A NEURAL-NETWORK-BASED BLIND BEAMFORMING ALGORITHM

Dan Tian, Jin-Kuan Wang, Yan-bo Xue, and Zhi-Gang Liu
School of Information Science and Engineering
Northeastern University
Shenyang，110004，China
E-mail: dtian@mail.neuq.edu.cn

1. Introduction

Beamforming, as a key technology in array signal processing, is widely used in the field of radar, sonar and communication. Conventional beamforming algorithms, such as the minimum-variance distortionless response (MVDR) and the recursive least square (RLS), are based on linear algebra. While these algorithms require time-consuming matrix inversion computation, are not practical for real-time implementation, and are sensitive to the accuracy of the knowledge of the steering vector, which restrain their applications in practical conditions. Neural network possesses strong fault-tolerant capability, generalization capability, and massive parallelism. It can be readily implemented in analog VLSI or optical hardware, or be implemented on special purpose massively parallel hardware. Many kinds of neural network models have been successfully used in beamforming [1], such as multilayer perceptron network [2], Hopfield network [3], radial-basis function neural network (RBFNN) [4], principal component analysis network [5] and fuzzy neural network [6]. In this paper, a fast beamforming algorithm is presented based on the strong numerical approximation, optimization and regularization capabilities of the RBFNN. Because this algorithm needn't know the quality of signal and channel, and the knowledge of the signal steering vector, it is a blind beamforming algorithm.

2. Signal Model

Consider a uniform linear array (ULA) with M omnidirectional sensors spaced by the distance d, and K narrow-band incoherent plane waves impinging from directions $\theta_0, \theta_1, \theta_2, ...\theta_{K-1}$. The observation vector is given by

$$\mathbf{X}(k) = s_0(k)\mathbf{a} + \mathbf{i}(k) + \mathbf{n}(k) = \mathbf{s}(k) + \mathbf{i}(k) + \mathbf{n}(k) \tag{1}$$

where $\mathbf{X}(k) = [x_1(k), x_2(k), ..., x_M(k)]^T$ is the complex vector of array observations, $s_0(k)$ is the signal waveform, $\mathbf{a}$ is the signal steering vector, $\mathbf{i}(k)$ and $\mathbf{n}(k)$ are the interference and noise components, respectively. The output of a narrowband beamforming is given by

$$y(k) = \mathbf{W}^H \mathbf{X}(k) \tag{2}$$

where $\mathbf{W} = [w_1, w_2, ..., w_M]^T$ is the complex vector of beamformer weights, and $(\cdot)^T$ and $(\cdot)^H$ stand for the transpose and Hermitian transpose, respectively.

The weight vector can be found from the maximum of the signal-to-interference-plus-noise ratio (SINR)

$$SINR = \frac{\mathbf{W}^H \mathbf{R}_s \mathbf{W}}{\mathbf{W}^H \mathbf{R}_{\mathbf{i+n}} \mathbf{W}} \tag{3}$$

where $\mathbf{R_s} = E\{\mathbf{s}(k)\mathbf{s}(k)^H\}$, $\mathbf{R_{i+n}} = E\{(\mathbf{i}(k)+\mathbf{n}(k))(\mathbf{i}(k)+\mathbf{n}(k))^H\}$.

3. Weight Vector Estimation

Taking MVDR beamformer as an example, we introduce a classical weight vector estimation algorithm. In this algorithm, to derive the optimal weight vector, the array output is minimized so that the desired signal is received with specific gain, $w^H a(\theta_0) = 1$, while the contributions due to noise and interference are minimized. Then the optimum weight vector is given by the following equation

$$w_{opt} = \frac{R_{xx}{}^{-1} a(\theta_0)}{a^H(\theta_0) R_{xx}{}^{-1} a(\theta_0)} \tag{4}$$

where $R_{xx} = E[x(t)x(t)^H]$ is the correlation matrix of the received signals. Since the above equation is not practical for real-time implementation, an adaptive algorithm must be used to adapt the weights of the array in order to track the desired signal and to place nulls in the directions of the interfering signals.

