GAUSS RBF EQUALIZER WITH LOW COMPUTATIONAL COMPLEXITY BY KERNEL METHOD

PHƯƠNG PHÁP KERNEL CHO BỘ CÂN BẰNG THÍCH ỨNG PHI TUYẾN ĐA THỨC VỚI ĐỘ PHỨC TẠP TÍNH TOÁN THẤP

Nguyễn Viết Minh

ABSTRACT

Volterra filters, neural networks and Kernel adaptive filters are usually used in nonlinear radio channel equalizations. The greatest advantage of these approaches is their good performance in equalizing but their high computational complexity also limits the convergence rate. Some methods have been proposed recently to solve the problem. In this paper, I extend the computational complexity reducing solution for RBF equalizer, thereby improve the convergence rate.

Keywords*: Adaptive equalization, kernel method, RBF filter.*

TÓM TẮT

Cân bằng kênh vô tuyến phi tuyến thường sử dụng bộ lọc Volterrra, mạng nơron hoặc bộ lọc thích nghi Kernel. Ưu điểm lớn nhất của chúng là thực hiện cân bằng tốt cho các kênh phi tuyến cao, nhưng thường gắn với nhược điểm là độ phức tạp tính toán khá cao dẫn tới hạn chế tốc độ hội tụ. Gần đây một số tác giả đã đề xuất giải pháp giảm độ phức tạp tính toán. Trong bài báo này, tác giả mở rộng giải pháp giảm độ phức tạp tính toán cho bộ cân bằng neuron RBF, qua đó cải thiện tốc độ hội tụ của bộ cân bằng.

Từ khóa*: Cân bằng thích nghi, lọc RBF , phương pháp kernel.*

Nguyễn Viết Minh
Học viện Công nghệ Bưu chính Viễn thông
Email: minhnv@ptit.edu.vn
Ngày nhận bài: 20/08/2017
Ngày nhận bài sửa sau phản biện: 30/09/2017
Ngày chấp nhận đăng: 16/10/2017

1. INTRODUCTION

In recent years, the generation of new transmission technologies such as MIMO, OFDM makes the nonlinear characteristic of radio channels evident, especially in satellite transmission. Nonlinear distortion therefore becomes a imperative problem.

One of the solutions is to use equalizers, which well solve the problem of the nonlinear channels. At first, the equalizers use the Volterra filters with their algorithm described by Volterra polynomial [1]. The greatest advantage of this method is its ability to equal channels with arbitrary nonlinear order, however, its computation is extremely complex due to the high-order polynomials. Recently, equalizers using RBF neural network has been taken into account [2,3]. Their structures are simple but their computational complexity are high and they can only obtain local optimization. In this paper, we propose a solution to overcome the problem of both methods by using Gauss Kernel methods. Our research method based on theoretical computation and simulation verification.

The following content is organized as follow: Section 2: Kernel Hilbert space regeneration, Section 3: Computational complexity reduction for the Gauss RBF equalizer, Section 4: Simulation results and Section 5: Conclusion.

2. KERNEL HILBERT SPACE REGENERATION

Hilbert space is a scalar product space with orthogonal normalized basic $\{x_k\}_{k=1}^{\infty}$, therefore the vectors do not need to be in the initial scalar product space but still can be performed in this form:

$$x = \sum_{k=1}^{\infty} a_k x_k \quad (1)$$

We can say x is developed on the basis of $\{x_k\}_{k=1}^{\infty}$; a_k is the coefficient of the expression.

We define a new vector:

$$y_n = \sum_{k=1}^{n} a_k x_k \quad (2)$$

When $n > m$, we have vector y_m is similar to (2); the square of the Euclidean distance between y_n and y_m is:

$$\|y_n - y_m\|^2 = \left\|\sum_{k=1}^{n} a_k x_k - \sum_{k=1}^{m} a_k x_k\right\|^2 = \left\|\sum_{k=m+1}^{n} a_k x_k\right\|^2$$
$$= \left\|\sum_{k=m+1}^{n} a_k^2 x_k^2\right\| = \sum_{k=m+1}^{n} a_k^2 \left\|x_k^2\right\|$$

