A SLIDING-WINDOW KERNEL RLS ALGORITHM AND ITS APPLICATION TO NONLINEAR CHANNEL IDENTIFICATION

Steven Van Vaerenbergh, Javier Vía and Ignacio Santamaría

Dept. of Communications Engineering, University of Cantabria, Spain
E-mail: {steven,jvia,nacho}@gtas.dicom.unican.es

ABSTRACT

In this paper we propose a new kernel-based version of the recursive least-squares (RLS) algorithm for fast adaptive nonlinear filtering. Unlike other previous approaches, we combine a sliding-window approach (to fix the dimensions of the kernel matrix) with conventional $L2$-norm regularization (to improve generalization). The proposed kernel RLS algorithm is applied to a nonlinear channel identification problem (specifically, a linear filter followed by a memoryless nonlinearity), which typically appears in satellite communications or digital magnetic recording systems. We show that the proposed algorithm is able to operate in a time-varying environment and tracks abrupt changes in either the linear filter or the nonlinearity.

1. INTRODUCTION

In recent years a number of kernel methods, including support vector machines [1], kernel principal component analysis [2], kernel Fisher discriminant analysis [3] and kernel canonical correlation analysis [4, 5] have been proposed and successfully applied to classification and nonlinear regression problems. In their original forms, most of these algorithms cannot be used to operate online since a number of difficulties are introduced by the kernel methods, such as the time and memory complexities (because of the growing kernel matrix) and the need to avoid overfitting of the problem.

Recently a kernel RLS algorithm was proposed that dealt with both difficulties [6]: by applying a sparsification procedure the kernel matrix size was limited and the order of the problem was reduced. In this paper we present a different approach, applying a sliding-window approach and conventional regularization. This way the size of the kernel matrix can be fixed rather than limited, allowing the algorithm to operate online in time-varying environments.

The basic idea of kernel methods is to transform the data $\mathbf{x}_i$ from the input space to a high dimensional feature space of vectors $\Phi(\mathbf{x}_i)$, where the inner products can be calculated using a positive definite kernel function satisfying Mercer's condition [1]: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$. This simple and elegant idea, also known as the "kernel trick", allows inner products in the feature space to be computed without making direct reference to the feature vectors.

A common nonlinear kernel is the Gaussian kernel

$$\kappa(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right). \quad (1)$$

This kernel will be used to calculate the elements of a *kernel matrix*. In the sliding-window approach, updating this kernel matrix means first removing its first row and first column and then adding a new row and column at the end, based on new observations. Calculation of its inverse is needed to update the algorithm's solution. To this end, two matrix inversion formulas were derived in Appendix A. One of them has already been used in kernel methods [7] but in order to fix the dimensions of the kernel matrix we also introduce a complementary formula.

The rest of this paper is organized as follows: in Section 2 a kernel transformation is introduced into linear least-squares regression theory. A detailed description of the proposed algorithm is given in Section 3, and in Section 4 it is applied to a nonlinear channel identification problem. Finally, Section 5 summarizes the main conclusions of this work.

2. LEAST-SQUARES REGRESSION

2.1. Linear Methods

The least-squares (LS) criterion [8] is a widely used method in signal processing. Given a vector $\mathbf{y} \in \mathbb{R}^{N\times 1}$ and a data matrix $\mathbf{X} \in \mathbb{R}^{N\times M}$ of observations, it consists in seeking the optimal vector $\mathbf{h} \in \mathbb{R}^{M\times 1}$ that solves

$$J = \min_{\mathbf{h}} \|\mathbf{y} - \mathbf{X}\mathbf{h}\|^2. \quad (2)$$

It should be clear that the solution $\mathbf{h}$ can be represented in the basis defined by the rows of $\mathbf{X}$. Hence it can also be written as $\mathbf{h} = \mathbf{X}^T\mathbf{a}$, making it a linear combination of the input patterns (this is sometimes denoted as the "dual representation").

For many problems however, not all data are known in advance and the solution has to be re-calculated as the new observations become available. An online algorithm is then needed, which in case of linear problems is given by the well-known recursive least-squares (RLS) algorithm [8].

This work was supported by MEC (Ministerio de Educación y Ciencia) under grant TEC2004-06451-C05-02.

