IMPROVING MULTIKERNEL ADAPTIVE FILTERING WITH SELECTIVE BIAS

Magno T. M. Silva, Renato Candido*  
Universidade de S˜ao Paulo  
{magno, renatocan} @lps.usp.br

Jerónimo Arenas-García, Luis A. Azpicueta-Ruiz†  
Univ. Carlos III de Madrid, Spain  
{jarenas, azpicueta} @tsc.uc3m.es

ABSTRACT

In this paper, we propose a scheme to simplify the selection of kernel adaptive filters in a multikernel structure. By multiplying the output of each kernel filter by an adaptive biasing factor between zero and one, the degrading effects of poorly adjusted kernel filters can be minimized, increasing the robustness of the multikernel scheme. This approach is able to deal with the lack of the necessary statistical information for an optimal adjustment of the filter and its structure. The advantages of the proposed scheme with respect to other multi-kernel solutions are checked by means of numerical examples in the context of signal prediction and system identification.

Index Terms— Nonlinear adaptive signal processing, kernel adaptive filtering, cooperative architectures

1. INTRODUCTION

Kernel adaptive filters (KAFs) are able to solve nonlinear problems by implicitly projecting the input vector to a high dimensional space, where a standard linear adaptive filter is employed (see, e.g., [1–3]). They have been mainly applied in system identification [4], echo cancellation [5], time series prediction [1, 6–8], and channel equalization [7, 8]. Nonlinear adaptive signal processing, kernel adaptive filtering, and cooperative architectures have been mainly applied in system identification [4], echo cancellation [5], time series prediction [1, 6–8], and channel equalization [7, 8].

The kernel version of the least-mean-squares (LMS) algorithm, denoted as KLMS [9], is the most popular among the different KAFs. KLMS maps the input column-vector x(n) ∈ ℜN into a high dimensional feature space F as ϕ(x(n)), using a Mercer’s kernel, which is a continuous, symmetric, and positive-definite function κ : ℜN × ℜN → ℜ, such that κ(x, x′) = ϕ(x)T ϕ(x′), where the superscript T denotes transposition. The Gaussian kernel is one of the most used kernel functions in the literature. It is defined as κ(x, x′) = e−∥x−x′∥2/(2σ2)), where ∥·∥ denotes the Euclidean norm and σ > 0 the kernel width [1].

In the space F, the LMS algorithm is used to update the filter weight column-vector Ω(n−1) in order to estimate the desired signal d(n). The Representer’s Theorem states that Ω(n−1) can be expressed as a linear combination of past input samples, so that the filter output can be obtained as

\[ y(n) = \varphi(x(n))^T \Omega(n-1) = \mu \sum_{i=1}^{n-1} e(i) \kappa(x(n), x(i)), \]  

where e(i) = d(i) − y(i) and μ is a step size.

KAFs offer a powerful alternative to nonlinear filters, such as Volterra and functional link adaptive filters [10, 11]. However, their main drawbacks are the selection of an appropriate kernel and the high computational burden and memory, since the dictionary size (expansion in (1)) grows linearly with the incoming samples.

To avoid this linear growth, different sparsification techniques have been proposed to include in the dictionary only informative data (see, e.g., [12–16]). In this context, an alternative to these techniques is the quantized KLMS (QKLMS) algorithm [3]. If the minimum distance between the input vector x(n) and all the vectors contained in the dictionary is smaller than a threshold, the dictionary is kept unchanged and the coefficient of the closest vector is updated. Otherwise, the input vector is included in the dictionary and its coefficient must also be added in the memory. The use of “redundant” data to locally update the coefficient of the closest vector leads to an algorithm with better accuracy and a more compact dictionary [3].

To select an appropriate kernel, many adaptive multiple kernel approaches have been proposed in the literature. For instance, the single-input multikernel LMS (SI-KLMS) scheme [17], where the kernel function is a convex combination of kernels, i.e., κ(x, x′) = ∑Lℓ=1 βℓκℓ(x, x′), where βℓ, ℓ = 1, · · · , L are coefficients fixed beforehand with βℓ > 0 and ∑Lℓ=1 βℓ = 1. The component kernels κℓ(·, ·) can simply be usual kernels with different parameter settings. Since the sum of Mercer’s kernels ∑Lℓ=1 βℓκℓ(·, ·) is also a valid Mercer’s kernel, this approach is equivalent to the monokernel KLMS but considers a different feature space.

