A Kernel Adaptive Filter based on ERLS-DCD Algorithm

Yoshiki OGAWA and Kiyoshi NISHIKAWA
Dept. Inform. & Commun. Syst., Tokyo Metropolitan Univ., Hino, Tokyo 191–0065, Japan
ogawa-yoshiki@sd.tmu.ac.jp, knishikawa@m.ieice.org

Abstract

In this paper, we propose a new kernel adaptive algorithm that is based on the ERLS-DCD algorithm. A conventional linear RLS algorithm is known to have a good convergence characteristics property while require a high computational complexity. Recently, the kernel-RLS algorithm is proposed for learning non-linear systems. However, the kernel-RLS requires more computation for learning, and this computational load is one of problems to implement it. We propose an algorithm that provides almost equivalent convergence property to the RLS, and almost equivalent computational complexity to the LMS. We show that proposed algorithm can improve the convergence characteristics compared to the conventional kernel adaptive filters through simulations.

keywords: Kernel Method, Adaptive Filter, ERLS-DCD

1. Introduction

In this paper, we propose a new kernel adaptive algorithm that is based on the ERLS-DCD algorithm.

A linear adaptive filter estimates parameters of an unknown system using its input and output signals. Adaptive filters have been used in various applications such as channel equalization in wireless digital communications [1].

An algorithm for adaptive learning is called an adaptive algorithm; LMS(least mean square) and RLS(recursive least square) are known as representative ones. However, the systems in wireless communication often have nonlinearity, it is shown that performance can be improved by using non-linear adaptive filters [2]. As a non-linear adaptive filter, several methods were proposed based on the kernel adaptive filters. By applying kernel method, it is enabled to learn non-linear systems using linear adaptive algorithm [2] [4] [5].

In this paper, we propose the kernel-ERLS-DCD adaptive algorithm that is an extended version of the linear adaptive algorithm, ERLS-DCD (exponentially weighted recursive least squares - dichotomous coordinate descent). The proposed method provides almost equivalent convergence property to the RLS, and almost equivalent computational complexity to the LMS. We derive the proposed algorithm and show that it improves the convergence characteristics compared to the conventional kernel adaptive filters through simulations.

In Sec.2, we describe conventional adaptive filters and kernel method. In Sec.3, we present a proposed algorithm. In Sec.4, we present simulation results to show validity. Finally, we give conclusions.

2. Preparation

Here, we briefly describe adaptive filters and the linear ERLS-DCD algorithm which we extend to the non-linear estimations.

2.1. Adaptive filter

Adaptive filters estimate the parameters of an unknown system from its input and output signals adaptively. We show a typical structure of an adaptive filter in Fig.1. In

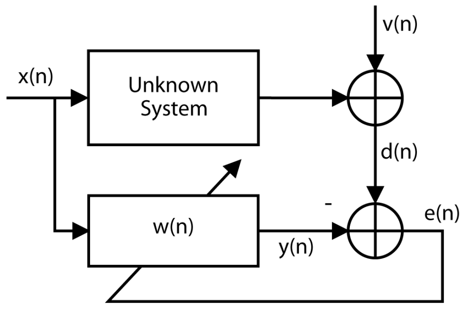

Figure 1: Structure of adaptive filter

this figure, $x(n)$, $d(n)$, $v(n)$, $y(n)$, $e(n)$ are input, desired, additive noise, output and error signals at time n respectively. The error signal $e(n)$ is defined as

$$\begin{aligned} e(n) &= d(n) - y(n) = d(n) - \mathbf{x}(n)^{\mathrm{T}}\mathbf{w}(n) && (1)\\ \mathbf{x}(n) &= [x(n)\ x(n-1)\ \cdots\ x(n-M-1)]^{\mathrm{T}} && (2)\\ \mathbf{w}(n) &= [w_0(n)\ w_1(n)\ \cdots\ w_{M-1}(n)]^{\mathrm{T}} && (3)\end{aligned}$$

where $\mathbf{w}(n)$ is the filter coefficient vector of adaptive filter, M denotes filter length and T denotes the transpose of a vector. An adaptive algorithm updates $w(n)$ in order to minimize the mean of $e(n)$ (or sum of squares of $e(n)$) recurrently.