4. Neural Blind Beamforming

In (4), to solve the optimum weight vector, time-consuming matrix inversion computation is required. Consider the neural network has parallel architecture, and its training phase and performance phase can be separated. In this paper, the neural network is used to solve this problem. The optimum weight vector is a nonlinear function of the correlation matrix [see Eq. (4)]. Note that a radial basis function neural network can approximate an arbitrary function from an input space of arbitrary dimensionality to an output space of arbitrary dimensionality. This network model is used to implement the mapping from correlation matrix to the weight vector.

A. The RBFNN Model

The radial-basis function network is a special three-layered feedforward network, which consists of the input layer, the output layer, and the hidden layer, and is shown in Figure1.

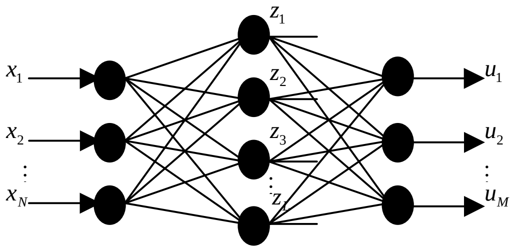

Fig. 1. Architecture of a three-layer RBFNN

Assume that the input layer, the hidden layer, and the output layer have N, L, M nodes respectively. Gaussian functions are selected as radial-basis function, the network output vector is given by

$$u_m = \sum_{l=1}^{L} \omega_{l,m} e^{\frac{\|x-c(l)\|^2}{\sigma_l^2}} \quad (m = 1, \cdots, M) \tag{5}$$

where $x = [x_1, x_2, \cdots, x_N]$ is the input vector to the network, $c(l)$ and σ_l^2 are the mean and standard deviation of the l th Gaussian function, $\omega_{l,m} (l = 1, \cdots, L; m = 1, \cdots, M)$ is the weight value from the m th node in the output layer to the l th node in the hidden layer. Training samples are divided into L classes by K -mean learning algorithm, $c(l)$ is the l th clustering center vector, and σ_l^2 is the average distance to the first few nearest neighbors of the means of the other Gaussian functions.

B. Generation of Training Data

To reduce the learning space, the redundant information in the input vector of the network should be eliminated, and the information related to the DOA of the sources should be extracted. The elements in covariance matrix are given by

$$R_{ml} = E[x_m x_l^*] = \sum\nolimits_{i=1}^{M} \sum\nolimits_{k=1}^{M} P_{ik} \exp\{ j2\pi fd[(l-1)\sin\theta_k - (m-1)\sin\theta_i]\} + \delta_{ml}\sigma_n^2 \tag{6}$$

where $(\cdot)^*$ denotes the complex conjugate , δ_{ml} is the Kronecker delta and $P_{ik} = E[(s_i s_k{}^*)]$ denotes the source correlation matrix. By exploiting the symmetry in correlation matrix, one need only consider either the upper or lower triangular part of the matrix [7]. In this paper, the upper triangular half of R is used. From (6), $R_{mm} = \sum_{i=1}^{M} P_i + \sigma_n^2$ does not carry any information on the position of the signal sources. So an L(L-1)/2 component vector can be constructed as network input

$$z = [R_{12}, R_{13}, \ldots, R_{1L}; R_{23}, \ldots, R_{2L}; \ldots; R_{(L-1),L}]$$

Since in practical application the power levels received at both mobile stations and basestations are kept at the same level by full power control, we normalize the input to unify the parameter space $\check{z} = z / \|z\|$, where $\|\cdot\|$ is the Euclidean norm.

