A SPLIT QUATERNION NONLINEAR ADAPTIVE FILTER

Che Ahmad Bukhari Che Ujang[†]*, Clive Cheong Took*[†]*, Alek Kavcic*[*] *and Danilo P. Mandic*[†]

[†] Electrical and Electronic Engineering Department, Imperial College London, SW7 2BT, UK
[*] Department of Electrical Engineering, University of Hawaii, HI 96822, USA
E-mails: {che.che-ujang07, c.cheong-took, d.mandic}@ic.ac.uk, alek@spectra.eng.hawaii.edu

ABSTRACT

A split quaternion learning algorithm for the training of nonlinear finite impulse response filters for the modelling of hypercomplex signals is proposed. A rigorous derivation takes into account the non-commutativity of the quaternion product, an aspect not taken into account in the existing nonlinear architectures, such as the Quaternion Multilayer Perceptron (QMLP). It is shown that the additional information present within the proposed algorithm provides an improved performance over QMLP. Simulation on both benchmark and real-world signals support the approach.

Index Terms— Machine Learning, Quaternion Multilayer Perceptron, Cauchy-Riemann-Fueter equation.

1. INTRODUCTION

Recent advances in sensor technology, and their applications (sensor networks, data fusions) have highlighted the need for multidimensional learning algorithm to process such multiple channels efficiently. Instead of treating each dimension separately, the signal needs to be processed as a whole, thus fully exploiting the multidimensional statistics. A number of machine learning algorithms have been developed in order to fulfill this objective [1, 2, 3].

The early approaches were based on the representation of multidimensional signals and the network parameters as vectors in $\mathbb{R}^n$. One of the first such learning algorithms is the Three-Dimensional Vectors Back-Propagation (3DV-BP) for neural networks [1]. However, its matrix operation does not exploit the coupling between the three dimensions. One improvement was the Vector Product Back Propagation (VP-BP), which addressed this issue through the use of vector products [2]. However, this algorithm encountered a problem of a zero weight adaptation term in the presence of non-zero error.

In order to overcome these issues for four dimensional signals, it is natural to consider the processing in the quaternion domain, $\mathbb{H}$. Quaternions were first conceived by W. Hamilton in 1843, and have found applications in robotics [4], molecular modeling [5], and computer graphics [6]. The dilemma of quaternion modelling versus the modelling in $\mathbb{R}^4$ has been long present [7], and quaternion based analysis in machine learning has not been as prominent as vector analysis.

In the area of neural networks, the Quaternion Multilayer Perceptron (QMLP) has been introduced and benefits from the algebraic properties of the quaternion domain resulting in an enhanced performance over previous algorithms of this kind [3]. One of the applications of QMLP algorithm is in polarized signal classification [8]. Despite the satisfactory results obtained, the performance of QMLP could be improved as it ignores the non-commutativity aspect of the quaternion product.

The aim of this paper is to introduce a quaternion valued nonlinear finite impulse response (FIR) adaptive filter. Due to a number of open issues in nonlinear quaternion functions, it is difficult to find an equivalent to a "fully" complex approach in $\mathbb{C}$. Hence, we resort to "split" quaternion nonlinear functions. A new learning algorithm, the Split Quaternion Nonlinear Adaptive Filter (SQAF), is derived in order to explicitly address the non-commutativity of the quaternion product. Simulation studies support the analysis on both benchmark and real-world multidimensional data.

2. QUATERNION ALGEBRA

A basic quaternion variable $q \in \mathbb{H}$ has a real part and three imaginary parts, which can be represented as

$$\begin{aligned} q &= [q_a, \mathbf{q}] \\ &= q_a + q_b\imath + q_c\jmath + q_d\kappa \end{aligned} \tag{1}$$

where $q_a, q_b, q_c, q_d \in \mathbb{R}$ and $\imath$, $\jmath$, κ, are orthogonal unit vectors.

