymbol{I}\\-\boldsymbol{I}&\boldsymbol{I}&\boldsymbol{O}&\boldsymbol{O}&\boldsymbol{O}\\-\boldsymbol{I}&\boldsymbol{O}&\boldsymbol{I}&\boldsymbol{O}&\boldsymbol{O}\\-\boldsymbol{I}&\boldsymbol{O}&\boldsymbol{O}&\boldsymbol{I}&\boldsymbol{O}\\-\boldsymbol{I}&\boldsymbol{O}&\boldsymbol{O}&\boldsymbol{O}&\boldsymbol{I}\end{pmatrix}. \tag{1.39c}$$

The interest reader is referred to [30] for stability and steady-state analyses of the KLMS algorithm. In the following we illustrate the accuracy of the analytical model through a simulation example.

1.6.1 Example 2

Consider the problem studied in [38, 39], for which

$$\begin{cases} y_n = \dfrac{y_{n-1}}{1+y_{n-1}^2} + u_{n-1}^3 \\ d_n = y_n + z_n \end{cases} \tag{1.40}$$

where the output signal y_n was corrupted by a zero-mean white Gaussian noise z_n with variance $\sigma_z^2 = 10^{-4}$. The input sequence $u(n)$ is zero-mean i.i.d. Gaussian with standard deviation $\sigma_u = 0.15$.

The proposed method was tested with a maximum MSE $J_{\max} = -21.5$ dB, a coherence level $\varepsilon_0 = 10^{-5}$ and a set of kernel bandwidths $\beta_0 \in \{0.0075,\ 0.01,\ 0.025,\ 0.05\}$[3]. For each value of β_0, 500 dictionary dimensions m_i, $i = 1, \ldots, 500$, were determined using 500 realizations of the input process. The length of each realization was 500 samples. Each m_i was determined as the minimum dictionary length required to achieve the coherence level ε_0. The value $m(\beta_0)$ was determined as the average of all m_i, rounded to the nearest integer. The values of $J_{\min}(\beta_0)$ were calculated from (1.25) for each pair (β_0, M). To this end, second-order moments $\boldsymbol{p}_{\kappa d}$ and $E\{d_n^2\}$ were estimated by averaging over 500 runs.

The step size value for each values of β_0 was chosen as $\eta = \eta_{\max}/10$ with $\eta_{\max}$ determined using the stability analysis in [30]. Table 1.1 shows the values of β_0 and η used in the simulations. For each simulation, the order m of the dictionary remained fixed. It was initialized for each realization by generating input vectors in $\mathcal{U}$ and filling the m positions with vectors that satisfy the desired coherence level. Thus, the initial dictionary is different for each realization. During each realization, the dictionary elements were updated at each iteration n so that the least recently added element is replaced with $\boldsymbol{u}_{n-1}$.

Figures 1.3 illustrates the accuracy of the analytical model for the four

[3]These values of β_0 are samples within a range of values experimentally verified to be adequate for the application.

Table 1.1: Summary of simulation results for Example 1.

β_0	m	η	$J_{\min}$ [dB]	$J_{\text{mse}}(\infty)$ [dB]	$J_{\text{ex}}(\infty)$ [dB]	n_ϵ
0.0075	17	0.143	-22.19	-22.04	-36.85	1271
0.01	13	0.152	-21.84	-21.69	-36.27	914
0.025	6	0.007	-21.53	-21.52	-49.15	7746
0.05	3	0.011	-21.52	-21.51	-47.00	2648

cases presented in Table 1.1. Figure 1.3 shows an excellent agreement between Monte Carlo simulations, averaged over 500 runs, and the theoretical predictions made by using (1.35).