2.2. ERLS-DCD algorithm [3]

The RLS algorithm minimizes the sum of squared errors using the following equation in order to learn unknown systems.

$$\varphi(n) = \sum_{n=1}^{N} \lambda^{N-n} e(n)^2 \qquad (4)$$

where λ is the forgetting factor that is a parameter to track a system variation. It is known to maintain good steadystate performance and convergence performance simultaneously. Now, we get the following equation called the normal equation through using eq.(1) in eq.(4).

$$\mathbf{R}(n)\hat{\mathbf{w}}(n) = \boldsymbol{\beta}(n) \tag{5}$$

where $\mathbf{R}$ is the auto-correlation matrix of $x(n)$ of size $(M+1)\times(M+1)$, β is $M+1$-length cross-correlation vector between $x(n)$ and $d(n)$. The standard RLS using matrix inversion lemma requires complexity $O(M^2)$ to obtain the solution [3].

On the other hand, an alternative method was proposed which approximates $\hat{\mathbf{w}}$ by using the following equations to reduce computation [3].

$$\begin{cases} \Delta\mathbf{R}(n) = \mathbf{R}(n) - \mathbf{R}(n-1) \\ \Delta\boldsymbol{\beta}(n) = \boldsymbol{\beta}(n) - \boldsymbol{\beta}(n-1) \\ \Delta\mathbf{w}(n) = \mathbf{w}(n) - \hat{\mathbf{w}}(n-1) \end{cases} \tag{6}$$

Then, expressions of updating a filter given by the following:

```
Initialize: $\hat{\mathbf{w}}(-1) = 0$,  $\mathbf{r}(-1) = 0$,  $\mathbf{R}(-1) = \mathbf{\Pi}$
  for $n = 0, 1, 2, \cdots$
    $\mathbf{R}(n) = \lambda\mathbf{R}(n-1) + \mathbf{x}(n)^{\mathrm{T}}\mathbf{x}(n)$    (7)
    $e(n) = d(n) - \mathbf{x}(n)^{\mathrm{T}}\hat{\mathbf{w}}(n-1)$    (8)
    $\boldsymbol{\beta}_0(n) = \lambda\mathbf{r}(n-1) + e(n)\mathbf{x}(n)$    (9)
    $\mathbf{R}(n)\Delta\mathbf{w}(n) = \boldsymbol{\beta}_0(n) \Rightarrow \Delta\hat{\mathbf{w}}(n), \mathbf{r}(n)$    (10)
    $\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \Delta\hat{\mathbf{w}}(n)$    (11)
```

where $\mathbf{\Pi}$, η are $\mathbf{\Pi} = \eta\mathbf{I}_N$, a small positive number and $\mathbf{I}_N$ denotes $N\times N$ identity matrix. (10) is called auxiliary normal equations that is written as the following using residual vectors $\mathbf{r}(n)$.

$$\begin{cases} \beta_0(n) = \mathbf{r}(n-1) + \Delta\beta_0(n) - \Delta\mathbf{R}(n)\hat{\mathbf{w}}(n-1) \\ \mathbf{r}(n-1) = \beta_0(n-1) - \mathbf{R}(n-1)\hat{\mathbf{w}}(n-1) \\ \hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \Delta\hat{\mathbf{w}}(n) \end{cases} \tag{12}$$

In this paper, we use the DCD algorithm [3] given by the following to solve the auxiliary normal equations.

```
Initialize: $\Delta\hat{\mathbf{w}} = 0$,  $\mathbf{r} = \beta_0$,  $\alpha = 1$,  $q = 0$
for $m = 1, \cdots, M_b$
  $\alpha = \alpha/2$    (13)
  for $n = 1, \cdots, N$    (14)
    if $|r(n)| > (\alpha/2)R$ then    (15)
      flag $= 1$,  $q = q + 1$    (16)
      $\Delta\hat{\mathbf{w}}(n) = \Delta\hat{\mathbf{w}}(n) + \mathrm{sign}(r(n))\alpha$    (17)
      $\mathbf{r} = \mathbf{r} - \mathrm{sign}(r(n))\alpha\mathbf{R}$    (18)
    if $q > N_u$ then the algorithm stops
  if flag $= 1$ then flag $= 0$ and goto (14)
```

Using the DCD algorithm, we can reduce computation by $O(M)$.

2.3. Kernel method

Kernel method is a non-linear mapping from the input space to the feature space to process by inner product. When vectors $\mathbf{u}$ and $\mathbf{u}'$ are mapped to the feature space $\mathbf{H}$ by nonlinear mapping ϕ, we denote them as $\phi(\mathbf{u})$ and $\phi(\mathbf{u}')$ respectively. Let us select a function κ that is defined by

$$\phi(\mathbf{u})^{\mathrm{T}} \cdot \phi(\mathbf{u}') = \kappa(\mathbf{u}, \mathbf{u}') \tag{19}$$

where $\cdot$ denotes the inner product, κ is called a kernel function. A kernel function allows us to calculate an inner product of vectors in the feature space. A kernel function need to be a symmetry, positive definite [1]. Gaussian kernel, Linear kernel, Polynomial kernel are representative kernel functions. Kernel adaptive filters are implemented to replace the inner product in linear adaptive algorithms with a kernel function (kernel trick) [2]. Kernel-RLS algorithm [4], kernel-NLMS algorithm [5] are proposed so far.