The relationship between the orthogonal unit vectors $\imath$, $\jmath$, κ are

$$\begin{aligned} \imath\jmath &= \kappa; \quad \jmath\kappa = \imath; \quad \kappa\imath = \jmath; \\ \imath\jmath\kappa &= \imath^2 = \jmath^2 = \kappa^2 = -1 \end{aligned} \tag{2}$$

Another quaternion operation crucial to this study is the multiplication, given by

$$\begin{aligned} wx &= [w_a, \mathbf{w}][x_a, \mathbf{x}] \\ &= [w_a x_a - \mathbf{w}\cdot\mathbf{x}, w_a\mathbf{x} + x_a\mathbf{w} + \mathbf{w}\times\mathbf{x}] \end{aligned} \tag{3}$$

where the symbols "$\cdot$" and "$\times$" denote respectively to the dot-product and cross-product. It is clear that $wx \neq xw$, due to the presence of the outer product.

Similar to the complex case, the conjugate of a quaternion q is $q^* = [q_a, \mathbf{q}]^* = [q_a, -\mathbf{q}]$, and the norm $\| q \|_2^2 = qq^*$. In this paper, all quantities are quaternion valued, unless stated otherwise.

3. NONLINEAR FUNCTIONS IN $\mathbb{H}$

The choice of a quaternion valued nonlinearity is still an open issue. In order for a nonlinear function to be "fully" quaternion, it must satisfy the Cauchy-Riemann-Fueter equation, which is an extension of the Cauchy-Riemann equation from $\mathbb{C}$ to $\mathbb{H}$ [9].

Authorized licensed use limited to: Imperial College London. Downloaded on June 9, 2009 at 05:20 from IEEE Xplore. Restrictions apply.

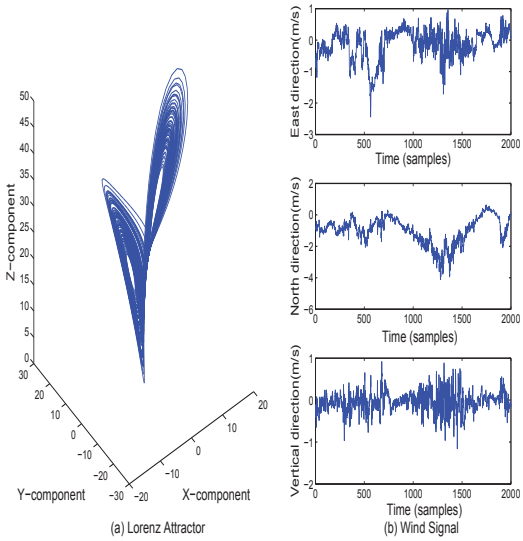

Fig. 1. Left: The 3D Lorenz attractor with parameter $\alpha = 10$, $\rho = 28$ and $\beta = 8/3$ in (22) . Right: The 3D wind signal.

Cauchy-Riemann equations state that for a function $f(z)$

$$\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\imath = 0 \tag{4}$$

where $z = x + y\imath$. Given that $f(z) = u(x,y) + v(x,y)\imath$, the Cauchy-Riemann conditions can be shown to be

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}; \quad \frac{\partial v}{\partial x} = -\frac{\partial u}{\partial y} \tag{5}$$

In practice, conditions (5) are very convenient and simplify the derivation of learning algorithms for nonlinear adaptive filtering in $\mathbb{C}$. The nonlinear functions which satisfy these conditions (5) are known as fully complex functions; Kim and Adali have proposed three classes of fully complex nonlinear functions to be used as approximations in Complex Multilayer Perceptron [10].

On the other hand, the Cauchy-Riemann-Fueter conditions in $\mathbb{H}$ state that for a function $f(q)$ [9]

$$\frac{\partial f}{\partial t} + \frac{\partial f}{\partial x}\imath + \frac{\partial f}{\partial y}\jmath + \frac{\partial f}{\partial z}\kappa = 0 \tag{6}$$

where $q = t + x\imath + y\jmath + z\kappa$.