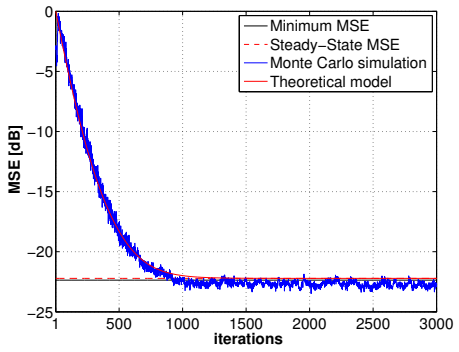
1.7 Bayesian Approaches to Kernel-Based Nonlinear Regression

1.7.1 Gaussian Processes for Regression

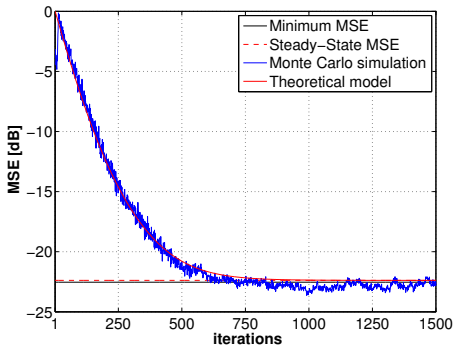
The more classical solutions of the nonlinear regression problem (1.3) using Bayesian techniques assume the knowledge of function $\psi(\cdot)$ except for a set of parameters that can be included in a parameter vector $\boldsymbol{\theta}$. This includes the cases in which the nonlinear function $\psi(\cdot)$ is actually known from physical considerations and cases in which $\psi(\cdot)$ represents a family of functions that can approximate a wide range of functional forms. In the former case, the entries of $\boldsymbol{\theta}$ tend to have physical interpretations. In the latter, they are usually parameters of an expansion to be fitted to the unknown function $\psi(\cdot)$. Kernel-based classical Bayesian solutions use the form (1.7) for $\psi(\cdot)$. Then, using the fixed size model (1.6) in (1.3), the nonlinear regression problem becomes

$$d_i = \sum_{j=1}^{m} \alpha_j \, \kappa(\boldsymbol{u}_i, \boldsymbol{u}_{\omega_j}) + \eta_i, \qquad i = 1, \ldots, m \tag{1.41}$$

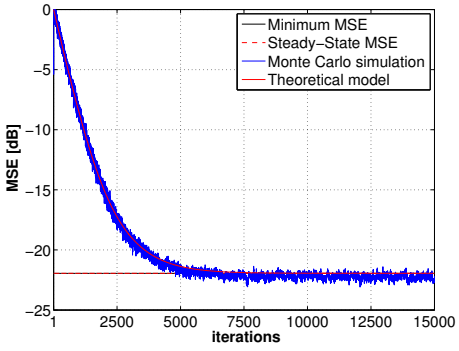
with $\boldsymbol{\theta} = [\alpha_1, \alpha_2, \ldots, \alpha_m, \sigma_\eta^2]^\top$, where $\eta_i \sim \mathcal{N}(0, \sigma_\eta^2)$ is assumed. The solution of the problem includes defining a prior distribution $p(\boldsymbol{\theta})$ for the unknown parameters and determining the posterior distribution $p(\boldsymbol{\theta}|d_1, \ldots, d_m)$. Once the posterior distribution is determined, an estima-



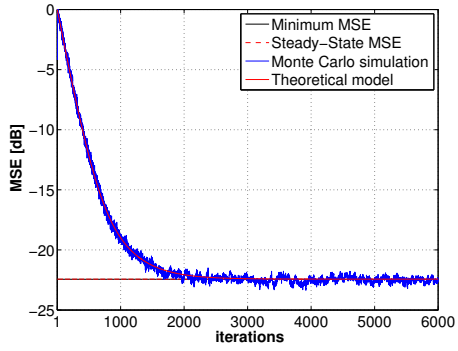
(a) $\beta_0 = 0.0075$ and $m = 17$.



(b) $\beta_0 = 0.01$ and $m = 13$.



(c) $\beta_0 = 0.025$ and $m = 6$.