3. Proposed Method

Here, we propose a kernel adaptive filter based on the ERLS-DCD algorithm. Conventional kernel adaptive filters are derived by replacing the inner product with a kernel function using previous input vectors. Similarly, the proposed algorithm updates the filter using the ERLS-DCD algorithm with some modification.

3.1. Application for the Kernel Method

The filter coefficient vector $\mathbf{w}'(n)$ for transformed inputs is able to be written by

$$\mathbf{w}'(n) = \alpha_1\phi(\mathbf{u}(1)) + \cdots + \alpha_n\phi(\mathbf{u}(n-1)) \tag{20}$$

where α_n is the corresponding coefficient. Then, the output $y(n)$ is rewritten as below.

$$\begin{aligned} &y(n) \\ &= \phi(\mathbf{x}(n))^{\mathrm{T}}\mathbf{w}'(n) && (21) \\ &= \phi(\mathbf{x}(n))^{\mathrm{T}}(\alpha_1\phi(\mathbf{x}(1)) + \cdots + \alpha_n\phi(\mathbf{x}(n-1))) && (22) \\ &= \alpha_1\phi(\mathbf{x}(n))^{\mathrm{T}}\phi(\mathbf{x}(1)) + \cdots + \alpha_n\phi(\mathbf{x}(n))^{\mathrm{T}}\phi(\mathbf{x}(n-1)) && (23) \\ &= \alpha_1\kappa(\mathbf{x}(n), \mathbf{x}(1)) + \cdots + \alpha_n\kappa(\mathbf{x}(n), \mathbf{x}(n-1)) && (24) \end{aligned}$$

Here, by defining the vector

$$\mathbf{h} = [\kappa(\mathbf{x}(n), \mathbf{x}(1)) \ \cdots \ \kappa(\mathbf{x}(n), \mathbf{x}(n-1))]^{\mathrm{T}}, \tag{25}$$

we get $y(n) = \mathbf{h}^{\mathrm{T}}\boldsymbol{\alpha}$ that is the same form used in the linear adaptive filter assuming that $\mathbf{h}$ is the input vector. We will consider to apply the ERLS-DCD algorithm for updating $\boldsymbol{\alpha}$.

3.2. Variation of Filter order

As continuing an estimation, the dimension of feature space will be increase, and formula of ERLS-DCD algorithm are needed to be adjusted. It means that infinite

[1] $\forall a_i, a_j \in \mathbf{R}, \sum_{i,j=1}^{\infty} a_i a_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \geq 0$

caluclations would be required. Accordingly, we make a choice based on the threshold, and make a set of the selected input vectors as **D** (we call it dictionary). If **h** is compared with the threshold value μ_0 as the following equation, and if the condition will would be satisfied, we assume that no similar vectors are in the dictionary, and the input vector will be added to the dictionary.

$$\max|\mathbf{h}_j|_{j=1,\cdots,m} < \mu_0 \tag{26}$$

where μ_0 is the threshold determined by the kernel function. Moreover, filter orders are increased as dictionary **D** updated, so we should modify ERLS-DCD algorithm according to this oparation.