At present, no results are available addressing "elementary transcendental" functions that satisfy the Cauchy-Riemann-Fueter conditions, making the fully quaternion nonlinear adaptive filtering a discipline in its infancy. For example, applying Cauchy-Riemann-Fueter equation (6) to the elementary transcendental tanh function yields

$$\begin{aligned}\frac{\partial tanh(q)}{\partial q} &= sech^2(q) + (sech^2(q)\imath)\imath + (sech^2(q)\jmath)\jmath + (sech^2(q)\kappa)\kappa \\ &= -2sech^2(q) \neq 0\end{aligned} \tag{7}$$

Hence, analogously to the split complex filtering in $\mathbb{C}$, we propose to use split quaternion nonlinear function, where each component of the quaternion valued signal is processed independently.

4. QMLP AND SQAF

The standard cost function to be minimized is given by

$$\begin{aligned}E(n) &= e_a^2(n) + e_b^2(n) + e_c^2(n) + e_d^2(n) &(8)\\ &= e(n)e^*(n) &(9)\end{aligned}$$

where *e(n)* = *d(n)* − *y(n)* and *y(n)* = $\overline{\sigma}$(*s(n)*), with *d(n)*, *y(n)*, and $\overline{\sigma}(\cdot)$ denoting respectively the desired signal, output signal and split quaternion nonlinear function. The "net input" *s(n)* is defined as *s(n)* = $\mathbf{w}^T(n)\mathbf{x}(n)$ where $\mathbf{w}(n)$ and $\mathbf{x}(n)$ correspond to the adaptive weight vector and the filter input, and symbols $(\cdot)^T$ and $(\cdot)^*$ denote the transpose and quaternion conjugate operator. The split quaternion nonlinear function is given by

$$\overline{\sigma}(q) = \overline{\sigma}(q_a) + \overline{\sigma}(q_b)\imath + \overline{\sigma}(q_c)\jmath + \overline{\sigma}(q_d)\kappa \tag{10}$$

4.1. Derivation of the learning algorithm for QMLP

The QMLP algorithm minimizes the cost function (8), whereby a gradient descent update of the coefficients, is given by $\mathbf{w}(n+1) = \mathbf{w}(n) - \mu\nabla_{\mathbf{w}}E(n)$. The error gradient for the outer layer of a neural network is given by [3]

$$\begin{aligned}\nabla_{\mathbf{w}}E(n) &= \frac{\partial E}{\partial \mathbf{w}_a} + \frac{\partial E}{\partial \mathbf{w}_b}\imath + \frac{\partial E}{\partial \mathbf{w}_c}\jmath + \frac{\partial E}{\partial \mathbf{w}_d}\kappa &(11)\\ &= e(n)\underbrace{\overline{\sigma}'\left(\mathbf{w}^T(n)\mathbf{x}(n)\right)\Big(-2\mathbf{x}^*(n)\Big)}_{\nabla_{\mathbf{w}}y(n)} &(12)\end{aligned}$$

where $\overline{\sigma}'$ denotes the derivative with respect to *s(n)*. From (12), it is clear that $\nabla_{\mathbf{w}}E(n)$ is a function of $\nabla_{\mathbf{w}}y(n)$ [the derivation is given in the Appendix], that is

$$\nabla_{\mathbf{w}}y(n) = \overline{\sigma}'\left(\mathbf{w}^T(n)\mathbf{x}(n)\right)\Big(-2\mathbf{x}^*(n)\Big) \tag{13}$$

However, if the gradient of (8) is derived based on *e(n)e*(n)*, it is evident that

$$\begin{aligned}\nabla_{\mathbf{w}}E(n) &= e(n)\frac{de^*(n)}{dw(n)} + \frac{de(n)}{dw(n)}e^*(n)\\ &= -\Big(e(n)\nabla_{\mathbf{w}}y^*(n) + \nabla_{\mathbf{w}}y(n)e^*(n)\Big) &(14)\end{aligned}$$

which clearly demonstrates that $\nabla_{\mathbf{w}}E(n)$ is also a function of $\nabla_{\mathbf{w}}y^*(n)$. Similarly, it can be shown that

$$\nabla_{\mathbf{w}}y^*(n) = \overline{\sigma}'\left(\mathbf{x}^H(n)\mathbf{w}^*(n)\right)\Big(4\mathbf{x}^*(n)\Big) \tag{15}$$

This is why it is crucial to consider (9) instead of (8), such that the non-commutativity quaternion product is highlighted.