(d) $\beta_0 = 0.05$ and $m = 3$.

Figure 1.3: Theoretical model and Monte Carlo simulation of KLMS for different kernel bandwidths. Ragged curves (blue): simulation results averaged over 500 runs. Continuous curves (red): Theory using (1.35).

tion criterion such as the minimum mean-square (MSE) error or the maximum *a posteriori* (MAP) can be used to estimate $\boldsymbol{\theta}$. Such classical Bayesian techniques are well documented and have given rise to popular estimators such as the least absolute shrinkage and selection operator (LASSO) proposed in [40] or the ridge regression [41]. The reader is referred, for instance, to [42, 43] for more details.

An alternative Bayesian approach that has gained popularity recently is to assume that the sequence of samples d_i, $i = 1, \ldots, m$ is characterized by a zero-mean Gaussian process ($\mathcal{GP}$) with covariance function in an RKHS [18] given by $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j)$. A common choice is the Gaussian kernel $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = \exp\left(-\|\boldsymbol{u}_i - \boldsymbol{u}_j\|^2/2\beta_0^2\right)$, where β_0 is the kernel bandwidth.

Defining the matrix $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m]$ and $\boldsymbol{d} = [d_1, \ldots, d_m]^\top$ we define the prior distribution for $\boldsymbol{d}$ as

$$\boldsymbol{d} \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{K} + \sigma_\eta^2 \boldsymbol{I}\right) \tag{1.42}$$

with $\boldsymbol{K}$ the Gram matrix whose entries $\boldsymbol{K}_{ij} = \kappa(\boldsymbol{u}_i, \boldsymbol{u}_j)$ are the kernel (covariance) functions [44] of the inputs $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$, and $\boldsymbol{I}$ is the $m \times m$ identity matrix.

Gaussian process regression aims at inferring the latent function distribution of f_* for a new (or test) input $\boldsymbol{u}_*$. Using the marginalization property [18], (1.42) can be obtained by integrating out f_* in the following joint distribution of $(\boldsymbol{d}, f_*)^\top$

$$\begin{bmatrix} \boldsymbol{d} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K} + \sigma_\eta^2 \boldsymbol{I} & \boldsymbol{\kappa}_* \\ \boldsymbol{\kappa}_*^\top & \kappa_{**} \end{bmatrix}\right) \tag{1.43}$$

with $\boldsymbol{\kappa}_*^\top = [\kappa(\boldsymbol{u}_*, \boldsymbol{u}_1), \ldots, \kappa(\boldsymbol{u}_*, \boldsymbol{u}_m)]$ and $\kappa_{**} = \kappa(\boldsymbol{u}_*, \boldsymbol{u}_*)$. The predictive distribution of f_*, or posterior of f_*, can be obtained by conditioning (1.43) on the data as

$$f_* | \boldsymbol{d}, \boldsymbol{U}, \boldsymbol{u}_* \sim \mathcal{N}\left(\boldsymbol{\kappa}_*^\top \left[\boldsymbol{K} + \sigma_\eta^2 \boldsymbol{I}\right]^{-1} \boldsymbol{d}\ ,\ \kappa_{**} - \boldsymbol{\kappa}_*^\top \left[\boldsymbol{K} + \sigma_\eta^2 \boldsymbol{I}\right]^{-1} \boldsymbol{\kappa}_*\right). \tag{1.44}$$

Using the Gaussian kernel to model the Gaussian process, the function estimation is done in an RKHS with universal approximating capability [29, p. 35] due to the smoothness and non-informativeness of this kernel.

