ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

Comparison of the Complex Valued and Real Valued Neural Networks Trained with Gradient Descent and Random Search Algorithms

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

1- Siemens AG - Corporate Technology
Muenchen, Germany.
2- Technischen Universitat Muenchen - Robotics dept.
Muenchen, Germany.
3- Siemens OOO- Corporate Technology
St. Petersburg, Russia

Abstract. Complex Valued Neural Network is one of the open topics in the machine learning society. In this paper we will try to go through the problems of the complex valued neural networks gradients computations by combining the global and local optimization algorithms. The outcome of the current research is the combined global-local algorithm for training the complex valued feed forward neural network which is appropriate for the considered chaotic problem.

1 The Differences between Feed-Forward Real Valued and Complex Valued Neural Networks

In the following paper we briefly introduce Real Valued Neural Network (further RVNN) structure [1] which consists of the neurons, where the last one can be described with the following equation (see eq.(1)):

$$y_i = \tanh\left(\sum_{j=1}^{T} W_{ij} X_j + Wb_j\right), \left[X_j, W_{ij}, Wb_j\right] \in \mathbb{R} \qquad (1)$$

where y_i is the output of the i^{th} neuron, X_j – is the j^{th} element of the input vector with T elements, W_{ij} is the matrix of weights, Wb_j is the vector of bias parameters and function tanh – is the activation (transition) function.

In the Complex Valued Neural Network (further CVNN) case inputs, weights, bias parameters and outputs are complex numbers $\left[X_j, W_{ij}, Wb_j\right] \in \mathbb{C}$. The first problem in the complex representation of the neural network is the activation function. Following the Liouville theorem one can show that every bounded entire function is constant (for the complete complex plane) [2]. This immediately means that if one wants his/her non linear function to be differentiable, it means the function will be at least unbounded. For example the tanh function will have singularity points which occur periodically. At these points function goes to infinite values which explode any computations. Following Haykin [3], the elegant way to avoid that is to use a sigmoid complex function which has a singularity at infinity and then limiting the search space

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

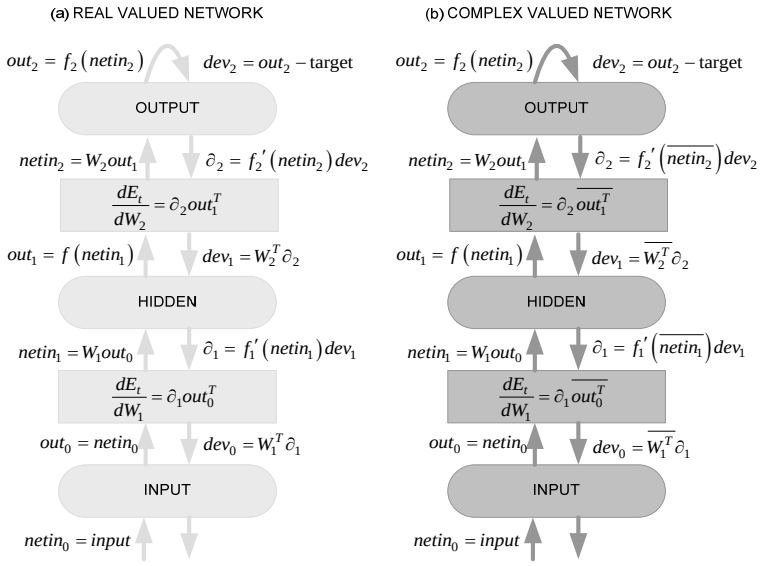
for the weights and the bias parameters they will never go from the "safe" region which in this case should be far from infinity. For instance, one can use the region $[-1..1]$ for the real and imaginary parts of the weights.

2 The Differences between the Complex Valued and Real Valued Back-Propagation Algorithms

The real valued back propagation algorithm is the local algorithm, which can be applied for any architecture. Let us briefly describe the Real Valued Back Propagation (further RVBP) algorithm. The goal of the neural network training is to minimize the approximation error. In order to do that one can use the RVBP and "ladder" algorithm (see scheme 1a) introduced in [4] by Zimmermann. Following the idea one should use Taylor expansion for the error and introduce the weights adaptation procedure based on this expansion.

$$E(W+\Delta W) = E(W) + g^T \Delta W + \tfrac{1}{2}\Delta W^T G \Delta W \qquad (2)$$

Then the rule for weights adaptation can be written as $\Delta w = -\eta \cdot \partial E / \partial W$, where η is a learning rate. Updating the weights using this rule one can find the local minimum for the error. The "ladder algorithm" allows an efficient computation of the partial derivatives of the error locally, between the layers.