To simplify the generation of the training data, in this paper, a new network mapping relationship is defined

$$w_0 = R_{xx}^{-1} a(\theta_0) \tag{7}$$

Then the network can be trained with examples of input-output pairs $\{(\overset{\vee}{z}{}^l, w_0^l); l = 1, 2, \cdots, N_T\}$, where N_T is the number of the training pairs. In the one-dimensional array, sources are located at elevation angels θ ranging from -90° to $+90^\circ$ to span the field of view of the antenna. Consider the condition of a single signal, and the learning resolution of θ as 1°, then 181 input-output pairs can be obtained. Compared to (4), the new mapping defined in this paper saves massive computation given by the following equation

$$w_{opt} = \alpha w_0 \tag{8}$$

where $\alpha = 1 / a^H(\theta_0) R_{xx}^{-1} a(\theta_0)$.

C. Network Output Post-Processing and Network Testing

In testing phase, input pre-processing method is the same as that of in training phase, i.e. both of them generate the network input vector $\overset{\vee}{z}$. Present $\overset{\vee}{z}$ at the input layer of the trained RBFNN, the output layer of the trained RBFNN will produce, as an output, the estimation of w_0. To obtain the estimation of the optimum weight for the array output, the following post-processing is required

$$w_{opt} = \left(\frac{1}{w_0^H R_{xx} w_0}\right)^* w_0 \tag{9}$$

Unlike the least mean-square, recursive least squares, or the sample matrix inversion algorithms, where the optimization is carried out whenever the directions of the desired or interfering signals change, in our approach, the weights of the trained network can be used to produce the optimum weights needed to steer the narrow beams of the adaptive array to the directions of desired users.

5. Simulation Results

Some simulations are conducted in this section to verify the proposed method. The sensor displacement is taken to be half the wavelength of the signal waves. The pattern of an array of 8 elements receiving one desired signal that arrives from 20° is shown in Figure 2. The SNR of the source is 10 dB with respect to the noise. Figure 3 illustrates the pattern of an array of 10 elements receiving a desired signal that arrives from 10°, and two interference that arrive from 0° and 20°. The SNR of the sources is 10 dB with respect to the noise. To evaluate the accuracy of the weight vector estimation, the network recalling error relative to desired value is given as $\varepsilon = (\|W_{MVDR}\| - \|W_{NNBF}\|) / \|W_{MVDR}\|$ [8], where W_{NNBF} is the post-processed weight vector estimated by neural beamformer. An array of 8 elements receives a single signal sampled from -90° to $+90^\circ$, with $\Delta\theta = 5^\circ$. The SNR of the source is 10 dB with respect to the noise. The appropriate degree of the weight vectors estimation is shown in Figure 4.

6. Conclusion

In this paper, a new beamforming algorithm is presented based on RBFNN. To simplify the generation of training data pairs, a new network mapping relationship is defined, which is post-processed to approximate Wiener solution. Computer simulations show the high degree of accuracy of our approach. Conventional beamforming algorithms require the knowledge of DOA estimation, while imprecise information can lead to the degradation of the performance. The algorithm presented in this paper is a blind algorithm, which solves the problem.