A convex combination of two KLMS filters was also proposed to select a proper kernel function [2, 6]. These works recur to the standard combination of two filters [18, 19], where the global output is computed as y(n) = η(n)y1(n) + (1 − η(n))y2(n) being yℓ(n), ℓ = 1, 2 the outputs of the KLMS filters and η(n) a mixing parameter. Depending on the value of η(n), the combination may show a preference for one of the kernels. If the mixing parameter takes an intermediate value, both filters are used simultaneously, and the combination may perform better than each individual filter.

Finally, the multiple-input multikernel LMS (MI-KLMS) scheme [7, 8] augments the dimensionality of the feature space by mapping input samples to multiple feature elements, which in turn are linearly combined to yield the output estimate. This has been proven advantageous in some scenarios when compared with the standard kernel regression paradigm, allowing the learning of multiple nonlinear features presented in the data [8]. Considering the multidimensional mapping of x(n) to a feature space M, given by ⌀(x(n)) = [ϕ1(x(n)) · · · ϕL(x(n))]T, with ϕℓ ∈ ℜL for ℓ = 1, · · · , L, and the inner product in M defined as ⌀(ϕ, ϕ′)M = ∑Lℓ=1 (ϕℓ, ϕ′ℓ), it is possible to show that M is a Hilbert space and presents the reproducing properties [8]. Therefore, there is a kernel K associated with the inner product in M, K(x, x′) = ⌀(ϕ(x), ⌀(ϕ′(x)))M. Thus, the filter output can be calculated by generalizing (1) to the new kernel,

\[ y(n) = \sum_{i=1}^{n-1} \mu(x(n)) \sum_{ℓ=1}^{L} \kappaℓ(x(i), x(n)). \]  

*The work of Silva has been partly supported by FAPESP under Grant 2017/20378-9 and CNPq under Grant 304715/2017-4.
†The work of Arenas-García has been partly supported by MINECO Projects TEC2014-52289-R, TEC2016-81900-REDT, and TEC2017-83838-R, and by Comunidad de Madrid Project PRICAM S2013/ICE-2933. The work of Azpicueta-Ruiz has been partly supported by MINECO Projects TEC2014-52289-R and TEC2017-83838-R.
This approach is equivalent to considering $L$ KLMS filters in parallel and adapting them using a single error signal given by $e(n) = d(n) - y(n)$ [8]. In terms of performance, simulation results show that MI-KLMS generally tends to present a lower mean-square error (MSE) in comparison with SI-KLMS as a trade-off to its higher computational cost. In some applications, MI-KLMS may present lower MSE in comparison to the convex combination of two KLMS filters [2, 7]. However, in certain situations, if the parameters of one kernel component are poorly adjusted, then the convex combination is able to select the best component filter (making $\gamma(n) \approx 0$ or $\gamma(n) \approx 1$) and may outperform SI-KLMS and MI-KLMS, which have their output degraded by the poorly adjusted kernel. We notice that all these $L$-kernel approaches present similar computational cost considering the same sparsification techniques for the dictionaries.

In this paper, we propose a scheme to improve the selection of kernel filters in the MI-KLMS scheme. By multiplying the output of each kernel filter by an adaptive biasing factor between 0 and 1 [20], we can minimize the degrading effects of poorly adjusted kernel filters in the MI-KLMS scheme. Thus, our approach gets advantages of MI-KLMS and of the convex combination of KLMS filters. The paper is organized as follows. The proposed scheme is shown in Section 2, followed by some simulation results and conclusions, in Sections 3 and 4, respectively.