Since $\{x_k\}_{k=1}^{\infty}$ is orthonormal, we have:

$$\|y_n - y_m\|^2 = \sum_{k=m+1}^{n} a_k^2 \quad (3)$$

The series in (3) will converge if $\{a_k\}$meet the conditions as below:

1. $\sum_{k=m+1}^{n} a_k^2 \to \infty \qquad n,m \to \infty$
2. $\sum_{k=1}^{m} a_k^2 < \infty$

In other words, $\{y_k\}_{k=1}^{\infty}$ is a Cauchy series. Therefrom we have an important conclusion: "*The scalar product space H is adequate if every Cauchy vector series taken from H converge to a limitation in H, the adequate scalar product space is so called the Hilbert space*".

In 1950, Aronszajn [3] gave a conception which defined Mercer Kernel is a consecutive symmetric positively specific function $k : UxU \to \mathbb{R}$, U is the output region and also the subset of $\mathbb{R}^L$. There are many Kernel types, in this paper we take into account two common Kernel functions as below:

Gaussian Kernel:

$$k(u,u') = \exp\left(-a\|u-u'\|^2\right) \quad (4)$$

This exponential function is often used in RBF neural equalizers.

Kernel polynomial:

$$k(u,u') = \left(u^T u'+1\right)^p \quad (5)$$

This function is often used in Volterra equalization.

The Kernel function has some properties:

Assume that there are two functions *h(.)* and *g(.)* in space H which are defined as below:

$$h = \sum_{i=1}^{I} a_i k(c_i,.) \quad (6)$$

$$g = \sum_{j=1}^{m} b_j k(\tilde{c}_j,.) \quad (7)$$

Here a_i, b_j is the expansion coefficients, $c_i, \tilde{c}_j \in U$. We have:

$$\langle h,g\rangle = \sum_{i=1}^{I}\sum_{j=1}^{m} a_i k(c_i,\tilde{c}_j) b_j \quad (8)$$

satisfies the conditions:

$\langle h,g\rangle = \langle g,h\rangle; \langle (cf+dg),h\rangle = c\langle f,h\rangle + d\langle g,h\rangle;$

$\|f\|^2 = \langle f,f\rangle \geq 0$

From (6), (7), (8) and set $g(.) = k(u,.)$ we have:

$$\langle h,k(u,.)\rangle = \sum_{i=1}^{I} a_i k(c_i,u) = h(u) \quad (9)$$

The expression in (9) is so called the regenerative attribute and the Kernel $k(u,u')$ expresses a function of two vectors: $u,u' \in U$ is called the regenerative Kernel of the vector sapce H.

On the basis of the above propeties, we express the Mercer theorem [Aronszajn, 1950]: The Kernel regenerative is expanded as below:

$$k(u,u') = \sum_{i=1}^{\infty} \varsigma_i \phi_i(u)\phi_i(u') \quad (10)$$

Here ς_i, ϕ_i are proper values and proper functions respectively. The proper values are not negative. Therefore we have structure of the mapping φ as below:

$$\varphi : U \to F$$
$$\varphi(u) = \left[\sqrt{\varsigma_1}\varphi_1(u), \sqrt{\varsigma_2}\varphi_2(u), \ldots\right] \quad (11)$$

$$\varphi(u)^T \varphi(u') = k(u,u') \quad (12)$$

3. COMPUTATIONAL COMPLEXITY REDUCTION FOR THE GAUSS RBF EQUALIZER

Assume that input signal of the RBF euqalizer u(n) is transformed into a characteristic space with depth number F equals to function φ(u). The weight of the signal traversing the equalizer is c. The ouput vector [4]:

$$f(u) = \varphi^T(u)c \quad (13)$$

Assume that weight vector of the equalizer is expressed as a linear combination of training vectors $\phi(y_j)$

$$c = \sum_{j=1}^{m} \alpha_j \phi(y_j) \quad (14)$$

y_j is a subset of x_l, α_j is the weight of y_j respectively. Therefore we can write the output of the equalizer from (13) as:

$$f(u) = \sum_{j=1}^{m} \left(\varphi^T(u)\varphi(y_j)\right)\alpha_j \quad (15)$$