2.2. Kernel Methods

The linear LS methods can be extended to nonlinear versions by transforming the data into a feature space. Using the transformed vector $\tilde{\mathbf{h}} \in \mathbb{R}^{M' \times 1}$ and the transformed data matrix $\tilde{\mathbf{X}} \in \mathbb{R}^{N \times M'}$, the LS problem (2) can be written in feature space as

$$J' = \min_{\tilde{\mathbf{h}}} \|\mathbf{y} - \tilde{\mathbf{X}}\tilde{\mathbf{h}}\|^2. \tag{3}$$

The transformed solution $\tilde{\mathbf{h}}$ can now also be represented in the basis defined by the rows of the (transformed) data matrix $\tilde{\mathbf{X}}$, namely as

$$\tilde{\mathbf{h}} = \tilde{\mathbf{X}}^T \boldsymbol{\alpha}. \tag{4}$$

Moreover, introducing the *kernel matrix* $\mathbf{K} = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^T$ the LS problem in feature space (3) can be rewritten as

$$J' = \min_{\boldsymbol{\alpha}} \|\mathbf{y} - \mathbf{K}\boldsymbol{\alpha}\|^2 \tag{5}$$

in which the solution $\boldsymbol{\alpha}$ is an $N \times 1$ vector. The advantage of writing the nonlinear LS problem in the dual notation is that thanks to the "kernel trick", we only need to compute $\mathbf{K}$, which is done as

$$\mathbf{K}(i,j) = \kappa(\mathbf{X}_i, \mathbf{X}_j), \tag{6}$$

where $\mathbf{X}_i$ and $\mathbf{X}_j$ are the i-th and j-th rows of $\mathbf{X}$. As a consequence the computational complexity of operating in this high-dimensional space is not necessarily larger than that of working in the original low-dimensional space.

2.3. Measures Against Overfitting

For most useful kernel functions, the dimension of the feature space, M', will be much higher than the number of available data points N (for instance, in case the Gaussian kernel is used the feature space will have dimension $M' = \infty$). In these cases, Eq. (5) could have an infinite number of solutions, representing an overfit problem.

Various techniques to handle this overfitting have been presented. One possible method is to reduce the order of the feature space [6, 4, 5]. A second method, used here, is to regularize the solution. More specifically, the norm of the solution $\tilde{\mathbf{h}}$ is penalized to obtain the following problem:

$$\begin{aligned} J'' &= \min_{\tilde{\mathbf{h}}} \|\mathbf{y} - \tilde{\mathbf{X}}\tilde{\mathbf{h}}\|^2 + c\|\tilde{\mathbf{h}}\|^2 & (7)\\ &= \min_{\boldsymbol{\alpha}} \|\mathbf{y} - \mathbf{K}\boldsymbol{\alpha}\|^2 + c\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha} & (8) \end{aligned}$$

whose solution is given by

$$\boldsymbol{\alpha} = \mathbf{K}_{reg}^{-1}\mathbf{y} \tag{9}$$

with $\mathbf{K}_{reg} = (\mathbf{K} + c\mathbf{I})$, c a regularization constant and $\mathbf{I}$ the identity matrix.

3. THE ONLINE ALGORITHM

In various situations it is preferred to have an online, i.e. recursive, version instead of a batch algorithm. In particular, if the data points $\mathbf{y}$ are the result of a time-varying process, an online algorithm able to track these time variations can be designed. In any case, the key feature of an online algorithm is that the number of computations required per new sample must not increase as the number if samples increases.

3.1. A Sliding-Window Approach

The presented algorithm is a regularized kernel version of the RLS algorithm. An online prediction setup assumes we are given a stream of input-output pairs $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots\}$. The sliding-window approach consists in only taking the last N pairs of this stream into account. For window n, the observation vector $\mathbf{y}_n = [y_n, y_{n-1}, \dots, y_{n-N+1}]^T$ and observation matrix $\mathbf{X}_n = [\mathbf{x}_n, \mathbf{x}_{n-1}, \dots, \mathbf{x}_{n-N+1}]^T$ are formed, and the corresponding regularized kernel matrix $\mathbf{K}_n = \tilde{\mathbf{X}}_n\tilde{\mathbf{X}}_n^T + c\mathbf{I}$ can be calculated.

Note that it is necessary to limit the number of data vectors $\mathbf{x}_n$, N, for which the kernel matrix is calculated. Contrary to standard linear RLS, for which the correlation matrices have fixed sizes depending on the (fixed) *dimension of the input vectors* M, the size of the kernel matrix in an online scenario depends on the *number of observations* N.