2. SELECTING KERNEL FILTERS IN MI-KLMS SCHEME

Fig. 1 depicts a block diagram of the proposed scheme, which is named as robust MI-KLMS (R-MI-KLMS) from now on. Our proposal is based on the inclusion of an upper layer on the MI-KLMS filter composed of different biasing factors with values between 0 and 1. At each branch, the factor $\lambda_i(n)$, $\ell = 1, 2, \ldots, L$ multiplies the output of the $\ell$th filter that composes MI-KLMS. In this way, the output of the proposed R-MI-KLMS can be calculated as

$$y_b(n) = \sum_{\ell=1}^{L} \lambda_\ell(n)y_\ell(n),$$

where $y_\ell(n)$ is the output of each individual adaptive kernel. This permits to weight the output of each kernel activating or deactivating the output of unnecessary kernels in the global filter output.

![Fig. 1. R-MI-KLMS with $L = 2$ KAFs applied to nonlinear system identification, where $v(n)$ is a measurement noise.](image)

All the biasing parameters are adapted to minimize the MSE of the global output, i.e., $E[e^2(n)]$, where $E[\cdot]$ stands for expectation and $e(n) = d(n) - y_b(n)$, as can be seen in Fig. 1 for $L = 2$ KAFs. Adapting these biasing factors will result in $\lambda_i(n) \to 1$ when the modeling capability of the kernel contributes to a reduction of the MSE, whereas $\lambda_i(n) \to 0$ otherwise. In the latter case, not only the output of the corresponding kernel will be suppressed, but also the gradient noise incurred by its adaptation, which is specially important in scenarios where the signal-to-noise ratio (SNR) is low [20].

An important difference between the MI-KLMS scheme and our proposal is related with the error employed to adapt each kernel. The update of all kernel filters in the MI-KLMS scheme use the same error $e(n) = d(n) - \sum_{\ell=1}^{L} y_\ell(n)$, whereas the $\ell$th kernel of the R-MI-KLMS scheme updates its coefficients with [21]

$$e_\ell(n) = d(n) - \sum_{\ell=1, \ell \neq \ell}^{L} \lambda_\ell(n)y_\ell(n),$$

in order to keep the best properties of each possible biased kernel of other branches. Thus, if $\lambda_i(n) \approx 0$ with $\ell \neq \ell$, Eq. (4) avoids incorporating in the update of the $\ell$th kernel information from kernels that are not useful or that incorporate a large amount of gradient noise.

Different strategies can be followed to adapt the biasing parameters of the R-MI-KLMS scheme. In our proposal, and following a similar strategy to that of [20], we reinterpret the output of each branch of R-MI-KLMS as a convex combination with a virtual kernel whose output is always zero. Making that, we get

$$y_b(n) = \sum_{\ell=1}^{L} \lambda_\ell(n)y_\ell(n) = \sum_{\ell=1, \ell \neq \ell}^{L} \lambda_\ell(n)y_\ell(n) + [1 - \lambda_\ell(n)] \cdot 0.$$  

In this way, we can adapt the value of $\lambda_i(n)$ with $\ell = 1, \ldots, L$ employing well-known adaptive rules [19].

However, instead of adapting directly each biasing factor $\lambda_i(n)$, we adapt an auxiliary biasing parameter $\alpha_i(n)$ univocally related with $\lambda_i(n)$ by means of the sigmoidal function [19], i.e., $\lambda_i(n) = \text{sgm}[\alpha_i(n - 1)] = [1 + e^{-\alpha(n-1)}]^{-1}$. For this, we follow a popular scheme using a power-normalized stochastic gradient descent method to minimize $e^2(n)$, i.e.,

$$\alpha_\ell(n) = \alpha_\ell(n - 1) + \frac{\mu_\ell}{p_\ell(n)}e(n)y_\ell(n)\lambda_\ell(n)[1 - \lambda_\ell(n)],$$

where $\mu_\ell$ is a step size and $p_\ell(n) = \beta_\ell p_\ell(n - 1) + (1 - \beta)y_\ell^2(n)$ represents a low-pass filtered version of $y_\ell^2(n)$, with $0 < \beta < 1$ being a smoothing factor. This scheme to adapt the biasing factors has two advantages:

- the use of the sigmoidal activation function is an easy way to keep $\lambda_i(n) \in [0, 1]$; and
- the factor $\lambda_i(n)[1 - \lambda_i(n)]$ in (6) reduces the gradient noise introduced in the adaptation of $\alpha_i(n)$ when the biasing factor is close to the limiting values 0 or 1, since the effective step size in (6) is reduced. It should be noted that $\lambda_i(n) = 1$ and $\lambda_i(n) = 0$ represent two important situations where a kernel is completely considered or discarded for the global output.

