Historical Consistent Complex Valued Recurrent Neural Network

Hans-Georg Zimmermann[1], Alexey Minin[2,3], and Victoria Kusherbaeva[3]

[1] Siemens AG, Corporate Technology, 81730 Muenchen, Mch-P, R. 53.220, Germany
[2] Technische Universitat Munchen, Institut fur Informatik VI, Boltzmannstrase 3, 85748 Muenchen, Germany
[3] Siemens LLC, Corporate Technology, 191186, St. Petersburg, Volynskiy lane 3 of. 905, Russia
{Hans_Georg.Zimmermann,Alexey.Minin,Victoria.Kusherbaeva.ext}@Siemens.com

Abstract. Recurrent Neural Networks are in the scope of the machine learning community for many years. In the current paper we discuss the Historical Consistent Recurrent Neural Network and its extension to the complex valued case. We give some insights into complex valued back propagation and its application to the complex valued recurrent neural network training. Finally we present the results for the the Lorenz system modeling. In the end we discuss the advantages of the proposed algorithm and give the outlook.

Keywords: complex valued neural networks, recurrent neural networks, complex valued recurrent neural networks, complex dynamics analysis.

1 Introduction

Historical Consistent Neural Network (further HCNN) was first described in the work of Zimmermann [1]. This architecture is very interesting due to the stability of training and simplicity of the construction. It allows a unique correspondence between the dynamical equations, neural network architecture and the locality of the learning algorithms. This architecture models the behavior of the dynamical system which can be described by the eq.1 below:

$$\begin{cases} s_t = f\left(s_{t-1}, u_t\right) \\ y_t = g\left(s_t\right) \end{cases} \tag{1}$$

where u is a model input, s is an internal state of the model and y is the model output, f and g are some transition functions (recurrent connection is done through the states). Modeling of such systems is of big interest in many application areas. But what happens if there is a need in complex valued neural network inputs? Even with real valued dynamics modeling there are a lot of unsolved problems with the things like stable training, learning and stopping criteria. In the following paper we will try to show the transition from the Real Valued HCNN (further RVHCNN) to the Complex Valued HCNN (further

T. Honkela et al. (Eds.): ICANN 2011, Part I, LNCS 6791, pp. 185–192, 2011.

CVHCNN). We will present our insights in the complex valued back-propagation and its application to the CVHCNN. At the end of the paper we present the results obtained modeling of Lorenz system dynamics.

2 Historical Consistent Complex Valued RNN

2.1 Complex Valued Back-Propagation

Complex Valued Back Propagation (further CVBP) was already discussed in many papers, here we can refer to the papers of Haykin [5], Hirose [6], Nitta [7], Kim [8] etc. One can see that the interest in this topic remains. In this subsection we will rearrange the knowledge in this area and show some insights on complex valued back propagation.

Complex Valued Error Function. The main thing in neural networks approximation problems is to minimize an error function. This function typically selected as a mean squared error MSE, root mean squared error $RMSE$ etc. First let us start with the definition of the error function which we are going to minimize in the complex valued case. Now, all neuron inputs (further *netin*), network outputs (further *netout*), targets (the desired output values) and network weights are complex numbers (consist of real and imaginary parts). The minimization of the complex valued error would be a complicated task due to many reasons discussed below. Therefore, in the majority of the papers mentioned above one can see the following error function (see eq.2):

$$E = \sum_{t=1}^{T} \left(\text{netout}_\tau - \text{target}_\tau\right) \overline{\left(\text{netout}_\tau - \text{target}_\tau\right)} \rightarrow \min_W \tag{2}$$