The proposed algorithm is summarized as below.

```
Initialize:
    $\mathbf{D}(-1) = \{x_0\},\ \boldsymbol{\alpha}(-1) = \mathbf{r}(-1) = 0,\ \mathbf{R}(-1) = \boldsymbol{\Pi}$
for $n = 0, 1, 2, \cdots$
    $\mathbf{h}(n) = \kappa(\mathbf{x}(n), \mathbf{D}(n-1))$    (27)
    if $\max|\mathbf{h}(n)_j|_{j=1,\cdots,m} > \mu_0$    (28)
        $\mathbf{D}(n) = \mathbf{D}(n-1)$    (29)
        $\mathbf{R}(n) = \lambda\mathbf{R}(n-1) + \mathbf{h}(n)^{\mathrm{T}}\mathbf{h}(n)$    (30)
        $e(n) = d(n) - \mathbf{h}(n)^{\mathrm{T}}\boldsymbol{\alpha}(n-1)$    (31)
        $\boldsymbol{\beta}_0 = \lambda\mathbf{r}(n-1) + e(n)\mathbf{h}(n)$    (32)
        $(\Delta\boldsymbol{\alpha}(n), \mathbf{r}) = \mathrm{DCD}(\mathbf{R}(n), \boldsymbol{\beta}_0)$    (33)
        $\boldsymbol{\alpha}(n) = \boldsymbol{\alpha}(n-1) + \Delta\boldsymbol{\alpha}(n)$    (34)
    else
        $\mathbf{D}(n) = \mathbf{D}(n-1) \cup \{x(n)\}$    (35)
        $\mathbf{h}(n) = \begin{bmatrix} \mathbf{h}_n \\ 1 \end{bmatrix}$    (36)
        $\mathbf{R}(n) = \lambda \begin{bmatrix} \mathbf{R}(n-1) & 0 \\ 0 & 1 \end{bmatrix} + \mathbf{h}(n)^{\mathrm{T}}\mathbf{h}(n)$    (37)
        $e(n) = d(n) - \mathbf{h}(n)^{\mathrm{T}} \begin{bmatrix} \boldsymbol{\alpha}(n-1) \\ 0 \end{bmatrix}$    (38)
        $\boldsymbol{\beta}_0 = \lambda \begin{bmatrix} \mathbf{r}(n-1) \\ 0 \end{bmatrix} + e(n)\mathbf{h}(n)$    (39)
        $(\Delta\boldsymbol{\alpha}(n), \mathbf{r}) = \mathrm{DCD}(\mathbf{R}(n), \boldsymbol{\beta}_0)$    (40)
        $\boldsymbol{\alpha}(n) = \begin{bmatrix} \boldsymbol{\alpha}(n-1) \\ 0 \end{bmatrix} + \Delta\boldsymbol{\alpha}(n)$    (41)
```

where $\mathbf{h}(n)$ is non-linear mapped input, $\boldsymbol{\alpha}(n)$ is the filter cofficients, μ_0 is threshold, $\mathbf{D}(n)$ is dictionary, DCD() is that solve normal equations by DCD algorithm. Compared with the linear ERLS-DCD algorithm, the proposed algorithm is different in eq.(36)-(39) and (41) because of variation of the dictionary.

4. Simulation Results

Here, we present simulation results that show a comparison of performances of the proposed algorithm and the conventional kernel adaptive filters.

4.1. Forward Prediction

We simulated adaptive prediction of a non-linear system that is given by the following equation [6].

$$\begin{aligned} d_n =& 0.1\sin(d_{n-1}\pi) + \left(0.8 - 0.5\exp\left(-d_{n-1}^2\right)\right)d_{n-1} \\ & - \left(0.3 + 0.9\exp\left(-d_{n-1}^2\right)\right)d_{n-2} \end{aligned} \tag{42}$$

Initial state (d_{-1}, d_{-2}) was given by random values between $[0, 1]$. We added a white Gaussin noise of SNR = 40[dB] to the generated signal, and evaluated in terms of the MSE. Signal length was 12000 and $M = 4$. When n = 4000, coefficients are changed to random value to occur system changing. We use Gaussian kernel given by the following as kernel function.

$$\kappa(\mathbf{a}, \mathbf{b}) = \exp\left(-\|\mathbf{a} - \mathbf{b}\|^2\right) \tag{43}$$

Values of parameters were set as $\lambda = 0.995$, $M_b = 16$, $N_u = 1$, $\mu_0 = 0.8$.

Fig.2 shows the MSE performances of kernel-NLMS, kernel-RLS and the proposed method. The Proposed method has as fine convergence property as kernel-RLS. Fig.3 shows the processing time. We can see that processing time for the proposed method is approximately 10 percent less than that of kernel-RLS.

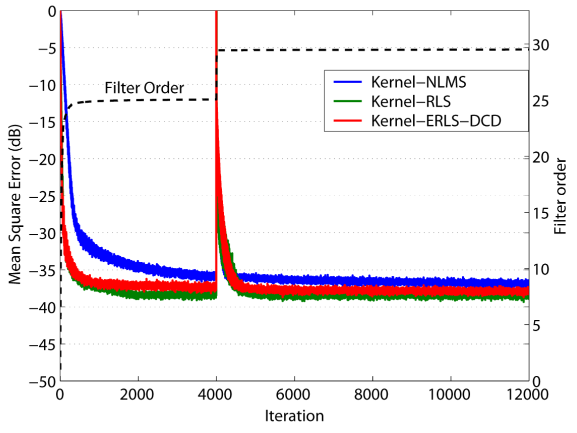

Figure 2: MSE performances of adaptive prediction

4.2. Reyreigh Channel Tracking

We also simulated tracking of a Rayleigh fading multipath channel [1] [4]. We choose parameters as the number of paths $M = 5$, the maximum Doppler frequency $f_D = 100Hz$, sampling rate $T_s = 0.8\mu s$. Signal length was 1000. When n = 500, coefficients of paths are changed to random values to occur system changing.

