World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:11, No:5, 2017

Online Prediction of Nonlinear Signal Processing Problems Based Kernel Adaptive Filtering

Hamza Nejib, Okba Taouali

***Abstract*—**This paper presents two of the most knowing kernel adaptive filtering (KAF) approaches, the kernel least mean squares and the kernel recursive least squares, in order to predict a new output of nonlinear signal processing. Both of these methods implement a nonlinear transfer function using kernel methods in a particular space named reproducing kernel Hilbert space (RKHS) where the model is a linear combination of kernel functions applied to transform the observed data from the input space to a high dimensional feature space of vectors, this idea known as the kernel trick. Then KAF is the developing filters in RKHS. We use two nonlinear signal processing problems, Mackey Glass chaotic time series prediction and nonlinear channel equalization to figure the performance of the approaches presented and finally to result which of them is the adapted one.

***Keywords*—**KLMS, online prediction, KAF, signal processing, RKHS, Kernel methods, KRLS, KLMS.

I. Introduction

SIGNALS and signal processing pervade our everyday lives, that's why it becomes the most interesting topic in electrical engineering. Signal processing is a large field, it is the operation that applied to an original input signal in order to produce a new output signal, and it can be applicable to implement a certain processing task, studying a certain signal and to predict a system's output which is the main object of this work. In the most cases, signal processing problems require online adaptive procedure with sparsification to handle the huge amount of data, and it needs a high-dimensional space hypothesis, so we are talking about the reproducing kernel Hilbert space (RKHS) [1] where the model is a linear combination of kernel functions applied to transform the observed data from the input space to a high dimensional feature space of vectors, this idea known as the kernel trick. All kernel methods formulate learning and estimate problems in RKHS, kernel methods are powerful nonlinear approaches and they are widely used in many fields such as bioinformatics, machine learning and nonlinear signal processing. For offline scenario, the most popular kernel methods are support vector machines (SVM) [2] and kernel principal component analysis (KPCA) [3], these methods are designed to regression, classification, and novelty detection, but they are inefficient for online prediction. The great methodology dealing with online prediction problems is adaptive filtering, were filtering is the process of removing certain portions of the input signal in order to create a new signal, so a filter is the most important operation in signal processing, it acts on a signal to produce a modified one. By developing adaptive filter in RKHS, we get the so-called "kernel adaptive filtering" (KAF) [4]. KAF algorithms are a family of online kernel learning algorithms, where a linear combination of kernel functions used to implement linear adaptive algorithms in order to obtain nonlinear filters in the original input space. The aim of this work is to use two of the most popular KAF algorithms, kernel least mean squares (KLMS) [5] and recursive recursive least squares (KRLS) [6], so as to predict output of nonlinear signal processing problems. The remainder of this paper is divided into four sections. We start by introducing adaptive filters in Section II. An overview of the reproducing kernel Hilbert space is presented in Section III. Next in Section IV, we give a brief introduction to both kernel recursive least squares and kernel least mean squares methods. We experimentally prove the most adaptive methods for online prediction problems in Section V. Finally Section VI summarizes the conclusions of this work.

Hamza Nejib is with the National Engineering School of Monastir, University of Monastir, Avenue Ibn El Jazzar, Monastir 5019, Tunisia (e-mail: hamza-nejib@outlook.com).

II. Adaptive Filtering

Adaptive filtering imply the changing filter coefficients by the time to adapt changing signal characteristics. An adaptive filter is a computational device that recursively models the relationship between the input and output signals of the filter. The adaptive filter consist of a signal processed by the filter, formulation defining the way how the output of the filter is computed according to its input, adjustable coefficients that update iteratively, and the adaptive algorithm describing how the coefficient are adjusted. The most knowing adaptive algorithms are those presented next in this paper. An adaptive filter requires an additional input signal d(n) and returns an additional output signal e(n). A typical adaptive filter contain the following elements:

- x(n) is the input signal
- y(n) is the corresponding output signal
- d(n) is an additional input signal to the adaptive filter
- e(n) is the error signal (difference between d(n) and y(n))

Adaptive filter can be used in different applications such as noise cancellation, inverse modeling, system identification and prediction problems witch is our focus in this article. Prediction is to estimate the values of a signal at a future time without having any prior knowledge. Given a random signal $x(n)$, the delayed version of the random signal is $u(n)$, the filter generates an output $y(n)$ in order to bring out the error $e(n)$ between the desired response (additional input) and the first output delivered by the filter. Then $e(n)$ denote the second system output.