where τ is a pattern number, T is the number of patterns and the bar above the brackets means complex value conjunction ($z = x + iy, z \in \mathbb{C}, \bar{z} = x - iy$). For more types of error function authors refer to the paper of Gangal [9] (in this paper authors describe a lot of different error functions). The property of the presented error function is that it is mapping from $\mathbb{C} \rightarrow \mathbb{R}$. This automatically means that error function is not analytical (see Zimmermann [3]), which means that it does not have neither analytical derivative nor Taylor expansion. Abscence of the Taylor expansion means impossibility to apply Steepest Descent algorithm for the Neural Network training. However, we can treat this error function with the so called Wirtinger calculus; here we refer to Brandwood [4]. Let $f(z) = u(z_r, z_{im}) + iv(z_r, z_{im})$ (here $z \in \mathbb{C}, z_r$ is the real part of z and z_{im} is the imaginary part of z, u and v are some real valued functions), then one can write two real valued variables as $z_r = (z + \bar{z})/2$, $z_{im} = (z - \bar{z})/2i$. One should consider z and $\bar{z}$ as independent from each other. Then function $f : \mathbb{C} \rightarrow \mathbb{C}$ can be expressed as $f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{C}$ by rewriting it as $f(z) = f(z_r, z_{im})$. Using the theorems below when evaluating the gradient, we can directly compute the derivatives with respect to the complex argument, rather than calculating

individual real-valued gradients. Here f is our error function written in the eq.2. Let $f : \mathbb{R} \times \mathbb{R} \to \mathbb{C}$ be a function of real variables x and y such that $g(z, \bar{z}) = f(z_r, z_{im})$, where $z = z_r + iz_{im}$ and that g is analytic with respect to z and $\bar{z}$ independently. The requirement for the analyticity of $g(z, \bar{z})$ with respect to z and $\bar{z}$ is equivalent to the condition on real differentiability of $f(z_r, r_{im})$ since we can move from one form of the function to the other using the simple linear transformation given above.

Theorem 1. *Let $f(z, \bar{z})$ be a real-valued function of the vector-valued complex variable z where the dependence on the variable and its conjugate is explicit. By treating z and $\bar{z}$ as independent variables, the quantity pointing in the direction of the maximum rate of change of $f(z, \bar{z})$ is $\nabla_{\bar{z}}(f(z))$.*

This theory has been studied extensively in Brandwood [4]. For us it is important that using Wirtinger calculus we can calculate the gradient for the non analytical error functions, we can calculate the Taylor expansion for the error function $(E(w + \Delta w) = E(w) - \eta g^T \bar{g} + \frac{\eta^2}{2} \bar{g}^T G \bar{g})$ and apply the Steepest Descent learning $\Delta w = -\eta \cdot \bar{g}$ for a small learning rate η. The figure below (see fig.1 shows the overall scheme for the CVBP. Using the presented CVBP one can train complex valued neural networks. Unfortunately, there is another issue which arises during the training of the complex valued neural networks, which is complex valued transition function.

Complex Valued Transition Function. As one can find from the mentioned above papers – analytical transition functions are unbounded due to the Liouville theorem (for example see Haykin [5]). This unboundness of functions can destroy

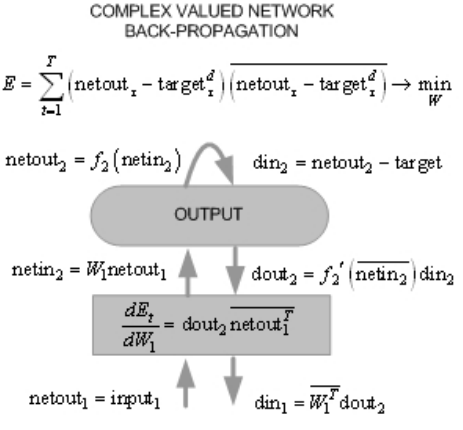

Fig. 1. Complex Valued Back-Propagation. Notations: *netin* - layer inputs, *netout* - layer outputs, *din* -layer derivative input, *dout* -layer derivative output, W_i are network weights, arrows show the information flow, $(\bar{\cdot})$ - means complex conjunction. The figure depicts the locality of the CVBP algorithm and independence of the BP from the network architecture.

any computations in few iterations. To solve the problem one can bound the weights, in case the "good" region for the function arguments is known. If not, it is possible to use the so-called "engineered" functions, which are artificial but do not allow unlimited growth of the output amplitude. To calculate the derivative of such functions one should use Wirtinger calculus discussed above. An example of such functions is given further: $f(z) = \tanh(z_r) + i \cdot \tanh(z_{im})$ or $f(z) = \tanh(r)e^{i \cdot \phi}$ (here r is absolute value of z and ϕ is angle of z). One can use the *sin* or *tanh* functions but he or she should take into account the position of singularities or the speed of ascending (descending) of these functions and do not allow the network arguments (weights) to go to infinite regions of these functions.