From a practical point of view, the adaptation of the biasing parameters has to be slightly modified [19]. To avoid the paralysis of the adaptation in (6), $\alpha_i(n)$ has to be restricted to a range of $[-\alpha^+, \alpha^-]$ (being $\alpha^+ = 4$ a common value used [18, 19]). As a consequence, to guarantee that $\lambda_i(n) \in [0, 1]$ under this restriction, the relation between $\lambda_i(n)$ and $\alpha_i(n)$ is given as [20]

$$\lambda_i(n) = \psi[\alpha_i(n - 1)] = \frac{\text{sgm}[\alpha_i(n - 1)] - \text{sgm}[-\alpha^-]}{\text{sgm}[-\alpha^-] - \text{sgm}[-\alpha^+]}.$$

The definition of $\lambda_i(n)$ in (7) gives rise to a slight modification of Eq. (6), as can be seen in Table 1, which shows the pseudocode of the proposed R-MI-KLMS scheme considering $L$ QKLMS filters. In this table, the dictionary for the $\ell$th KAF at time instant $n$ is denoted as $C_\ell(n) = \{x(c_{\ell,j})\}_{j=1}^{N_{\ell}(c)}$, where $x(c_{\ell,j})$ is the $j$th element and $N_{\ell}(c)$ is its cardinality, which can vary from a time instant to another. The index $c_{\ell,i} \in \{1, 2, \cdots, n - 1\}$ is used to distinguish the dictionary elements $x(c_{\ell,1}), \cdots, x(c_{\ell,N_{\ell}(\ell),\ell})$ from the input vector $x(n)$. Analogously, the vector $a_{\ell}(n) = [a_{\ell,1}(n) \cdots a_{\ell,N_{\ell}(\ell),\ell}]^T$ contains the coefficients of the $\ell$th KAF.
Input: \( (x(n) \in \mathbb{R}^n, d(n)) \), \( n = 1, 2, \ldots \)

Initialization: Choose step sizes \( \mu_0 > 0 \) and \( \mu_\alpha, \ell = 1, \ldots, L \), kernel widths \( \sigma \), thresholds \( \varepsilon \), \( \ell = 1, \ldots, L \), forgetting factor \( \beta \) and initialize the dictionaries \( C_\ell(n) = \{x(1)\} \), the parameters \( \alpha_\ell(1) = \alpha^+, p_\ell(1) = 1 \), and coefficient vectors \( \alpha_\ell(1) = [\mu d(1)] \), \( \ell = 1, \ldots, L \).

For \( n = 2, 3, \ldots \), do:

For \( \ell = 1, \ldots, L \), do:

\[
\lambda_\ell(n) = \psi[\alpha_\ell(n-1)] = \text{sgm}[\alpha_\ell(n-1)] - \text{sgm}[-\alpha^+] \frac{\text{sgm}[\alpha^+]}{\text{sgm}[-\alpha^+]}
\]

\[
y_\ell(n) = \sum_{j=1}^{N_\ell} a_{\ell,j}(n-1) \psi[\alpha_\ell(\ell), x(c_{\ell,j})]
\]

\[
\psi'[\alpha_\ell(n-1)] = \text{sgm}[\alpha_\ell(n-1)](1 - \text{sgm}[\alpha_\ell(n-1)])
\]

\[
p_\ell(n) = \beta p_\ell(n-1) + (1 - \beta) \psi'[\alpha_\ell(n-1)]
\]

end

\[ y_\ell(n) = \sum_{\ell=1}^{L} \lambda_\ell(n)y_\ell(n) \]

\[ e(n) = d(n) - y_\ell(n) \]