In the RBF equalizer, we consider the Gaussian RBF:

$$k(a,b) = \exp\left(-\|a-b\|^2\right) \quad (16)$$

Assume that vectors u, c are expressed as below:

$$u_n = \left[u(n),\ldots,u(n-M+1)\right] \quad (17)$$

$$c = \left[c_0,\ldots,c_{M-1}\right] \quad (18)$$

From (13), (14), (15) and from instance time *n* we have:

$$f(u_n) = \begin{bmatrix} \varphi(u_n)^T & \varphi(y_1) \\ \varphi(u_n)^T & \varphi(y_2) \\ \vdots & \vdots \\ \varphi(u_n)^T & \varphi(y_m) \end{bmatrix}^T \alpha \quad (19)$$

Combine with the Kernel fuction:

$$k(u,c) = \varphi(u)^T.\varphi(c)$$

Then we achieve:

$$f(u_n)=\begin{bmatrix} k(u_n,y_1) \\ k(u_n,y_2) \\ \vdots \\ k(u_n,y_m) \end{bmatrix}^T \alpha = h_n\alpha \quad (20)$$

Here

$$h_n=\left[k(u_n,y_1),\ldots,k(u_n,y_m)\right]^T \quad (21)$$

From (16) we have:

$$k(a,b)=\exp\left(-\|a-b\|^2\right)=\exp\left(2a^Tb-a^Ta-b^Tb\right) \quad (22)$$

Use h_n from (21) and the relationship between (16) and (22) we have:

$$h_n=\exp\left(\begin{bmatrix} 2u_n^Ty_1-u_n^Tu_n-y_1^Ty_1 \\ 2u_n^Ty_2-u_n^Tu_n-y_2^Ty_2 \\ \vdots \\ 2u_n^Ty_m-u_n^Tu_n-y_m^Ty_m \end{bmatrix}\right) \quad (23)$$

From (23) we have some remarks:

+ In the expression (23) there are 3 comlumns with the components as below:

$$2u_n^Ty_j,\ u_n^Tu_n,\ y_j^Ty_j \quad j=1,2,\ldots,m$$

+ The components of the first column and the second column contains u_n and they are updated over instant time *n*.

+ To compute the first column $u_n^Ty_j$ we have to compute *m* scalar products at each time step *n*.

+ In the second column we only have to define one scalar product $u_n^Tu_n$

+ The third column 3 only contains the scalar product of history output vectors in the dictionary: $y_j^Ty_j \quad (j=1,2,\ldots,m)$ therefore we can save the memory space and the number of computations.

The computation algorithm is shown as below:

Begin

$$U_1=u_1^Tu_1$$

$$D_1=y_1=\begin{bmatrix} u_1 \\ U_1 \end{bmatrix}$$

$$h_1=1;\quad \alpha_1=0;\quad m=1$$

Here m = 2,3,...

$$U_n=u_n^Tu_n$$

$$h_n=\exp\left(D_{n-1}^T\begin{bmatrix} 2u \\ -1 \end{bmatrix}-U_n1\right)$$

If maximum value $|h_n(j)|>\mu_0$

$$D_n=D_{n-1}$$

Else

$$m=m+1$$

$$h_n=\begin{bmatrix} h_n \\ 1 \end{bmatrix}$$

$$D_n=D_{n-1}\cup\begin{Bmatrix} x_n \\ X_n \end{Bmatrix}$$

End

End

Here "1" is a vector with length *m*, its components are 1.

4. SIMULATION RESULTS

In this section we use simulation method to evaluate the computational complexity reducing solution by using Kernel method for the Gauss RBF equalizer. The input signal is described as below:

$$u_n=\left(0,8-0,5\exp\left(-u_{n-1}^2\right)\right)x_{n-1} -\left(0,3+0,9\exp\left(-u_{n-1}^2\right)\right)x_{n-2}+0,1\sin\left(x_{n-1}\pi\right) \quad (24)$$

The initial value of x_{-1},x_{-2} is within $(0,1)$. The input signal order *M* is 4, AWGN noise with 40dB SNR. The threshold value μ_0 = 0.8, the signal length is 10000. The coefficient value in expression (24) is changed at n = 4000 to express the tracing ability.

Our evaluation is based on mean squared error (MSE). The used traning algorithm is KNLMS as proposed and we make a comparision to linear NLMS and conventionnal KNLSM. The simulation is done with m_{max} = 32.