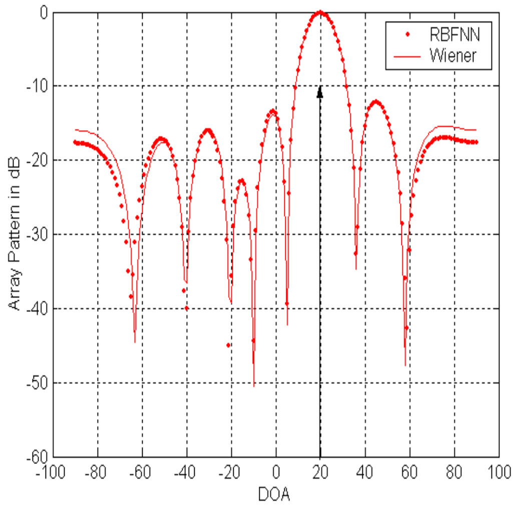

Fig. 2. Adapted pattern of an eight-element array receiving a single signal

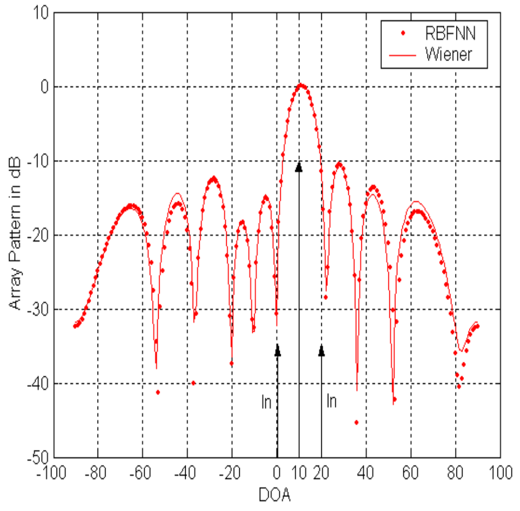

Fig. 3. Adapted pattern of a ten-element array receiving one desired signal and two interfering

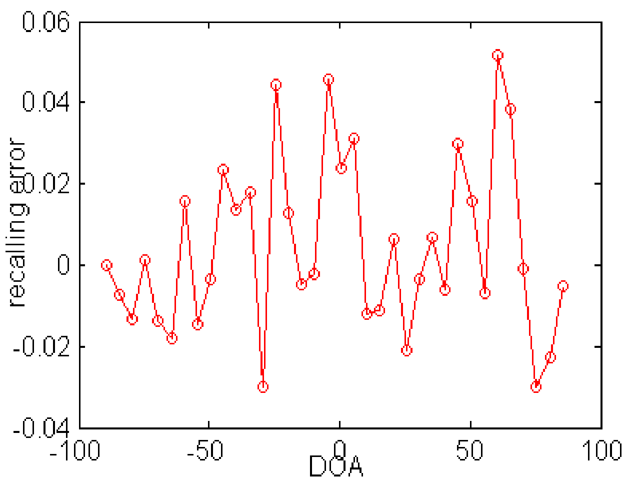

Fig. 4. Network recalling error

References

[1] K.-L. Du, A. K. Y. Lai, K. K. M. Cheng, and M. N. S. Swamy, “Neural methods for antenna array signal processing:a review,” Signal Processing, vol.82, pp. 547-561, Apr.2002.

[2] A. B. Suksmono and A. Hirose, “Adaptive beamforming by using complex-valued multiplayer perceptron,” ICANN/ICONIP, pp. 959-966, 2003.

[3] M. Hirari and M. Hayakawa, “Direction of arrival estimation using blind separation of sources,” Radio Sci, vol.34, pp. 693-701, Mar.1999.

[4] Seigiy A. Vorobyov and Andrzej Cichocki, “Hyper radial basis function neural networks for interference cancellation with nonlinear processing of reference signal,” Digital Signal Processing, vol.11, pp. 205-221, Mar.2001.

[5] S. Fiori, “A neural minor component analysis approach to robust constrained beamforming,” IEE Proceedings - Vision, Image and Signal Processing, vol.150, pp. 205 – 218, Apr.2003.

[6] F. J. Lin and R. J. Wai, “Hybrid control using recurrent fuzzy neural network for linear-induction motor servo drive,” IEEE Trans.Fuzzy Systems, vol.9, pp.102-115, Jan.2001.

[7] A. H. EI Zooghby, C. G. Christodoulou, and M. Georgiopoulos, “A neural-network-based linearly constrained minimum variance beamformer,” Microwave and Optical Technology Letters, vol.21, pp. 451-455, Jun.1999.

[8] K.-L. Du, K. K. M. Cheng, and M. N. S. Swamy, “A fast neural beamformer for antenna arrays,” IEEE International Conference on Communications, pp. 139-144, April-May 2002.