III. Reproducing Kernel Hilbert Space

A. Positive-Definite Kernel (PDK)

Let X be an input space, the function $k : X \times X \to \mathbb{R}$ is a positive definite kernel [7] if:

1) $k(a_i, a_j) = k(a_j, a_i)$
2) $\forall\ a_1, ..., a_n \in X^n$ and $p_1, ..., p_n \in \mathbb{R}_+$ then:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} p_i p_j k(a_i, a_j) \geq 0$$

B. Reproducing Kernel

Let H be a Hilbert space, $\forall\ f, g \in H$, the inner product is designed by $< f, g >$, and the norm of f in H is $||f|| =< f, f >^{\frac{1}{2}}$. The function $k : X \times X \to \mathbb{R}$ labeled reproducing kernel of H iff:

1) $\forall\ x, y \in X$, $k_x(y) = k(x, y)$.
2) $\forall\ x \in X$, $\forall\ f \in H$, then: $f(x) =< f, k_x >_H$

C. RKHS

The reproducing kernel Hilbert space (RKHS) [1] is a Hilbert space where, there is a reproducing kernel k of H. The RKHS, denoted by H_k, the norm indicated by $||.||_{H_k}$, and the inner product symbolized by $< ., . >_{H_k}$.

D. Optimization Problem

The main goal is to infer a functional relation $\hat{y} = f(x)$ based on a set of training experimental data $D = \{(x_1, y_1), ..., (x_n, y_n)\}$, where $x_i = [x_{i,1}, ..., x_{i,d}]^T \in \mathbb{R}^d$ and $y_i = [y_1, ..., y_i]^T \in \mathbb{R}$ are the inputs and the outputs, respectively. Thanks to the statistical learning theory [8], the problem in the RKHS H_k, can be figured as a minimization of the regularized empirical risk. Thus, the function $\hat{y} \in H_k$ has the following form [9]:

$$\hat{y} = \min_{f \in H_k} \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda||f||^2_{H_k} \tag{1}$$

where n is the number of training data and λ is a regularization parameter chosen in order to ensure a generalization ability to the solution [10]. Thanks to the Representer Theorem [2], a large class of optimization problems in the RKHS have solutions that can be expressed as kernel expansions in terms of the training data only. Then, the optimization problem can be expressed as:

$$\hat{y} = \sum_{i=1}^{n} \alpha_i k(x_i, x) \tag{2}$$

where α is a vector of parameters. The number of parameters is equal to the number of observations used in the training sequence.

IV. Kernel Adaptive Filtering Approaches

The basic idea behind kernel adaptive filtering can be figured as follows:

1) Transform data into a high dimensional feature space

$$\phi_i = \phi(u_i) \tag{3}$$

2) Construct a linear model in the feature space H

$$y =< \omega, \phi(u) >_H \tag{4}$$

3) Adapt iteratively coefficients by gradient descent

$$\omega_i = \omega_{i-1} + \eta\ \Delta J_i \tag{5}$$

4) Compute the output

$$f_i(u) =< \omega_i, \phi(u) >_H = \sum_{j=1}^{n_i} \alpha_i k(u, c_j) \tag{6}$$

5) According to the universal approximation theorem, $f_i(u)$ can approximate any continuous training data mapping arbitrarily close in the L_p norm. Then the equation 5 will expressed as follow:

$$\omega_i = \omega_{i-1} + e(i)\ \phi(u_i) \tag{7}$$

Hence

$$f_i(u) = f_{i-1} + \eta\ e(i) k(u_i, .) \tag{8}$$

A. Kernel Mean Least Squares

The kernel least mean square (KLMS) [5] is the nonlinear version of the least mean square where the hypothesis space is the RKHS, this approach was developed in 2007. The KLMS is the simplest and one of the most knowing kernel adaptive filtering algorithms. The kernel least mean squares algorithm needs an error in order to update the filter coefficients, that's why it is an efficient algorithm dealing with online problems.