In [6], a kernel RLS algorithm is designed that limits the matrix sizes by means of a sparsification procedure, which maps the samples to a (limited) dictionary. It allows both to reduce the order of the feature space (which prevents overfitting) and to keep the complexity of the algorithm bounded. In our approach these two measures are obtained by two different mechanisms. On one hand, the regularization against overfitting is done by penalizing the solutions, as in (9). On the other hand, the complexity of the algorithm is reduced by considering only the observations in a window with fixed length. The advantage of the latter approach is that it is able to track time variations without any extra computational burden.

3.2. Updating the Inverse of the Kernel Matrix

The calculation of the updated solution $\boldsymbol{\alpha}_n$ requires the calculation of the $N \times N$ inverse matrix $\mathbf{K}_n^{-1}$ for each window. This is costly both computationally and memory-wise (requiring $O(N^3)$ operations). Therefore an update algorithm is developed that can compute $\mathbf{K}_n^{-1}$ solely from knowledge of the data of the current window $\{\mathbf{X}_n, \mathbf{y}_n\}$ and the previous $\mathbf{K}_{n-1}^{-1}$. The updated solution $\boldsymbol{\alpha}_n$ can then be calculated in a straightforward way using Eq. (9).

Given the regularized kernel matrix $\mathbf{K}_{n-1}$, the new regularized kernel matrix $\mathbf{K}_n$ can be constructed by removing the first row and column of $\mathbf{K}_{n-1}$, referred to as $\hat{\mathbf{K}}_{n-1}$, and adding kernels of the new data as the last row and column:

$$\mathbf{K}_n = \begin{bmatrix} \hat{\mathbf{K}}_{n-1} & \mathbf{k}_{n-1}(\mathbf{x}_n) \\ \mathbf{k}_{n-1}(\mathbf{x}_n)^T & k_{nn} + c \end{bmatrix} \tag{10}$$

Algorithm 1 Summary of the proposed adaptive algorithm.

Initialize $\mathbf{K}_0$ as $(1+c)\mathbf{I}$ and $\mathbf{K}_0^{-1}$ as $\mathbf{I}/(1+c)$
for $n = 1, 2, \ldots$ **do**
  Obtain $\hat{\mathbf{K}}_{n-1}$ out of $\mathbf{K}_{n-1}$
  Calculate $\hat{\mathbf{K}}_{n-1}^{-1}$ according to Eq. (12)
  Obtain $\mathbf{K}_n$ according to Eq. (10)
  Calculate $\mathbf{K}_n^{-1}$ according to Eq. (11)
  Obtain the updated solution $\boldsymbol{\alpha}_n = \mathbf{K}_n^{-1}\mathbf{y}_n$
end for

where $\mathbf{k}_{n-1}(\mathbf{x}_n) = [\kappa(\mathbf{x}_{n-N+1}, \mathbf{x}_n), \ldots, \kappa(\mathbf{x}_{n-1}, \mathbf{x}_n)]^T$ and $k_{nn} = \kappa(\mathbf{x}_n, \mathbf{x}_n)$.

Calculating the inverse kernel matrix $\mathbf{K}_n^{-1}$ is done in two steps, using the two inversion formulas derived in appendices A.1 and A.2. Note that these formulas do not calculate the inverse matrices explicitly, but rather derive them from known matrices maintaining an overall time and memory complexity of $O(N^2)$ of the algorithm.

First, given $\mathbf{K}_{n-1}$ and $\mathbf{K}_{n-1}^{-1}$, the inverse of the $N-1 \times N-1$ matrix $\hat{\mathbf{K}}_{n-1}$ is calculated according to Eq. (12). Then $\mathbf{K}_n^{-1}$ can be calculated applying the matrix inversion formula from Eq. 11, based on the knowledge of $\hat{\mathbf{K}}_{n-1}^{-1}$ and $\mathbf{K}_n$. The complete algorithm is summarized in Alg. (1).

4. EXAMPLE PROBLEM: IDENTIFICATION OF A NONLINEAR WIENER SYSTEM

In this section, we consider the identification problem for a nonlinear Wiener system, and compare the performance of the proposed kernel RLS algorithm to the standard approach using a multilayer perceptron (MLP). Since kernel methods provide a natural nonlinear extension of linear regression methods, the proposed system is supposed to perform well compared to the MLP [7].

4.1. Experimental Setup

The nonlinear Wiener system is a well-known and simple nonlinear system which consists of a series connection of a linear filter and a memoryless non-linearity (see Fig. 1). Such a nonlinear channel can be encountered in digital satellite communications [9] and in digital magnetic recording [10]. Traditionally, the problem of blind nonlinear equalization or identification has been tackled by considering nonlinear structures such as MLPs [11], recurrent neural networks [12], or piecewise linear networks [13].