Now let us describe the architecture of the Complex Valued Historical Consistent Neural Network (further CVHCNN).

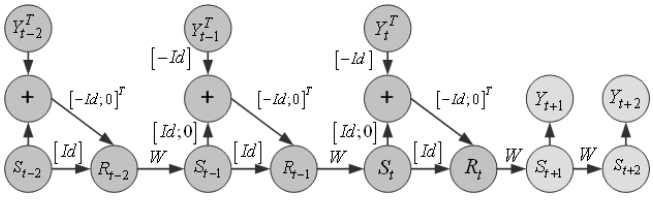

Fig. 2. HCNN architecture. Notations: Y^T is target, R_t is teacher forcing block, S_i is state vector, W is shared weights matrix,$[Id]$ is identity matrix, $(\cdot)^T$ means matrix transpose

2.2 Architecture Description and Insights on Training

The Historical Consistent Neural Network (further HCNN, for more information authors refer to Zimmermann [1]) can do the modeling of the systems which correspond to the eq.1. This architecture is working in the following way (see fig. 2): some bias signal is coming to the most left state of the network (often it is a random noise distributed normally). This network does not have inputs (in the usual sense of this word, states transfer the information from one state to the next one, doing unfolding in time), which means it is autonomous. It tries to model the given outputs with an internal dynamics. To train it one should use the so called Teacher Forcing (Zimmermann [1]) for the time moments from $t-n$ to t (see fig.2), where n is the number of network layers (recurrence layers). One can use the following system of equations to describe the teacher forcing training:

$$\begin{cases} \begin{bmatrix} \tau \leq t : s_{\tau+1} = f(W(s_\tau - [Id;0]\,(y_\tau - y^d_\tau))) \\ \tau > t : s_{\tau+1} = f(W s_\tau) \end{bmatrix} \\ \forall t : y_\tau = [Id, 0] s_\tau \end{cases} \tag{3}$$

where $[Id, 0] = \left[\underbrace{\overbrace{1, 1, ..., 1}^{N_o}, 0, 0, ..., 0}_{N_s}\right]$, N_o is number of network outputs, N_s is number of network states (number of hidden neurons), f is non linear transition function, W is the weights matrix. Note, that matrix W and function f are every time (at each state of the model) the same matrix and function y_τ is the network output, y_τ^d is the network desired output. To have the forecast one should apply matrix W and then transition function f to the state vector, to obtain the network outputs one has to apply the matrix $[Id; 0]$ to the obtained state vector. Iteratively applying a matrix W and a function f one can obtain the forecast for the needed horizon.

Using this teacher forcing training and the CVBP algorithm we can train the CVHCNN. One should admit the stability of CVHCNN training. The answer to this fact is that due to the teacher forcing we avoid uncontrolled behavior of the information flow inside the CVHCNN which can be rather dangerous for the stability of computations caused by the unlimited functions or function singularities.

2.3 Problem Description and Modeling Results

Problem Description. The Lorenz Problem is an example of a non-linear dynamic system corresponding to the long-term behavior of the Lorenz oscillator. The Lorenz oscillator is a 3-dimensional dynamical system [10]:

$$\begin{cases} \dfrac{dx}{dt} = \sigma(y - x); \dfrac{dy}{dt} = x(\rho - z) - y; \dfrac{dz}{dt} = xy - \beta z \end{cases} \tag{4}$$

here $x, y, z \in \mathbb{C}$, parameters we will take as following: $\beta = 8/3; \quad \rho = 28; \sigma = 10; \quad h = 0.01$ where h – the step in Euler scheme of differential equation system solving. Complex values can be obtained by multiplying x, y, z by $e^{i \cdot sin(t)}$, where t – time.