Fig.4 shows the MSE performances of kernel-NLMS, kernel-RLS and the proposed method. The Proposed method provides almost same convergence property as that of the kernel-RLS as well as the simulation in 4.1. Fig.5 shows the processing time. We can see that the proposed method has a stable speed of processing similar to kernel-NLMS

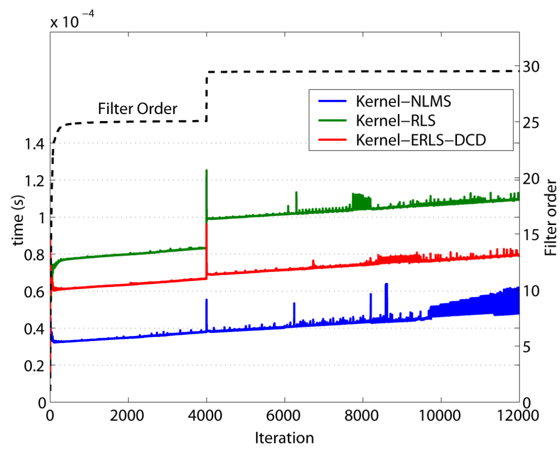

Figure 3: Processing times of adaptive prediction

in spite of the fact that a processing time of kernel-RLS has increased with filter order.

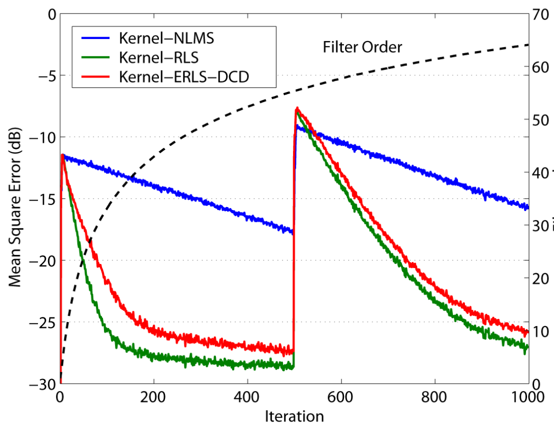

Figure 4: MSE performances of tracking of a Rayleigh fading

5. Conclusion

In this paper, we proposed a kernel adaptive algorithm that has fine convergence property and processing time compare to the conventional kernel-RLS adaptive filters.

References

[1] Ali H. Sayed “Fundamentals of Adaptive Filtering” WILEY, 2003.

[2] Weifeng Liu, José C. Príncipe, Simon Haykin “Kernel Adaptive Filtering” WILEY, 2010.

[3] Yuriy V. Zakharov, George P. White, and Jie Liu “Low-Complexity RLS Algorithms Using Dichotomous Coordinate Descent Iterations” IEEE Trans. Signal Processing, Vol.56, No.7, Jul 2008.

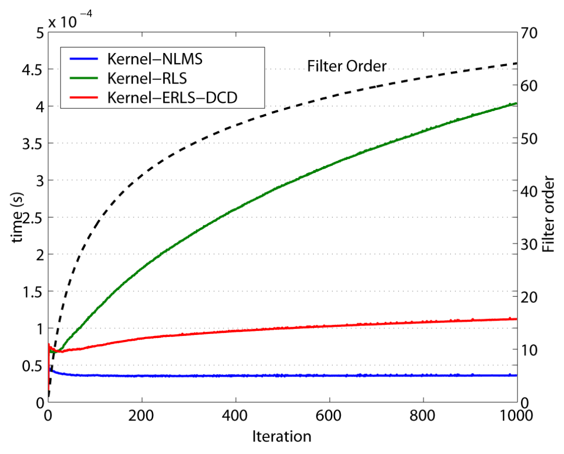

Figure 5: Processing times of tracking of a Rayleigh fading

[4] Weifeng Liu, Il Park, Yiwen Wang, and José C. Príncipe “Extended Kernel Recursive Least Squares Algorithm” IEEE Trans. Signal Processing, Vol.57, No.10, Oct 2009.

[5] Weifeng Liu, Puskal P. Pokharel, and José C. Príncipe “The Kernel Least-Mean-Square Algorithm” IEEE Trans. Signal Processing, Vol.56, No.2, Feb 2008.

[6] Cédric Richard, José Carlos M. Bermudez, and Paul Honeine “Online Prediction of Time Series Data With Kernels” IEEE Trans. Signal Processing, Vol.57, No.3, Mar 2009.