For \( \ell = 1, \ldots, L \), do:

\[
e_\ell(n) = d(n) - \left[y_\ell(n) + \sum_{\ell=1}^{L} \lambda_\ell(n) y_\ell(n) \right]
\]

\[
\alpha_\ell(n) = \alpha_\ell(n-1) + \mu e_\ell(n)y_\ell(n)\psi'[\alpha_\ell(n-1)]
\]

if \( \alpha_\ell(n) > \alpha^+ \)

\[ \alpha_\ell(n) \leftarrow \text{sgm}[\alpha_\ell(n)] \alpha^+ \]

end

\[ \text{dis}(x(n), C_\ell(n)) = \min_{1 \leq \ell \leq N_\ell} \|x(n) - x(c_{\ell,j})\| \]

If \( \text{dis}(x(n), C_\ell(n)) \leq \varepsilon \), keep the dictionary unchanged:

\[ C_\ell(n+1) = C_\ell(n) \]

and quantize \( x(n) \) to the closest vector by updating the coefficient of that vector, i.e.,

\[ a_{\ell,j}(n) = a_{\ell,j}(n-1) + \mu e_{\ell,j}(n), \]

where

\[ j^* = \arg \min_{1 \leq \ell \leq N_{\ell}} \|x(n) - x(c_{\ell,j})\| \]

Otherwise, assign a new center and corresponding new coefficient:

\[ C_\ell(n+1) = C_\ell(n) \{x(n), x(n)\} \]

\[ \alpha_\ell(n) = [\alpha_\ell(n-1), \mu e_{\ell,j}(n)] \]

end

3. SIMULATION RESULTS

In this section, we compare the performance of our proposal, with that of three multikernel solutions: MI-KLMS [7,8], SI-KLMS [17], and with the adaptive convex combination of two KLMS, denoted here as CC-KLMS [2,6]. Due to the inherent advantages of QKLMS, we consider this algorithm in the Pseudocode of Table 1 as well as in all simulations of this section. Therefore, the letter “Q” appears in the acronyms of all schemes. We consider only the Gaussian kernel function and \( L = 2 \) filters for all schemes.

In the first set of simulations, we consider the problem of nonlinear prediction, where the desired sequence is computed as [7]

\[
d(n) = [0.8 - 0.5 \exp(-d^2(n-1))] d(n-1) - [0.3 + 0.09 \exp(-d^2(n-1))] d(n-2) + 0.1 \sin(d(n) - 1)\pi, \]

for \( 0 \leq n < N_{\ell}/2 \), and

\[
d(n) = [0.2 - 0.7 \exp(-d^2(n-1))] d(n-1) - [0.8 + 0.8 \exp(-d^2(n-1))] d(n-2) + 0.2 \sin(d(n-1) - 1)\pi, \]

for \( N_{\ell}/2 \leq n < N_{\ell}, \) being the number of iterations and \( d(n) = 0.1 \) for \( n < 0 \).

Considering \( N = 2, N_{\ell} = 10^4 \), and adjusting the parameters as in Table 2, we obtain the results shown in Figs 2 and 3. Fig. 2 shows the learning curves, evolution of the mixing parameter \( q(n) \) for CC-QKLMS, and evolution of the biasing parameters \( \lambda_1(n) \) and \( \lambda_2(n) \) for the R-MI-KLMS scheme, considering kernel widths as \( \sigma_1 = 0.1 \) and \( \sigma_2 = 1 \). We can observe that the CC-QKLMS scheme performs as its best component filter, which is confirmed by the mixing parameter evolution. SI-QKLMS1 outperforms SI-QKLMS2, which shows the importance of choosing adequately the step size and the dictionary threshold for this scheme. Finally, MI-QKLMS and its robust version present the same performance, since both component filters are considered to compute the overall output of R-MI-KLMS, which can be confirmed by the evolution of the biasing parameters. These schemes outperform all multiple kernel solutions considered in the simulation in terms of convergence rate and steady-state error.