Modeling Results. The task was to achieve the maximum possible forecast for all 3 coordinates of Lorenz system development (see eq.4). Let us describe the CVHCNN architecture. The number of layers is 15, the number of states is 15, the matrix weights are randomly initialized in the $[-0.2, -0.2i; 0.2, 0.2i]$ rectangle, the number of outputs is 3 (x, y and z coordinates respectively), the activation function is $f(z) = tanh(z)e^{i \cdot \phi}$, learning rate η was equal to 0.005. The data structure is like following. Training set contains 1400 data points, Test set – 583 and Forecast set – 20. Test set contains 1 step predictions for 583 moments in time. The Forecast set contains 20 steps iterated prediction (network uses forecasted values to predict the next one). All sets follow one after another (time series are historical (dynamical) consistent). To estimate the quality of the network we have used the Root Mean Squared Error and R^2 – is the Determination coefficient. The negative R^2 values arise because of the smooth Lorenz system behavior (desired outputs are close to their average values, which

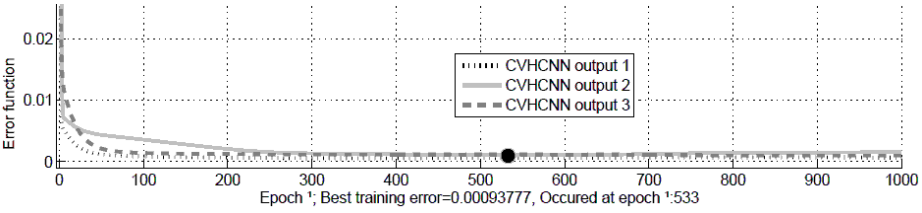

Fig. 3. Training Error Convergence

Table 1. Test set results. Output 1 is x coordinate, Output 2 is y coordinate etc.

RMS for the test set			R^2 for the test set		
output #	absolute	angle	output #	absolute	angle
output 1	0.0005	0.0002	output 1	0.97	0.99
output 2	0.0011	0.0001	output 2	0.90	0.99
output 3	0.0008	0.0001	output 3	0.88	0.99

makes R^2 very negative despite network outputs). Each experiment was repeated 10 times and the results were averaged except the best and the worst.

Training set results are presented at the fig. 3: the error function for all three outputs is exponentially decreasing and is nearly 0. The Test set results are presented for the absolute and angle parts of the CVHCNN output. From the tables below one can see that the Test set results are very promising: all RMS values are close to 0, R^2 values are close to 1. Forecast set results are presented for the absolute part (see fig. 4) and for the angle (phase) part (see fig. 5) of the CVHCNN output. Tables below (see 2) show the statistics for the 20 steps prediction. One should be very careful while treating the $R^2 < 0$. In case desired outputs do not change significantly the denominator of the R^2 is very small, which makes the complete fraction very big. Subtracting this big value from 1 makes the R^2 negative. One should admit that the CVHCNN is giving not only the forecast for the absolute part of the complex output, which contains Lorenz values, but also predicts the sin of time, which stands at the angle (phase) part of the complex output. Therefore by looking at the angle value we can say to which moment in time the prediction is related. In case the prediction for the time (phase of the complex output) starts behaving incorrect (we know this moment

Table 2. Forecast set results. Output 1 is x coordinate, Output 2 is y coordinate etc.

RMS for 20 steps forecast			R^2 for 20 steps forecast		
output #	absolute	angle	output #	absolute	angle
output 1	0.0000	0.0201	output 1	0.86	< 0
output 2	0.0000	0.0059	output 2	0.70	< 0
output 3	0.0007	0.0004	output 3	< 0	< 0