1.7.2 Spectral Unmixing of Hyperspectral Images

The problem of unmixing hyperspectral images has been receiving an increasing interest in the recent years for various remote sensing applications (see for instance review papers [45, 46] and references therein). Spectral unmixing (SU) consists of identifying the spectral signatures of the pure materials (referred to as "endmembers") contained in a hyperspectral image and estimating the proportions of these materials (referred to as "abundances") in each pixel of the image. A classical model used for this unmixing procedure is the linear mixing model (LMM), which expresses a given pixel of the image $\boldsymbol{y} \in \mathbb{R}^L$ (acquired in L spectral bands) as a linear combination of a given number R of endmembers $\boldsymbol{m}_r$:

$$\boldsymbol{y} = \sum_{r=1}^{R} a_r \boldsymbol{m}_r + \boldsymbol{n} = \boldsymbol{M}\boldsymbol{a} + \boldsymbol{n} \tag{1.45}$$

where a_r is the abundance of the material $\boldsymbol{m}_r$ in the pixel $\boldsymbol{y}$, $\boldsymbol{a} = (a_1, ..., a_R)^\top$, $\boldsymbol{M} = [\boldsymbol{m}_1, ..., \boldsymbol{m}_R]$ is a matrix built with the different endmembers contained in the image and $\boldsymbol{n} \in \mathbb{R}^L$ is an additive noise vector. Under this model, the abundances are required to satisfy the so-called positivity and sum-to-one constraints

$$a_r \geqslant 0, \quad \sum_{r=1}^{R} a_r = 1 \tag{1.46}$$

when the vectors $\boldsymbol{m}_r$ for $r = 1, ..., R$ form set of all pure materials contained in the image. The LMM (1.45) has shown interesting properties for the SU of hyperspectral images. However, as explained in [46], this model has some severe limitations when the image contains intimate mixtures or when the light scattered by a given material reflects on other materials before reaching the sensor (leading to multipath effects). In these cases, nonlinear mixing models should be preferred to identify the endmembers contained in the image and to estimate the abundances of the pure materials in each pixel of the image. A generic model for nonlinear spectral unmixing can be written as

$$\boldsymbol{y} = f(\boldsymbol{M}, \boldsymbol{a}) + \boldsymbol{n} \tag{1.47}$$

where f is a nonlinear function whose shape can be difficult to be known a priori. In this context, Gaussian processes can be investigated to infer this nonlinear function and thus to perform nonlinear unmixing of hyperspectral images.

To apply Gaussian processes to the SU problem, we first define $\boldsymbol{s}_i$ as the transpose of the i-th row of matrix $\boldsymbol{M}$. Hence, $\boldsymbol{s}_i^\top$ contains the contributions of all endmembers to the spectral component y_i of $\boldsymbol{y}$ in the LMM, and $y_i = \boldsymbol{a}^\top \boldsymbol{s}_i + n_i$ in (1.45).

Using the joint distribution (1.43) with inputs $\boldsymbol{u}_i = \boldsymbol{s}_i$, it is straightforward to show that the posterior distribution (also referred to as predictive distribution) of f_* for a new input $\boldsymbol{s}_*$ can be written as

$$f_*|\boldsymbol{y}, \boldsymbol{M}, \boldsymbol{s}_* \sim \mathcal{N}\left(\boldsymbol{\kappa}_*^\top \left[\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right]^{-1} \boldsymbol{y}\ ,\ \kappa_{**} - \boldsymbol{\kappa}_*^\top \left[\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right]^{-1} \boldsymbol{\kappa}_*\right) \quad (1.48)$$

where $\boldsymbol{\kappa}_*^\top = [\kappa(\boldsymbol{s}_*, \boldsymbol{s}_1), \ldots, \kappa(\boldsymbol{s}_*, \boldsymbol{s}_L)]$ and $\kappa_{**} = \kappa(\boldsymbol{s}_*, \boldsymbol{s}_*)$.