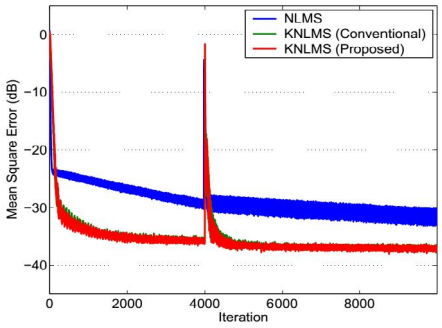

Figure 1. The comparison between the convergence characteristics of linear NLMS, conventional KNLMS and the proposed KNLMS method

The simulation result in Figure 1 shows that the convergence characteristic of the proposed method is similar to conventional KNLMS. Table 1 statistics the required processing time to update the filter weights. Note that with conventional KNLMS, the update time varies from filter order therefore the average time values are showed in the table.

Table 1. The computation time comparison

Algoritms	**Required time for updating weights once** (μs)
NLMS	26.0
Conventional KNLMS	103.6
Proposed KNLMS	40.0

We can see that the computation time of proposed method is less than half of the traditional method. Note that this is a huge decrease since the traditional method requires a higher filter order. It shows that by choosing the optimal orders, the proposed method can be executed with lower computational complexity.

5. CONCLUSION

In this paper, we proposed a complexity reduction for nonlinear Gaussian RBF equalizer. Our method are based on the symmetry characteristic of the Kernel function and the scalar product characteristic in the Hilbert space to reduce the number of computations. By theoretical method combining with simulation method, this paper has proved the ability to reduce the computational complexity, thereby making the convergence rate of the equalizers increased compares to the previous RBF equalizers.

REFERENCES

[1]. H. Zhao and J. Zhang, "A Novel Adaptive Equalizer with DCT Domain Second-Order Volterra Series for Nonlinear channel," Innovative Computing, Information and Control, 2007. ICICIC '07. Second International Conference on, Kumamoto, 2007, pp. 568-568.

[2]. W. Liu, P. P. Pokharel, and J. C. Pr´ıncipe, "The kernel leastmean-square algorithm," IEEE Transactions on Signal Processing, vol. 56, no. 2, pp. 543–554, 2008.

[3]. W. Liu, J. C. Pr´ıncipe, and S. Haykin, Kernel Adaptive Filtering: A Comprehensive Introduction, Jonh Wiley & Sons, New-York, 2010.

[4]. Kiyoshi Nishikawa and koji Makizaki, "Fix order implementation method of kernel adaptive filters with lower computational complexity," APSIPA ASC, 2011.

[5]. 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 Transactions on Signal Processing, vol. 60, no. 5, pp. 2208–2222, 2012.

[6]. C. Richard and J.-C. M. Bermudez, "Closed-form conditions for convergence of the gaussian kernel-least-mean-square algorithm," in Proc. Asilomar, Pacific Grove, CA, USA, Nov. 2012.

[7]. W. Gao, J. Chen, C. Richard, J. Huang, and R. Flamary, "Kernel LMS algorithm with forward-backward splitting for dictionary learning," in Proc. IEEE ICASSP, Vancouver, Canada, 2013, pp. 5735–5739.

[8]. W. Gao, J. Chen, C. Richard, and J. Huang, "Online dictionary learning for kernel LMS," IEEE Transactions on Signal Processing, vol. 62, no. 11, pp. 2765–2777, 2014.

[9]. J. Chen, W. Gao, C. Richard, and J.-C. M. Bermudez, "Convergence analysis of kernel LMS algorithm with pre-tuned dictionary," in Proc. IEEE ICASSP, Florence, Italia, May 2014.

[10]. J. Chen, C. Richard, J.-C. M. Bermudez, and P. Honeine, "Variants of non-negative least-mean-square algorithm and convergence analysis," Tech. Rep., University of Nice SophiaAntipolis, France, 2014. Available at http://www.cedricrichard.fr/Articles/chen2013variants.pdf.

[11]. J. Chen, J.-C. M. Bermudez, and C. Richard, "Steady-state performance of non-negative least-mean-square algorithm and its variants," IEEE Signal Processing Letters, 2014.