4.2. Derivation of SQAF

Based on the properties of quaternion algebra and nonlinear gradient descent, the SQAF for FIR adaptive filters is now derived. Since the nonlinear tanh function is employed as the nonlinear function, the properties of odd-symmetric functions apply so that $\overline{\sigma}\big(-\mathbf{w}^T(n)\mathbf{x}(n)\big) = -\overline{\sigma}\big(\mathbf{w}^T(n)\mathbf{x}(n)\big)$. This odd-symmetric property is exploited explicitly as follows

$$\overline{\sigma}'^*\big(\mathbf{w}^T(n)\mathbf{x}(n)\big) = \overline{\sigma}'\left(\mathbf{x}^H(n)\mathbf{w}^*(n)\right) \tag{16}$$

Authorized licensed use limited to: Imperial College London. Downloaded on June 9, 2009 at 05:20 from IEEE Xplore. Restrictions apply.

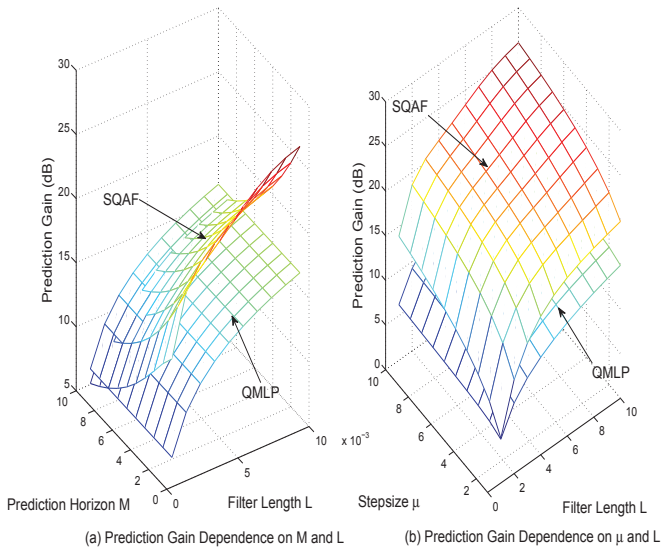

Fig. 2. The performance of SQAF and QMLP on the prediction of 3D Lorenz signal.

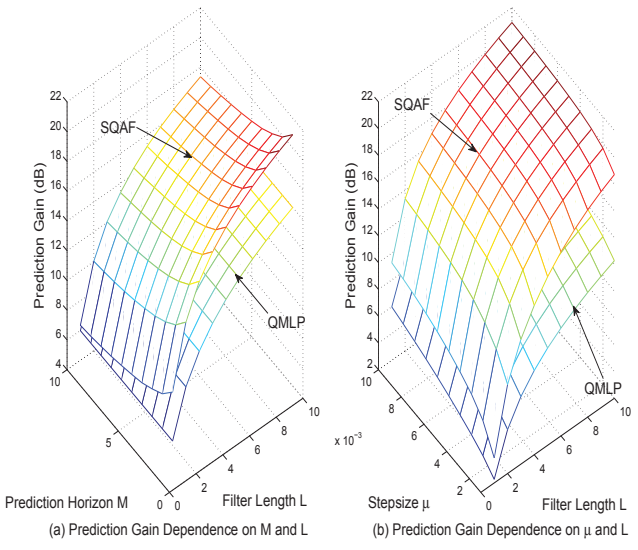

Fig. 3. The performance of SQAF and QMLP on the prediction of 3D wind signal.

This is possible due to the split quaternion nature of the filter where the nonlinearity is applied elementwise.