The extension to a multivariate predictive distribution with test data $\boldsymbol{M}_* = [\boldsymbol{s}_{*1}, \ldots, \boldsymbol{s}_{*L}]^\top$ is straightforward and yields

$$\begin{aligned} \boldsymbol{f}_*|\boldsymbol{y}, \boldsymbol{M}, \boldsymbol{M}_* \sim \mathcal{N}\Big(&\boldsymbol{K}_* \left[\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right]^{-1} \boldsymbol{y}\ , \\ &\boldsymbol{K}_{**} - \boldsymbol{K}_* \left[\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right]^{-1} \boldsymbol{K}_*^\top\Big) \end{aligned} \quad (1.49)$$

where $[\boldsymbol{K}_*]_{ij} = \kappa(\boldsymbol{s}_{*i}, \boldsymbol{s}_j)$ and $[\boldsymbol{K}_{**}]_{ij} = \kappa(\boldsymbol{s}_{*i}, \boldsymbol{s}_{*j})$.

The predicted function $\boldsymbol{f}_*$ can finally be estimated by computing the mean of (1.49) following the MMSE principle. Then,

$$\boldsymbol{f}_*^{\mathrm{MMSE}} = \boldsymbol{K}_* \left[\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right]^{-1} \boldsymbol{y}. \quad (1.50)$$

Assuming the choice of the Gaussian kernel $\kappa(\boldsymbol{s}_i, \boldsymbol{s}_j) = \exp\left(-\|\boldsymbol{s}_i - \boldsymbol{s}_j\|^2 / 2\beta_0^2\right)$, the practical evaluation of $\boldsymbol{f}_*^{\mathrm{MMSE}}$ requires the values of the model parameters $\boldsymbol{\theta} = [\beta_0, \sigma_n^2]$ to be estimated. Following a Bayesian approach, a prior has to be defined for $\boldsymbol{\theta}$ and the posterior $p(\boldsymbol{\theta}|\boldsymbol{M}, \boldsymbol{y})$ has to be maximized with respect to $\boldsymbol{\theta}$. Using the type II maximum likelihood approximation [18], another solution for estimating $\boldsymbol{\theta}$ is to maximize the marginal likelihood $\log p(\boldsymbol{y}|\boldsymbol{M}, \boldsymbol{y})$ with respect to $\boldsymbol{\theta}$, where

$$\begin{aligned} \log p(\boldsymbol{y}|\boldsymbol{M}, \boldsymbol{y}) = &-\frac{1}{2} \boldsymbol{y}^\top \left[\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right]^{-1} \boldsymbol{y} \\ &- \frac{1}{2} \log \left|\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right| - \frac{L}{2} \log(2\pi). \end{aligned} \quad (1.51)$$

1.7.3 Example 3

Consider a nonlinearly mixed hyperspectral image pixel generated using the simplified generalized bilinear model (GBM) used in [47] with a new scaling that permits the control of the degree of nonlinearity for each nonlinear pixel generated. More precisely, the nonlinearly mixed pixels are generated using the following model

$$\boldsymbol{y} = k\,\boldsymbol{M}\boldsymbol{a} + \boldsymbol{\mu} + \boldsymbol{n} \tag{1.52}$$

where $0 \leqslant k \leqslant 1$, $\boldsymbol{\mu} = \gamma \sum_{i=1}^{R-1} \sum_{j=i+1}^{R} a_i a_j \boldsymbol{m}_i \odot \boldsymbol{m}_j$ is the nonlinear term, γ is the parameter that governs the amount of nonlinear contribution, $\odot$ is the Hadamard product, and n is an additive white Gaussian noise with variance σ_n^2. Given the parameters $\boldsymbol{M}$, $\boldsymbol{a}$, γ and σ_n^2, this model generates samples with same energy and SNR as the LMM if $k = \left[-2E_{\ell\mu} + \sqrt{4E_{\ell\mu}^2 - 4E_\ell(E_\mu - E_\ell)}\right]/2E_\ell$, where $E_\ell = \|\boldsymbol{y}_\ell\|^2$ is the energy of a noiseless linear pixel (i.e., $\boldsymbol{a}^\top \boldsymbol{M}^\top \boldsymbol{M}\boldsymbol{a}$), $E_{\ell\mu} = \boldsymbol{y}_\ell^\top \boldsymbol{\mu}$ is the "cross-energy" of the linear and nonlinear parts, and $E_\mu = \|\boldsymbol{\mu}\|^2$ is the energy of the nonlinear contribution. The degree of nonlinearity of a pixel is then defined as the ratio of the nonlinear portion to the total pixel energy