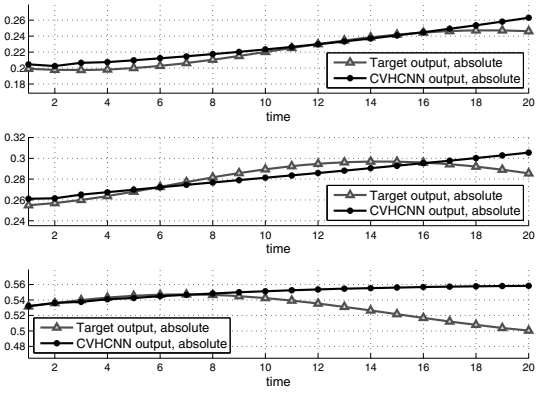

Fig. 4. Forecast for 20 steps, absolute parts of the CVHCNN outputs

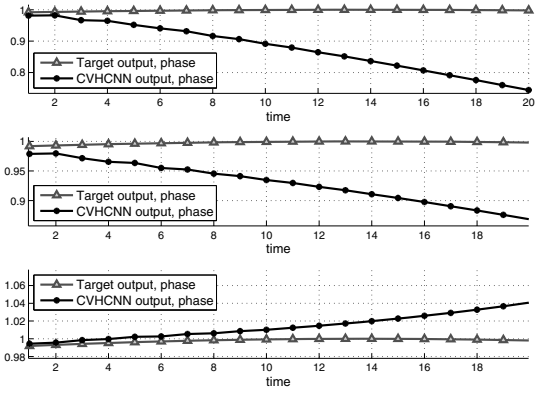

Fig. 5. Forecast for 20 steps, angle parts of the CVHCNN outputs

since time is changing in a linear manner) then we cannot trust our absolute parts predictions any more. The last statement is not proven statistically or theoretically, therefore it should be checked for consistency.

3 Conclusions and Outlook

In the present paper we have shown the back propagation algorithm, presented its locality and extended it for the complex valued case. Then we discussed the CVHCNN which is recurrent. Then we showed that CVHCNN is trainable, moreover the training is robust under certain conditions. We have considered two real world problems and showed the applicability of the CVHCNN for the modeling of such systems (both, real valued and complex valued dynamics).

One of the advantages which arise from the complex representation is the natural representation of time inside the complex valued network which means that

we have the prediction not only for the absolute part of the complex (which is for example Lorenz coordinate) but also the time moment to which this prediction is related. This gives us a possibility to think about the continuous dynamics modeling, which was not possible to do to the moment. In the future we are planning to apply CVHCNN for the continuous dynamics modeling and to show, that trained network can give the prediction which includes the moment in time to which this prediction is related. This option is to be investigated in the future.

References

1. Zimmermann, H.G., Grothmann, R., Schafer, A.M., Tietz, C.: Dynamical Consistent Recurrent Neural Networks. In: Proc. of the Int. Joint Conference on Neural Networks (IJCNN), vol. 3, pp. 1537–1541 Montreal (2005)
2. Schaefer, A.M., Zimmermann, H.G.: Recurrent Neural Networks are Universal Approximators. In: Proc. of International Conference on Artificial Neural Networks (ICANN), Athens. LNCS, vol. 17(4), pp. 253–263. Springer, Heidelberg (2006)
3. Zimmermann, H.G., Minin, A., Kusherbaeva, V.: Comparison of the Complex Valued and Real Valued Neural Networks Trained with Gradient Descent and Random Search Algorithms. In: European Symposium on Artificial Neural Networks, ESANN 2011 (to appear 2011)
4. Brandwood, D.H.: A complex gradient operator and its application in adaptive array theory. IEE Proceedings, F: Communications, Radar and Signal Processing 130(1), 1116 (1983)
5. Leung, H., Haykin, S.: The Complex Back Propagation. IEEE Transactions on Signal Processing 39(9), 2101–2104 (1991)
6. Hirose, A.: Continuous Complex-Valued Back-propagation Learning. Electronics Letters 28(20), 1854–1855 (1992)
7. Nitta, T.: An Extension of the Back-Propagation Algorithm to Complex Numbers. Neural Networks 10(8), 1391–1415 (1997)
8. Kim, T., Adali, T.: Fully Complex Multi-Layered Perceptron Network for Nonlinear Signal Processing. VLSI Signal Processing 32, 29–43 (2002)
9. Gangal, A., Kalra, P., Chauhan, S.: Performance Evaluation of Complex Valued Neural Networks Using Various Error Functions. Enformatika 23, 27–32 (2007)
10. Lorenz, E.N.: Deterministic nonperiodic flow. Lecture Supplement 20, 130–141 (1963)