Here we consider a supervised identification problem, in which moreover at a given time instant the linear channel coefficients are changed abruptly to compare the tracking capabilities of both algorithms: During the first part of the simulation, the linear channel is $H_1(z) = 1 + 0.0668z^{-1} - 0.4764^{-2} + 0.8070^{-3}$ and after receiving 500 symbols it is changed into $H_2(z) = 1 - 0.4326z^{-1} - 0.6656z^{-2} + 0.7153^{-3}$. A binary signal is sent through this channel and then the nonlinear function $y = \tanh(x)$ is applied on it, where x is the linear channel output. Finally, white Gaussian noise is added to match an SNR of 20dB. The Wiener system is then treated as a black box of which only input and output are known.

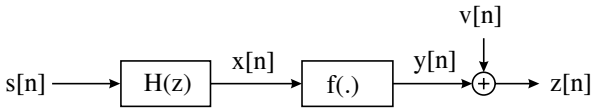

Fig. 1. A nonlinear Wiener system.

4.2. Simulation Results

System identification was first performed by an MLP with 8 neurons in its hidden layer and then using the sliding-window kernel RLS algorithm, for two different window sizes N. For both methods we applied time-embedding techniques in which the length L of the linear channel was known. More specifically, the used MLP was a time-delay MLP with L inputs, and the input vectors for the kernel RLS algorithm were time-delayed vectors of length L, $\mathbf{s}(n) = [\mathbf{s}(n-L+1), \ldots, \mathbf{s}(n)]$. In each iteration, system identification was performed by estimating the output sample corresponding to the next input sample, and comparing it to the actual output. The mean square error (MSE) for both approaches is shown in Fig. 2. Most noticeable is the fast convergence of the kernel RLS algorithm: convergence time is of the order of the window length.

Further note that the structure of the nonlinear system has not been exploited while performing identification. Obviously, the presented kernel RLS method can be extended and used as the basis of a more complex algorithm that models better the known system structure. For instance, the solution to the nonlinear Wiener identification problem could be found as the solution to two coupled LS problems, where the first one applies a linear kernel on the input data and the second one applies a nonlinear kernel on the output data. Also, the implications of not knowing the correct linear channel length remain to be studied. We will consider this and other extensions as future research lines.

5. CONCLUSIONS

A kernel-based version of the RLS algorithm was presented. Its main features are the introduction of regularization against overfitting (by penalizing the solutions) and the combination of a sliding-window approach and efficient matrix inversion formulas to keep the complexity of the problem bounded. Thanks to the use of a sliding-window the algorithm is able to provide tracking in a time-varying environment.

First results of this algorithm are promising, and suggest it can be extended to deal with the nonlinear extensions of most problems that are classically solved by linear RLS. Future research lines also include its direct application to online kernel canonical correlation analysis (kernel CCA).

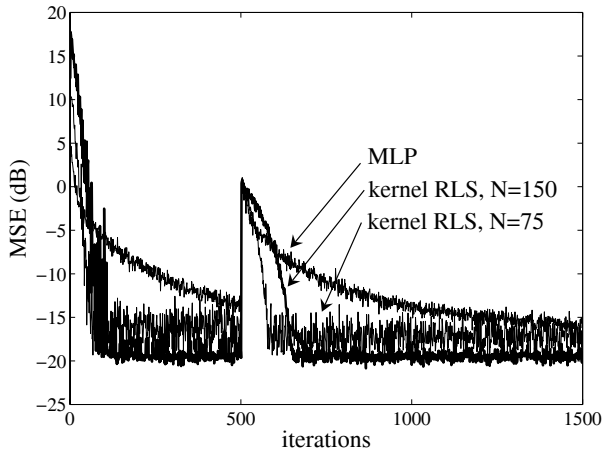

Fig. 2. MSE of the identification of the nonlinear Wiener system of Fig. 1, for the standard method using an MLP and for the window-based kernel RLS algorithm with window length $N = 150$ (thick curve) and $N = 75$ (thin curve). A change in filter coefficients of the nonlinear Wiener system was introduced after 500 iterations. The results were averaged out over 250 Monte-Carlo simulations.

6. REFERENCES

[1] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag New York, Inc., New York, USA, 1995.

[2] 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.