$$\eta_d = \frac{2kE_{\ell\mu} + E_\mu}{k^2 E_\ell + 2kE_{\ell\mu} + E_\mu} \tag{1.53}$$

so that $0 \leqslant \eta_d \leqslant 1$. For the simulations presented here, the endmember matrix $\boldsymbol{M}$ was composed of $R = 3$ materials (green grass, olive green paint and galvanized steel metal) extracted from the spectral library of the software ENVI$^{\text{TM}}$ [48]. Each endmember $\boldsymbol{m}_i$, $i = 1, 2, 3$, has $L = 826$ bands. The model parameters were $\boldsymbol{a} = [0.3,\, 0.6,\, 0, 1]^\top$, $k = 0.3162$ and $\gamma = 6.1464$ so that $\eta_d = 0.9$ and $\sigma_n^2 = 4.074 \times 10^{-4}$ was chosen to produce an SNR of 25dB.

Figure 1.4 shows the actual values of a test observation vector $\boldsymbol{y}_*$ (in blue) and the estimations using both a least squares fitting assuming a linear model (1.45) (in green) and a nonlinear Gaussian process based nonlinear fitting (in magenta). These results clearly illustrate the better fitting properties of the Gaussian process model for nonlinear regression problems.

Figure 1.4: Gaussian process based nonlinear function estimation. Actual pixel: Blue ragged curve. Least squares prediction assuming a linear model: Green curve. Gaussian process based nonlinear prediction: Magenta curve.

1.8 Conclusion

This chapter have presented a brief introduction to kernel-based nonlinear signal processing. This is a vast area that encounters a multitude of applications, and constitutes a very active area of research. To introduce the basic principles we concentrated on the nonlinear function estimation and regression problems, as they appear in a significant amount of nonlinear signal processing applications. After a brief introduction to reproducing kernel Hilbert spaces we discussed online kernel-based function approximation and presented a set of recursive algorithms to solve this problem. We then discussed online nonlinear system identification using the KLMS algorithm with emphasis on a methodology to study the stochastic behavior of the kernel-based adaptive estimator. Finally, we returned to the nonlinear function estimation problem, this time using a Bayesian approach based on Gaussian processes, and discussed its application to the unmixing of hyperspectral images. We presented illustrative examples for each of the applications discussed. We hope that this brief introduction will provide the reader with an appreciation of the potential of kernel-based methods for solving nonlinear signal processing problems.

Bibliography

[1] B. Schölkopf, J. C. Burges, and A. J. Smola, *Advances in kernel methods*, MIT Press, Cambridge, MA, 1999.

[2] V. N. Vapnik, *The nature of statistical learning theory*, Springer, New York, NY, 1995.

[3] N. Aronszajn, "Theory of reproducing kernels," *Trans. Amer. Math. Soc.*, vol. 68, no. 3, pp. 337–404, May 1950.

[4] J. Mercer, "Functions of positive and negative type and their connection with the theory of integral equations," *Philos. Trans. Roy. Soc. London Ser. A*, vol. 209, pp. 415–446, 1909.

[5] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K. R. Müller, "Fisher discriminant analysis with kernels," in *Proc. Advances in neural networks for signal processing*, Y. H. Hu, J. Larsen, E. Wilson, and S. Douglas, Eds., San Mateo, CA, 1999, pp. 41–48, Morgan Kaufmann.

[6] F. Abdallah, C. Richard, and R. Lengellé, "An improved training algorithm for nonlinear kernel discriminants," *IEEE Transactions on Signal Processing*, vol. 52, no. 10, pp. 2798–2806, 2004.