1) Principle: Unlike the least mean squares (LMS), the kernel LMS transform the input $u(i)$ from the original space into a high dimensional space H by the kernel mapping function ϕ. After that the training data will be expressed as $\{\phi(i), d(i)\}$ where $\phi(i) = \phi(u(i))$. Then the KLMS algorithm [11] is the follow:

$$\begin{cases} \omega(0) = 0 \\ e(i) = d(i) - < \omega(i-1), \phi(i) > \\ \omega(i) = \omega(i-1) + \eta\ e(i)\ \phi(i) \end{cases} \tag{9}$$

Were η is the step size, $e(i)$ is the prediction error at iteration i and $\omega(i)$ indicates the estimate of weight vector in H. Knowing that f_i is the composition of $\omega(i)$ and ϕ, ie. $f_i =< \omega(i), \phi(.) >$. According to the expression $\phi(u) = k(u, .)$, the learning rule return to the original space and the KLMS algorithm will be

$$\begin{cases} f_0 = 0 \\ e(i) = d(i) - f_{i-1}(u(i)) \\ f_i = f_{i-1} + \eta\ e(i)\ k(u(i), .) \end{cases} \tag{10}$$

This algorithm turn out an increasing Radial Basis Function (RBF) network by giving a new kernel unit for every new sample with input $u(i)$ as the center and $\eta\ e(i)$ as the coefficient, that's why it needs a procedure handles this problem.

2) Sparsification: online learning algorithms need a dictionary to store the training data to be used iteratively, the problem here is that the dictionary increase linearly with the training data resulting big amount of memory with unuseful data and high power consummation influencing to the algorithm's performance. To avoid this issue, it must discover a technique dealing with growth, able to limit the size of dictionary by obsleting the training data that doesn't satisfy to some criteria. The criteria varying from approach to anther. There is various sparcification techniques proposed in the literature, for the KLMS algorithm it is required to use the Novelty Criterion (NC) [4].

3) Novelty Criterion: The steps of this sparsification procedure are the follows:

1) Forming the dictionary

$$C(i) = \{c_j\}_{j=1}^{m_i} \tag{11}$$

A new trainig data arrives $(u(i+1), d(i+1))$

2) Compute the distance to the present dictionary

$$dis = \min_{c_j \in C} ||u(i+1) - c_j|| \tag{12}$$

3) If $dis < \delta_1$, then $u(i+1)$ will not be added into the dictionary
4) Otherwise, the prediction error is computed and only if $|e(i+1)| > \delta_2$, $u(i+1)$ will be accepted as a new center.

The δ_1 and δ_2 are two arbitrary parameters. In most cases $\delta_1 = 0.1$ and δ_2 equal to the square of mean square error. That's what we will proved next in the experiments section.

B. Kernel Recursive Least Squares

The kernel recursive least squares was proposed by Engel [6] in 2004, KRLS is the nonlinear or kernelized version of RLS, this method is an efficient online approach that find the least squares linear predictor by minimizing the weighted loss function. The KRLS algorithm produce high accuracy and has fast convergence rate, it deals with both memory and computational complexities.