The cost function (9) can be further expressed as

$$\begin{aligned} E(n) &= \Big(d(n)-y(n)\Big)\Big(d^*(n)-y^*(n)\Big) \\ &= d(n)d^*(n)-d(n)y^*(n) \\ &\quad -y(n)d^*(n)+y(n)y^*(n) \end{aligned} \tag{17}$$

The error gradient of (17) can be calculated as

$$\begin{aligned} \nabla_{\mathbf{w}}E(n) &= -d(n)\nabla_{\mathbf{w}}y^*(n)-\nabla_{\mathbf{w}}y(n)d^*(n) \\ &\quad +y(n)\nabla_{\mathbf{w}}y^*(n)+\nabla_{\mathbf{w}}y(n)y^*(n) \end{aligned} \tag{18}$$

Replacing (13) and (15) into (18) yields

$$\begin{aligned} \nabla_{\mathbf{w}}E(n) &= -4e(n)\overline{\sigma}'\left(\mathbf{x}^H(n)\mathbf{w}^*(n)\right)\mathbf{x}^*(n) \\ &\quad +2\overline{\sigma}'\left(\mathbf{w}^T(n)\mathbf{x}(n)\right)\mathbf{x}^*(n)e^*(n) \end{aligned} \tag{19}$$

Finally, the weight update for the SQAF for the training of quaternion valued nonlinear adaptive filters can be expressed as

$$\begin{aligned} \mathbf{w}(n+1) &= \mathbf{w}(n)+\mu\Big(2e(n)\overline{\sigma}'\left(\mathbf{x}^H(n)\mathbf{w}^*(n)\right)\mathbf{x}^*(n) \\ &\quad -\overline{\sigma}'\left(\mathbf{w}^T(n)\mathbf{x}(n)\right)\mathbf{x}^*(n)e^*(n)\Big) \end{aligned} \tag{20}$$

where μ is the real-valued learning rate.

5. SIMULATIONS

Simulations were performed in an M-step prediction setting as to compare SQAF with QMLP. The SQAF was implemented with a filter length L whereas the QMLP had one hidden layer comprising L inputs, three hidden neurons and one output neuron. Both algorithms were fed with an input $\mathbf{x}(n)=[x(n),x(n-1),\cdots,x(n-L)]^T$. The nonlinear function used was the tanh function.

The prediction gain R_p was used as a quantitative measure of performance, and is defined as

$$R_p = 10\log_{10}\frac{\sigma_x^2}{\sigma_e^2} \tag{21}$$

where σ_x^2 and σ_e^2 denote respectively the estimated variance of the input and error. Two three-dimensional (3D) processes were considered (pure quaternion): the Lorenz attractor and a 3D wind field.

5.1. Lorenz Attractor

The Lorenz attractor is governed by coupled partial differential equations

$$\begin{aligned} \frac{\partial x}{\partial t} &= \alpha(y-x); \quad \frac{\partial y}{\partial t} = x(\rho-z)-y; \\ \frac{\partial z}{\partial t} &= xy-\beta z \end{aligned} \tag{22}$$

where α, ρ and β >0. The Lorenz system was initialized with the following parameters: $\alpha=10$, $\rho=28$ and $\beta=8/3$. The three-dimensional plot of the Lorenz attractor is shown in Figure 1(a). Figure 2 shows the performance of both algorithms as a function of the prediction horizon M, with $\mu=10^{-2}$ and as a function of the stepsize μ, with M=1.

It can be seen that the prediction gain for SQAF was generally higher than that of QMLP.

5.2. Wind Forecasting

In the next simulation, a 3D wind field was used as an input. Figure 1(b) shows the 3D wind data dimension-wise. Figure 3 depicts

Authorized licensed use limited to: Imperial College London. Downloaded on June 9, 2009 at 05:20 from IEEE Xplore. Restrictions apply.

the performance of SQAF and QMLP as a function of the prediction horizon M and stepsize μ.

The prediction gain for the SQAF was always better than that of QMLP in both case studies (varying learning rate and prediction horizon), thus indicating the benefits of fully exploiting the quaternion algebra .