Scheme 1. a) Back propagation algorithm for the 3 layered real valued feed forward neural networks. b) Back propagation algorithm for the 3 layered complex valued feed forward neural networks (at this schemes bias is not presented for simplicity). The bar above some values means conjunction of the complex value. The picture follows the notations given by Zimmermann in [1].

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

Now let us discuss the complex valued feed forward neural network [3 - 6]. The complex valued neural network error can be presented in the way explained at the eq.(3) below:

$$E(w)=\sum_{t=1}^{T}\left(y_t-y_t^d\right)\overline{\left(y_t-y_t^d\right)}\rightarrow\min_{w} \tag{3}$$

where y_t is an output, $y_t{}^d$ is a desired output (target output), w are network weights and bias parameters and T is the amount of patterns for NN training. Here $\overline{\left(y_t-y_t^d\right)}$ is conjugated to the $\left(y_t-y_t^d\right)$. After the error is calculated one should expand this error using Taylor expansion in order to obtain the rule for the weights adaptation. $\Delta w=-\eta\cdot\overline{\partial E/\partial w}$. For this purpose define the $\partial E/\partial w$, since now $E\in\mathbb{R}$ and $\left[w_{ij},\triangle w\right]\in\mathbb{C}$. Unfortunately, $\partial E/\partial w$ is not defined since the derivative of the error does not exist, which means the following (see eq.(4) below):

$$\frac{\partial E}{\partial w}=\frac{\partial\left(y_t-y_t^d\right)\overline{\left(y_t-y_t^d\right)}}{\partial w}=z\frac{\partial\bar{z}}{\partial z}+z\frac{\partial\bar{z}}{\partial z},\frac{\partial\bar{z}}{\partial z}=\lim_{h\rightarrow 0}\frac{\overline{(z+h)}-\bar{z}}{h}=\begin{Bmatrix}1 & h\in\mathbb{R}\\ -1 & h\in\mathbb{C}\end{Bmatrix} \tag{4}$$

where h is a small step. The first term of the eq.7 is well defined, while the second term is not defined in mathematical sense. This second term makes the error non analytical.

The solution for this problem is given by Wirtinger calculus. Let $f(z)=u(z_r,z_{im})+iv(z_r,z_{im})$, then one can write the 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 [2].

Theorem 1. If the function $f(z,\bar{z})$ is real-valued and analytic with respect to z and $\bar{z}$, all stationary points can be found by setting the derivative (in the sense just given) with respect to either z or $\bar{z}$ to zero.

Theorem 2. 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 $\nabla_{\bar{z}}(f(z))$.

3 Combination of the Global and Local optimization Algorithms

Due to the problems with the non analytical error function it has been decided to start the training with the global search algorithm, which does not require any gradient information. Then after we have found the region for the local minimum, we can apply the gradient descent algorithm to converge to this minimum.

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

3.1 **Random Search Algorithm and its Complex Valued Case**

Following the works [7, 8] the adaptive random search method is a global stochastic optimization technique which does not require any a priori information about an optimization problem.

Let $I_i \subset X$ be a perspective interval for variable x_i, $i \in 1:n$; a Cartesian product of sets I_i, $i \in 1:n$ is a perspective domain with center point x_i^0, $i \in 1:n$.

The process of random search is divided into steps N_s (in terms of neural networks – epochs). On every step the vector x^j, $j \in 1:N_s$ is randomly selected according to some distribution (in this work step-function distribution) and the value of the objective function $\Phi^j = \Phi(x^j)$ is calculated.

By using $\Phi_{min}^j = \min\{\Phi^j, \Phi_{min}^j\}$ a minimal value of the objective function in the step j, $j \in 1:N_s$ is calculated.

A wide set of experiments have been conducted, in order to show adaptive random search algorithm's effectiveness and to make recommendations for choosing heuristic parameter values [7, 8].

In the case of the CVNN the algorithm works independently with real and imaginary parts of the complex, thus, expanding the dimension of optimization space twice.

3.2 **Combination of the Complex Valued Back Propagation and the Random Search Algorithm**

As it was mentioned above, calculation of gradients is a problem in the complex valued case. In order to simplify the problem it has been decided to use non gradient global optimization method (RSA) [7, 8] and to use it as an initialization method, which should find the local minimum region. In order to converge to the minimum itself we have decided to apply the gradient descent method with very small learning rate in order to reach the minimum. Therefore, first we make several hundred epochs of RSA and then we apply the CVGD. This combination proved to be better than separate usage of both algorithms for the considered problem which will be described in details in the Results chapter).