1) Principle: Given a sequence of training data $D_i = \{(x_1, d_1), (x_2, d_2)...(x_i, d_i)\}$, taking from some system. In the prediction problem, the main goal is to find the best $\hat{y}$ for d_i given $D_{i-1} \cup \{x_i\}$, by online procedure where the predictor will be updated for each new training data. Then the KRLS algorithm suppose a functional form denoted by

$$\hat{y} = f(x_i) \tag{13}$$

and minimizes the loss function L

$$L = \min_f \sum_{i=1}^{N} |d_i - f(x_i)|^2 + \lambda ||f||^2 \tag{14}$$

According to the kernel trick, the functional will be as follow

$$f(x) = < w, \phi(x) > \tag{15}$$

Hence, the loss function will be:

$$L = \min_w \sum_{i=1}^{N} |d_i - < w, \phi(x) > |^2 + \lambda ||w||^2 \tag{16}$$

Then, the KRLS algorithm determine the loss function recursively and estimate the weight vector as a linear combination of $\{\phi(x_i)\}$.

$$w = \sum_{i=1}^{N} a(i)\ \phi(x_i) = \Phi_i\ a(i) \tag{17}$$

where

$$\phi_i = [\phi(x_1), \phi(x_2), ..., \phi(x_i)] \tag{18}$$

and

$$a(i) = [a(1), a(2), ..., a(i)]^T \tag{19}$$

Finally, the predictor has the following expression:

$$\hat{y} = \sum_{i=1}^{N} a_i < \phi(x_i), \phi(x_t) > = a_i\ k(x_i, x_t) \tag{20}$$

2) ALD Sparsification: The Approximate Linear Dependence (ALD) is a powerful sparsification technique, which based on sparse dictionary of input data, where the new input data will added only if it cannot be represented as a combination of other input data already stored in the dictionary. In this context, the application $\phi(x_t)$ must be approximately linearly dependent on the sequence $\{\phi(x_1),\ \phi(x_2), ..., \phi(x_{t-1})\}$, according to an approximation threshold ν, arbitrarily chosen, where the algorithm uses a projection error δ_t to obtain the sparse dictionary, this projection error must satisfy the following condition:

$$\delta_t = \min_a || \sum_{i=1}^{t-1} a_i\ \phi(x_i) - \phi(x_t)||^2 < \nu \tag{21}$$

3) Algorithm: The kernel recursive least squares algorithm with ALD sparsification can simplify by the following:

1) Initialize the sparse dictionary and the weight vector.
2) Take the first input data x_1 and put it into the dictionary. Compute the weight vector.
3) Receive the new training data (x_t, d_t). Execute the ALD test.
4) If the criterion $\delta_t < \nu$ is satisfy, then keep the dictionary unchanged.
5) If ALD test fails, then added the new input data into the dictionary. Update the weight vector.
6) Prepare the adjusted dictionary and the updated weight vector for the next iteration.

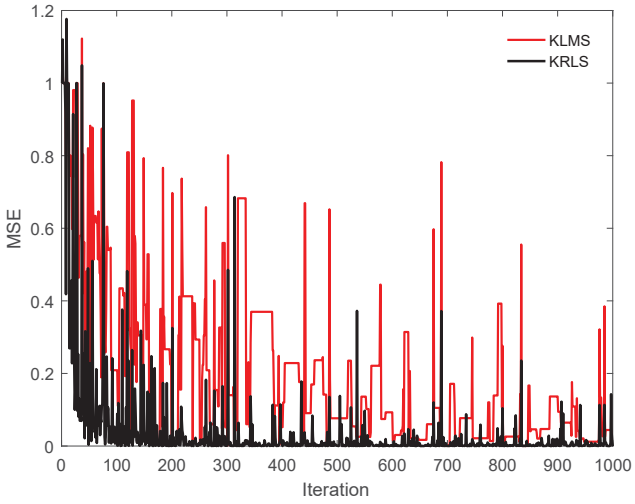

Fig. 1 Evolution of MSE for NSE

V. Experiments

Before starting a simulation, we have to clarify some points. First, the performance of the predictor is defined by the Mean Square Error (MSE), which is an estimator of the thorough deviations between predicted and original values. It is describing as follow:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - d_i)^2 \tag{22}$$