[7] B. Schölkopf, A. J. Smola, and K. R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[8] T. J. Dodd, V. Kadirkamanathan, and R. F. Harrison, "Function estimation in Hilbert space using sequential projections," in *Proc. IFAC Conference on Intelligent Control Systems and Signal Processing*, Faro, Portugal, 2003, pp. 113–118.

[9] T. J. Dodd, B. Mitchinson, and R. F. Harrison, "Sparse stochastic gradient descent learning in kernel models," in *Proc. Second International Conference on Computational Intelligence, Robotics and Autonomous Systems*, Singapore, 2003.

[10] Y. Engel, S. Mannor, and R. Meir, "Kernel recursive least squares," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, 2004.

[11] J. Kivinen, A.J. Smola, and R.C. Williamson, "Online learning with kernels," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2165–2176, 2004.

[12] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. Signal Process.*, vol. 57, no. 3, pp. 1058–1067, Mar. 2009.

[13] T. J. Dodd and R. F. Harrison, "Estimating Volterra filters in Hilbert space," in *Proc. IFAC Conference on Intelligent Control Systems and Signal Processing*, Faro, Portugal, 2003, pp. 538–543.

[14] Y. Wan, C. X. Wong, T. J. Dodd, and R. F. Harrison, "Application of a kernel method in modeling friction dynamics," in *Proc. IFAC World Congress*, Czech Republic, 2005.

[15] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, Prentice Hall, Upper Saddle River, 1993.

[16] H. L. Van Trees, *Detection, Estimation, and Modulation Theory - Part I*, Wiley Interscience, New York, 2001.

[17] Adriaan van den Bos, *Parameter Estimation for Scientists and Engineers*, John Wiley & Sons, 2007.

[18] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, 2006.

[19] J. Jahn, *Introduction to the Theory of Nonlinear Optimization*, Springer, 3rd edition, 2007.

[20] L. Hoegaerts, *Eigenspace methods and subset selection in kernel based learning*, Ph.D. thesis, Katholieke Universiteit Leuven, 2005.

[21] J. A. K. Suykens, T. van Gestel, J. de Brabanter, B. de Moor, and J. Vandewalle, *Least Squares Support Vector Machines*, World Scientific, Singapore, 2002.

[22] B. J. de Kruif and T. J. A. de Vries, "Pruning error minimization in least squares support vector machines," *IEEE Transactions on Neural Networks*, vol. 14, no. 3, pp. 696–702, 2003.

[23] J. A. K. Suykens, J. de Brabanter, L. Lukas, and J. Vandewalle, "Weighted least squares support vector machines: robustness and sparse approximation," *Neurocomputing*, vol. 48, pp. 85–105, 2002.

[24] G. C. Cawley and N. L. C. Talbot, "Improved sparse least-squares support vector machines," *Neurocomputing*, vol. 48, pp. 1025–1031, 2002.

[25] L. Hoegaerts, J. A. K. Suykens, J. Vandewalle, and B. de Moor, "Subset based least squares subspace regression in RKHS," *Neurocomputing*, vol. 63, pp. 293–323, 2005.

[26] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K. R. Müller, G. Rätsch, and A. J. Smola, "Input space versus feature space in kernel-based methods," *IEEE Transactions On Neural Networks*, vol. 10, no. 5, pp. 1000–1017, 1999.

[27] P. Honeine, C. Richard, and J. C. M. Bermudez, "On-line nonlinear sparse approximation of functions," in *Proc. IEEE ISIT'07*, Nice, France, June 2007, pp. 956–960.

[28] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. Signal Process.*, vol. 57, no. 3, pp. 1058 –1067, March 2009.

[29] W. Liu, J. C. Principe, and S. Haykin, *Kernel Adaptive Filtering*, Jonh Wiley & Sons, Inc., 2010.