[3] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller, "Fisher discriminant analysis with kernels," in *Proc. NNSP'99*, Y.-H Hu, J. Larsen, E. Wilson, and S. Douglas, Eds. Jan. 1999, pp. 41–48, IEEE.

[4] F. R. Bach and M. I. Jordan, "Kernel independent component analysis," *Journal of Machine Learning Research*, vol. 3, pp. 1–48, 2003.

[5] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor, "Canonical correlation analysis: An overview with application to learning methods," Technical Report CSD-TR-03-02, Royal Holloway University of London, 2003.

[6] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least squares-algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, Aug. 2004.

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

[8] A.H. Sayed, *Fundamentals of Adaptive Filtering*, Wiley, New York, USA, 2003.

[9] G. Kechriotis, E. Zarvas, and E. S. Manolakos, "Using recurrent neural networks for adaptive communication channel equalization," *IEEE Trans. on Neural Networks*, vol. 5, pp. 267–278, Mar 1994.

[10] N. P. Sands and J. M. Cioffi, "Nonlinear channel models for digital magnetic recording," *IEEE Trans. Magn.*, vol. 29, pp. 3996–3998, Nov 1993.

[11] D. Erdogmus, D. Rende, J. C. Principe, and T. F. Wong, "Nonlinear channel equalization using multilayer perceptrons with information-theoric criterion," in *Proc. IEEE Workshop on Neural Networks and Signal Processing XI*, Falmouth, MA, Sept 2001, pp. 401–451.

[12] T. Adali and X. Liu, "Canonical piecewise linear network for nonlinear filtering and its application to blind equalization," *Signal Process.*, vol. 61, no. 2, pp. 145–155, Sept 1997.

[13] P. W. Holland and R. E. Welch, "Robust regresison using iterative reweighted least squares," *Commun. Statist. Theory Methods*, vol. A. 6, no. 9, pp. 813–827, 1997.

A. MATRIX INVERSION FORMULAS

A.1. Adding a row and a column

To a given non-singular matrix $\mathbf{A}$ a row and column are added as shown below, resulting in matrix $\mathbf{K}$. The inverse matrix $\mathbf{K}^{-1}$ can then be expressed in terms of the known elements and $\mathbf{A}^{-1}$ as follows:

$$\mathbf{K} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{b}^T & d \end{bmatrix}, \quad \mathbf{K}^{-1} = \begin{bmatrix} \mathbf{E} & \mathbf{f} \\ \mathbf{f}^T & g \end{bmatrix}$$

$$\Rightarrow \begin{cases} \mathbf{A}\mathbf{E} + \mathbf{b}\mathbf{f}^T &= \mathbf{I} \\ \mathbf{A}\mathbf{f} + \mathbf{b}g &= \mathbf{0} \\ \mathbf{b}^T\mathbf{f} + dg &= 1 \end{cases}$$

$$\Rightarrow \mathbf{K}^{-1} = \begin{bmatrix} \mathbf{A}^{-1}(\mathbf{I} + \mathbf{b}\mathbf{b}^T\mathbf{A}^{-1H}g) & -\mathbf{A}^{-1}\mathbf{b}g \\ -(\mathbf{A}^{-1}\mathbf{b})^T g & g \end{bmatrix} \tag{11}$$

with $g = (d - \mathbf{b}^T\mathbf{A}^{-1}\mathbf{b})^{-1}$.

A.2. Removing the first row and column

From a given non-singular matrix $\mathbf{K}$ a row and column are removed as shown below, resulting in matrix $\mathbf{D}$. The inverse matrix $\mathbf{D}^{-1}$ can then easily be expressed in terms of the known elements of $\mathbf{K}^{-1}$ as follows:

$$\mathbf{K} = \begin{bmatrix} a & \mathbf{b}^T \\ \mathbf{b} & \mathbf{D} \end{bmatrix}, \quad \mathbf{K}^{-1} = \begin{bmatrix} e & \mathbf{f}^T \\ \mathbf{f} & \mathbf{G} \end{bmatrix}$$

$$\Rightarrow \begin{cases} \mathbf{b}e + \mathbf{D}\mathbf{f} &= \mathbf{0} \\ \mathbf{b}\mathbf{f}^T + \mathbf{D}\mathbf{G} &= \mathbf{I} \end{cases}$$

$$\Rightarrow \mathbf{D}^{-1} = \mathbf{G} - \mathbf{f}\mathbf{f}^T/e. \tag{12}$$