Second, the execution time of the algorithms T_e, this term depending on the simulation machine. For our works, the training data experiments were run on a Windows 10 Microsoft system with Intel(R) Core(TM) i7-4770 CPU at 3.40 GHz and 32GB of RAM. MATLAB was used to run all tests. For all algorithms, a Radial Basis Function (RBF) kernel was used:

$$k(x,y) = \exp(\frac{-||x-y||^2}{2\sigma^2}) \tag{23}$$

where σ is the kernel's parameter. Third, the validation of the estimated model to predicted one. The important challenge in modeling is how good is the predicted model? One way to check this out is to simulate it and compare the predictor output with original output. We are going to select a section of the original data (validation data) that was not used in building the predictor. Once the validation data has been pre-processed, we use the MSE to view the quality of prediction. Finally, the training vector which it can be expressed by:

$$x_i = [u(i-l), u(i-l-1), u(i-l-2), .., u(i-1)]^T \tag{24}$$

where u is the input signal and l is the input dimension. The two signal processing problems used are the Mackey Glass chaotic time series and the nonlinear channel equalization.

A. Nonlinear Channel Equalization

1) Description: The nonlinear channel equalization (NCE) is used widely to prove performance of many KAF algorithms, the NCE model contain two elements, sequence connection of linear filter and memory-less non-linearity. The process is describing by [12]. A binary signal $\{b(1), b(2), ..., b(N)\}$ is fed into a nonlinear channel. When this binary signal has been received by the channel, the signal deviated with the addition of a Gaussian noise, then it will be figured as the corresponding noisy signal $(u(1), ..., u(N))$. The goal of NCE is to form an inverse filter that reconstructs the original signal with the minimum error. The training data is the sequence:

$$x_i = \{([u(i), u(i+1), ..., u(i+l)],\ b(i-D))\} \tag{25}$$

where l is the input dimension (embedding length) and D is the equalization delay (equalization time lag). The NCE model is defined as follow

$$x(i) = b(i) + 0.5\ b(i-1) \tag{26}$$

$$u(i) = x(i) - 0.9\ x(i)^2 = n(i) \tag{27}$$

where $n(i)$ is the white Gaussian noise with a variance v.

2) Parameters: We use NCE for both KLMS and KRLS algorithm, where we choose the Radial Basis Function kernel with $\sigma = 1$, the training size is 1000 (N=1000), input dimension of 4 ($l = 4$) and the equalization delay equal to zero ($D = 0$). The variance of the white Gaussian noise is fixed to 0.1 (v = 0.1).
For the KLMS algorithm, we use the step size $\eta = 0.15$, null value of Bias and the sparsification parameters are $\delta_1 = 0.05$ and $\delta_2 = 0.01$. The KRLS specific algorithm parameters are the regularization parameter $\lambda = 0.9$, the forgetting factor $\mu = 1$ and the threshold $\nu = -1.5$. The execution times are $T_e(KLMS) = 0.03s$ and $T_e(KRLS) = 2.04s$.

3) Comparative Results: Fig. 1 show the MSE determined between the prediction output and the desired signal of both algorithms, and it obviously prove that the KRLS has the lower MSE rate and then gives the best performance for this prediction problem.

B. Mackey Glass

1) Description: The Mackey Glass chaotic time series (MG) [13] is produced by numerical integration of time delay ordinary difference equation that was proposed as a model of white blood cell production

$$\frac{du(t)}{dt} = \frac{a\ u(t-\tau)}{1+u(t-\tau)^{10}} - b\ u(t) \tag{28}$$

where $a = 0.2$, $b = 0.1$ and $\tau = 30$. Then MG will be denoted by MG30, it is discretized at a sampling time of six seconds.