[30] W. D. Parreira, J. C. M. Bermudez, C. Richard, and J.-Y. Tourneret, "Stochastic behavior analysis of the Gaussian kernel least-mean-square algorithm," *IEEE Trans. Signal Process.*, vol. 60, no. 5, pp. 2208–2222, May 2012.

[31] W. D. Parreira, J. C. M. Bermudez, C. Richard, and J.-Y. Tourneret, "Steady-state behavior and design of the Gaussian KLMS algorithm,"

in *19th Eur. Signal Process. Conf. (EUSIPCO 2011)*, Barcelone, Spain, 2011, pp. 121–125, EURASIP.

[32] W. D. Parreira, J. C. M. Bermudez, C. Richard, and J.-Y. Tourneret, "Stochastic behavior analysis of the Gaussian kernel least mean square algorithm," in *2011 IEEE Int. Conf. Acoust. Speech Signal Process.*, Prague, 2011, pp. 4116–4119.

[33] C. Richard and J. C. M. Bermudez, "Closed-form conditions for convergence of the Gaussian kernel-least-mean-square algorithm," in *2012 Conf. Rec. Forty Sixth Asilomar Conf. Signals, Syst. Comput.*, Pacific Grove, CA, Nov. 2012, pp. 1797–1801, IEEE.

[34] J. Chen, W. Gao, C. Richard, and J. C. M. Bermudez, "Convergence analysis of kernel LMS algorithm pre-tuned dictionary," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, May 2014, IEEE, pp. 7293–7297.

[35] J. Omura and T. Kailath, "Some useful probability distributions," Tech. Rep. 7050-6, Stanford Electronics Laboratories, Stanford University, Stanford, California, USA, 1965.

[36] J. Minkoff, "Comment: On the unnecessary assumption of statistical independence between reference signal and filter weights in feedforward adaptive systems," *IEEE Trans. Signal Process.*, vol. 49, no. 5, pp. 1109, May 2001.

[37] A. H. Sayed, *Fundamentals of adaptive filtering*, John Wiley & Sons, Hoboken, NJ, 2003.

[38] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networs*, vol. 1, no. 1, pp. 3–27, March 1990.

[39] D. P. Mandic, "A generalized normalized gradient descent algorithm," *IEEE Signal Processing Letters*, vol. 2, pp. 115–118, February 2004.

[40] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[41] I. E. Frank and J. H. Frieman, "A statistical view of some chemometrics regression tools," *Technometrics*, vol. 35, no. 2, pp. 109–135, 1993.

[42] G. A. F. Seber and C. J. Wild, *Nonlinear Regression*, Wiley-Interscience, 2003.

[43] C. M. Bishop, Ed., *Pattern Recognition and Machine Learning*, Information Science and Statistics. Springer, New York, 2006.

[44] B. Schölkopf and A. J. Smola, *Learning with Kernels*, MIT Press, Cambridge, MA, 2002.

[45] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot, "Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches," *IEEE J. Sel. Topics Appl. Earth Observations and Remote Sens.*, vol. 5, no. 2, pp. 354–379, 2012.

[46] N. Dobigeon, J.-Y. Tourneret, C. Richard, J. C. M. Bermudez, S. McLaughlin, and A. O Hero, "Nonlinear unmixing of hyperspectral images: Models and algorithms," *IEEE Signal Process. Mag.*, vol. 31, no. 1, pp. 82–94, Jan. 2014.

[47] Y. Altmann, N. Dobigeon, J.-Y. Tourneret, and J. C. M. Bermudez, "A robust test for nonlinear mixture detection in hyperspectral images," in *IEEE Int. Conf. Acoust. Speech, Signal Process.*, Vancouver, Canada, 2013, pp. 2149–2153, IEEE Signal Processing Society.

[48] RSI (Research Systems Inc.), "ENVI User's guide Version 4.0," Sept. 2013.