![Fig. 2. MSE considering signal prediction performed with CC-QKLMS (a), with the mixing parameter (d); SI-QKLMS (b); MI-MQKLMS and R-MIQKLMS (c), with biasing parameters (e). Settings \( \sigma_1 = 0.1, \sigma_2 = 1, \mu_\alpha = \mu_\alpha = 1.5 \), and other parameters as in Table 2. To facilitate the visualization, the MSE curves are smoothed using a moving average filter with 64 taps.](image_url)
observe that R-MI-QKLMS outperforms other multikernel schemes in terms of steady-state EMSE in the identification application with two different systems modified from [4]. For the first half of the experiment, we consider the first system, where the desired signal is 
\[d_1(n) = s_2(n) + z_1(n),\]
with
\[s_2(n) = \frac{s_1(n-1)}{1+s_1^2(n-1)} + x^3(n-1),\]
where \(x(n)\) is the input signal and \(z_1(n)\) is a zero-mean Gaussian noise with variance \(\sigma^2_{z_1} = 10^{-4}\). At \(n = 5 \times 10^4\), we consider an abrupt change that leads to a second nonlinear system, which generates a desired sequence computed as \(d_2(n) = \psi(s_2(n)) + z_2(n)\), with \(z_2(n)\) a zero mean Gaussian noise with variance \(\sigma^2_{z_2} = 10^{-6}\), \(s_2(n) = x(n) + 0.5x(n-1) - 0.2s_2(n-1) + 0.3s_2(n-2)\), and
\[\psi(s) = \begin{cases} \frac{s}{3\sqrt{0.1 + 0.9s^2}} & \text{for } s \geq 0 \\ \frac{-s}{3[1 - \exp(0.7s)]} & \text{for } s < 0 \end{cases}\]

Fig. 4 shows the learning curves in terms of excess MSE (EMSE), evolution of the mixing parameter \(\eta(n)\) for CC-QKLMS, and evolution of the biasing parameters for R-MI-QKLMS. The input signal \(x(n)\) was i.i.d. randomly generated from a zero-mean Gaussian distribution with standard deviation \(\sigma_x = 0.15\). We can observe that CC-QKLMS performs as the best component filter. The performance of SI-QKLMS2 is better than that of SI-QKLMS1 in the identification of the first system but for the second system, SI-QKLMS1 outperforms SI-QKLMS2, emphasizing the importance of adjusting the step size and dictionary threshold adequately for this scheme. This result also shows that, even considering a smaller number of elements in the dictionary (by using a lower value for the dictionary threshold), it is possible to obtain an improved performance, depending on the other parameters. In this scenario, MI-QKLMS presents a poor performance and it is possible to observe the importance of using the improved version R-MI-QKLMS. This robust version is able to minimize the degrading effects of unnecessary kernels, achieving a learning curve with a convergence rate slightly higher than that of CC-QKLMS.

Table 3 shows the gain of R-MI-QKLMS in relation to other multikernel schemes in terms of steady-state EMSE in the identification of the second system for different values of SNR. We can observe that R-MI-QKLMS outperforms other multikernel schemes for the considered range of SNR. The advantage of R-MI-QKLMS is especially remarkable for low SNRs, which is useful in the frequent situations in which the SNR is unknown or changes over time. This is an important capability that the CC-QKLMS scheme lacks.

4. CONCLUSION
KAFs are important tools to solve nonlinear problems such as time series prediction, system identification, among others. Besides the high computational burden and memory, the selection of an appropriate kernel is essential to obtain a good performance in terms of MSE. To circumvent this problem, multikernel schemes that combine two or more KAFs with different kernel functions have been proposed in the literature. However, if the parameters of one kernel component are poorly adjusted, the performance of these schemes can be degraded. In this paper, we proposed a robust multikernel scheme that is able to activate or deactivate the output of unnecessary kernels in the global filter output. With a slight increase in the computational cost in comparison with MI-QKLMS, the proposed scheme can outperform other multikernel solutions, in scenarios where the settings of one or more kernels are not appropriate, and/or when the SNR is low. In a future work, we intend to theoretically analyze the proposed scheme and consider it in other applications.
5. REFERENCES