2) Parameters: The Gaussian noise with variance v=0.002 and zero mean. The parameter of the used kernel is $\sigma = 1$. We use 300 samples for the training data and another 150 for testing, where the input dimension is $l = 6$, ie.

$$x(i) = [u(i-6), u(i-5), ..., u(i-1)]^T$$

The specific parameters of the KLMS algorithm are the thresholds of the sparsification procedure $\delta_1 = 0.01$ and $\delta_2 = 0.05$, the step size $\eta = 0.3$ and the Bias equal to zero. For

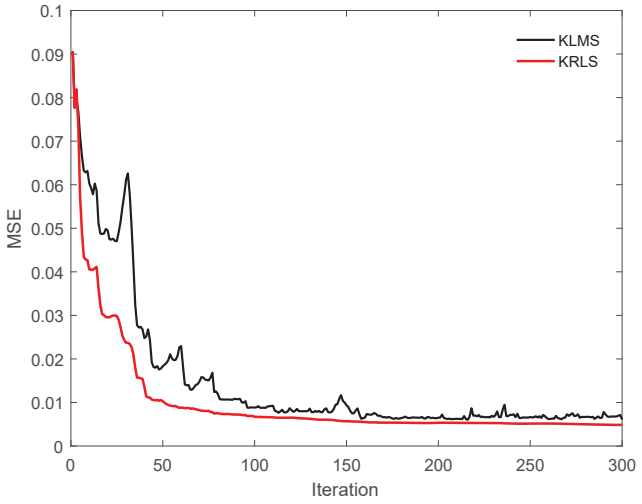

Fig. 2 Evolution of MSE for MG

KRLS, the regularization papmeter is $\lambda = 0.8$, the forgetting factor $\mu = 1$ and the sparsification threshold is $\nu = 0.2$. After running simulation, the execution time for such algorithm is $T_e(KLMS) = 0.065s$ and $T_e(KRLS) = 0.066s$.

3) Comparative results: Fig. 2 gives the MSE of both algorithms, where we can extract that the KLMS algorithm performance is close to the KRLS, but the second presents the minimum MSE values.

VI. Conclusions

Two KAF algorithms were presented KRLS and KLMS, these algorithms are efficient in online prediction problems which give a good correlation between predicted and real values and they are recommended for nonlinear signal processing. As the comparative study shows, the KLMS is the simplest, where it takes the least time execution and the KRLS more adapted in term of performance. In future study, we interest to implement some KAF methods with multiple kernel, where there are more than KRHS space.

References

[1] T. Hofmann, B. Scholkopf, and A. J. Smola, "A Tutorial Review of RKHS Methods in Machine Learning," 2005.
[2] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.
[3] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, Jul 1998.
[4] W. Liu, J. C. Principe, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*, 1st ed. Wiley Publishing, 2010.
[5] W. Liu, P. P. Pokharel, and J. C. Principe, "The kernel least-mean-square algorithm," *IEEE Transactions on Signal Processing*, vol. 56, no. 2, pp. 543–554, 2008.
[6] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, 2004.
[7] K. Fukumizu, *Elements of Positive Definite Kernel and Reproducing Kernel Hilbert Space*, 1st ed., Institute of Statistical Mathematics, ROIS, Department of Statistical Science, Graduate University for Advanced Studies, 4 2008.
[8] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995.
[9] O. Taouali, I. Elaissi, and H. Messaoud, "Dimensionality reduction of {RKHS} model parameters," {*ISA*} *Transactions*, vol. 57, pp. 205 – 210, 2015.
[10] O. Taouali, E. Elaissi, and H. Messaoud, "Design and comparative study of online kernel methods identification of nonlinear system in rkhs space," *Artificial Intelligence Review*, vol. 37, no. 4, pp. 289–300, 2011.
[11] B. Chen, S. Zhao, P. Zhu, and J. C. Principe, "Quantized kernel least mean square algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 1, pp. 22–32, 2012.
[12] D. J. Sebald and J. A. Bucklew, "Support vector machine techniques for nonlinear equalization," *IEEE Transactions on Signal Processing*, vol. 48, no. 11, pp. 3217–3226, Nov 2000.
[13] L. Glass and M. Mackey, *From Clocks to Chaos: The Rhythms of Life*, ser. Princeton paperbacks. Princeton University Press, 1988.