6. DISCUSSIONS

In both the Lorenz attractor and wind forecasting simulations, as M increased, the prediction gain deteriorated as the algorithm needed to predict more steps into the future. On the other hand, the increment of learning rate from 10^{-3} to the optimum value of 10^{-2} increased the prediction gain for both algorithms. The performance of the SQAF was generally better than that for QMLP, as it takes into account a more complete information about the multidimensional signal. It was noted that the QMLP was less affected by the size of prediction horizon as compared to the SQAF. The deterioration of the QMLP prediction gain (Figure 2 and Figure 3) with the increase of prediction horizon is almost negligible due to the structural richness of the neural network compared to the single layer FIR architecture of SQAF. Another aspect that needs to be addressed is the computational complexity of the algorithms. The SQAF requires 68L+24 multiplications and 60L+18 additions, whereas 108L+216 multiplications and 96L+168 additions are needed for QMLP. For instance, the computational complexity of QMLP is more than double that of SQAF even when L=1.

In summary, the advantages SQAF has over QMLP are

- Taking into account the non-commutativity of the quaternion product leads to an improved performance;
- The simple architecture of SQAF allows for easier implementation;
- Computational complexity is much lower, making SQAF more suitable for real-time processing.

7. CONCLUSIONS

A split quaternion stochastic gradient algorithm for the training of quaternion valued nonlinear adaptive filters has been proposed. This has been achieved by employing the odd-symmetry of some elementary transcendental functions based on the non-commutativity of the quaternion product. It has also been shown that there are no "fully-complex" extensions of elementary transcendental functions in $\mathbb{C}$, as these do not satisfy the Cauchy-Riemann-Fueter conditions in $\mathbb{H}$. The proposed algorithm has been shown to exhibit excellent performance for the prediction of real-world vector fields.

8. REFERENCES

[1] T. Nitta and T. Furuya, "A 3D vector version of the back-propagation algorithm," *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, vol. 2, pp. 511–516, Nov. 1992.

[2] T. Nitta, "A back-propagation algorithm for neural networks based on 3D vector product," *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, vol. 1, pp. 589–592, Oct. 1993.

[3] P. Arena, L. Fortuna, G. Muscato, and M. G. Xibilia, "Neural networks in multidimensional domains: fundamentals and new trends in modelling and control," *Lecture Notes in Control and Information Sciences Springer-Verlag London*, vol. 234.

[4] D. Biamino, G. Cannata, M. Maggiali, and A. Piazza, "MAC-EYE: a tendon driven fully embedded robot eye," *In Proceedings IEEE-RAS International Conference on Humanoid Robots*, pp. 62–67, 2005.

[5] C. F. F. Karney, "Quaternions in molecular modelling," *Journal of Molecular Graphics and Modelling*, vol. 25, pp. 595–604, 2007.

[6] S. B. Choe and J. J. Faraway, "Modeling head and hand orientation during motion using quaternions," *Journal of Aerospace*, vol. 113, pp. 186–192, 2004.

[7] O. Heaviside, "Vectors versus quaternions," *Nature*, vol. 47, no. 1223, 1893.

[8] S. Buchholz and N. L. Bihan, "Polarized signal classification by complex and quaternionic multi-layer perceptrons," *International Journal of Neural Systems*, vol. 18, no. 2, 2008.

[9] A. Sudbery, "Quaternionic analysis," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 85, pp. 199–225, 1979.

[10] T. Kim and T. Adali, "Approximation by fully complex multilayer perceptrons," *Neural Computation*, vol. 15, pp. 1641–1666, 2003.

A. APPENDIX

To calculate $\nabla_{\mathbf{w}} y(n)$, $\mathbf{w}^T(n)\mathbf{x}(n)$ is first expanded into

$$\mathbf{w}^T(n)\mathbf{x}(n) = \begin{bmatrix} \mathbf{w}_a^T\mathbf{x}_a - \mathbf{w}_b^T\mathbf{x}_b - \mathbf{w}_c^T\mathbf{x}_c - \mathbf{w}_d^T\mathbf{x}_d \\ \mathbf{w}_a^T\mathbf{x}_b + \mathbf{w}_b^T\mathbf{x}_a + \mathbf{w}_c^T\mathbf{x}_d - \mathbf{w}_d^T\mathbf{x}_c \\ \mathbf{w}_a^T\mathbf{x}_c + \mathbf{w}_c^T\mathbf{x}_a + \mathbf{w}_d^T\mathbf{x}_b - \mathbf{w}_b^T\mathbf{x}_d \\ \mathbf{w}_a^T\mathbf{x}_d + \mathbf{w}_d^T\mathbf{x}_a + \mathbf{w}_b^T\mathbf{x}_c - \mathbf{w}_c^T\mathbf{x}_b \end{bmatrix} \quad (23)$$

The gradient $\nabla_{\mathbf{w}} y(n)$ can be further defined as

$$\nabla_{\mathbf{w}} y(n) = \nabla_{\mathbf{w}_a} y(n) + \nabla_{\mathbf{w}_b} y(n)\imath + \nabla_{\mathbf{w}_c} y(n)\jmath + \nabla_{\mathbf{w}_d} y(n)\kappa \quad (24)$$

Using (23), the derivatives on the righthand side of (24) can be computed as

$$\nabla_{\mathbf{w}_a} y(n) = \overline{\sigma}'\left(\mathbf{w}^T(n)\mathbf{x}(n)\right)(\mathbf{x}_a + \mathbf{x}_b\imath + \mathbf{x}_c\jmath + \mathbf{x}_d\kappa) \quad (25)$$

$$\begin{aligned} \nabla_{\mathbf{w}_b} y(n)\imath &= \overline{\sigma}'\left(\mathbf{w}^T(n)\mathbf{x}(n)\right)(-\mathbf{x}_b + \mathbf{x}_a\imath - \mathbf{x}_d\jmath + \mathbf{x}_c\kappa)\imath \\ &= \overline{\sigma}'\left(\mathbf{w}^T(n)\mathbf{x}(n)\right)(-\mathbf{x}_a - \mathbf{x}_b\imath + \mathbf{x}_c\jmath + \mathbf{x}_d\kappa) \end{aligned} \quad (26)$$

$$\begin{aligned} \nabla_{\mathbf{w}_c} y(n)\jmath &= \overline{\sigma}'\left(\mathbf{w}^T(n)\mathbf{x}(n)\right)(-\mathbf{x}_c + \mathbf{x}_d\imath + \mathbf{x}_a\jmath - \mathbf{x}_b\kappa)\jmath \\ &= \overline{\sigma}'\left(\mathbf{w}^T(n)\mathbf{x}(n)\right)(-\mathbf{x}_a + \mathbf{x}_b\imath - \mathbf{x}_c\jmath + \mathbf{x}_d\kappa) \end{aligned} \quad (27)$$

$$\begin{aligned} \nabla_{\mathbf{w}_d} y(n)\kappa &= \overline{\sigma}'\left(\mathbf{w}^T(n)\mathbf{x}(n)\right)(-\mathbf{x}_d - \mathbf{x}_c\imath + \mathbf{x}_b\jmath + \mathbf{x}_a\kappa)\kappa \\ &= \overline{\sigma}'\left(\mathbf{w}^T(n)\mathbf{x}(n)\right)(-\mathbf{x}_a + \mathbf{x}_b\imath + \mathbf{x}_c\jmath - \mathbf{x}_d\kappa) \end{aligned} \quad (28)$$

Finally, substituting (25) -(28) into (24) yields

$$\nabla_{\mathbf{w}} y(n) = \overline{\sigma}'\left(\mathbf{w}^T(n)\mathbf{x}(n)\right)\Big(-2\mathbf{x}^*(n)\Big) \quad (29)$$

Authorized licensed use limited to: Imperial College London. Downloaded on June 9, 2009 at 05:20 from IEEE Xplore. Restrictions apply.