The typical behavior of the training error can be seen at the fig.1. below (summarized graph for 10 runs of the algorithms).

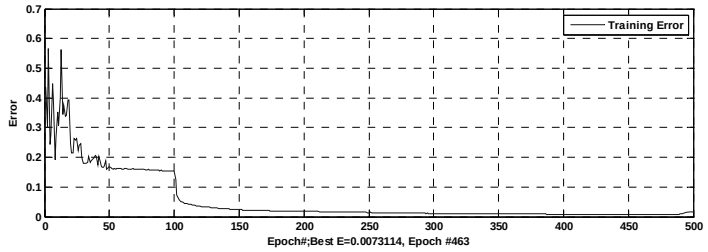


Fig.1. Training Error behaviors. First 100 epochs – complex random search, next 400 epochs is complex gradient descent. Best error is 0.007 at epoch # 463.

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

4 Experimental Results and Conclusions

The problem which we considered as a test case for the mentioned above algorithms is a well known chaotic problem:

$$X(t+1) = \lambda X(t)\left(1 - X(t)\right) \tag{5}$$

where λ is data complexity. This parameter was selected to be 3.9 which is quite high complexity of the data ($X(0) = 0.01$). Training set was chosen to be first 1000 values, test set – next 400 values. As the absolute part the values of the eq. (5) have been used. As a phase of the complex number it has been decided to use the sinus of time since sin function makes the time bounded. This is one of the advantages of the complex representation of data that one can naturally deal with time even in the feed forward networks.

First task is to make one step prediction for the test set. For this purpose 6 lagged values of the $X(t)$ will be used. Target value is the X_{t+1} value. Second task is to make 20 steps iterated forecast for the first 20 points of the test set. Iterated forecast means that we use the forecasted value as an input at the next iteration, therefore after some iteration all inputs are replaced by the forecasted values.

To present the results we have decided to use the following statistical measures: Mean Squared Error and the adjusted R^2 (adjusted coefficient of determination).

The architecture of the feed forward neural network was chosen to have 6 inputs, 40 hidden nodes in two hidden layers (20-20) and 1 output. The structure of the nonlinearities was chosen to have linear activation at the input and output layers and hyperbolic tangent in both hidden layers. Each neural network was used 5 times and then the results were averaged. The learning rate for the neural network training was selected to be relatively small 0.002.

The results for the CVNN are presented at the table 1 below. To calculate the statistics for the complex valued output we had to separate the complex onto the absolute part and the phase part (following the Euler representation). Table 2 shows the results for the RVNN.

Table 1. Results for the complex valued neural network are presented. RSA/CVGD shows how many epoch were given to the random search and how many epoch were given for the gradient descent.

Epochs	Adj. R^2 for 1 step		Error for 1 step		Adj. R^2 for 20 steps		Error for 20 steps	
RSA/CVGD	angle	abs	angle	abs	angle	abs	angle	abs
0/500	0.86	0.51	0.06	0.04	0.67	**-0.11**	0.15	**0.12**
100/400	0.83	0.56	0.08	0.03	-0.50	-0.73	0.72	0.19
200/300	**0.94**	**0.93**	0.00	0.01	0.42	-0.16	0.27	0.13
300/200	0.81	0.58	0.09	0.03	**0.73**	-0.30	**0.12**	0.14
400/100	0.82	0.24	0.08	0.06	0.62	-0.15	0.18	0.13
500/0	0.12	-1.82	0.43	0.25	-1.58	-1.63	1.25	0.30

Table 2. Results for the real valued neural network are presented. RSA/GD shows how many epochs were given to the random search and how many epochs were given for the Gradient Descent.

Epochs	Adj. R^2	Error	Adj.R^2	Error

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

RSA/CVGD	for 1 steps	for 1 steps	for 20 steps	for 20 steps
0/500	**0.97**	0.00	-0.05	0.12
100/400	0.96	0.00	-0.01	0.11
200/300	0.91	0.00	-0.22	0.14
300/200	0.83	0.01	-0.36	0.15
400/100	0.78	0.01	-0.29	0.14
500/0	0.44	0.05	-0.45	0.16

Analyzing the tables 1 and 2 one can say, that for this example both networks are giving the same quality of the forecast. For 20 steps prediction both networks are predicting for 5-6 steps ahead with $R^2>0.9$.

Experiments have shown that RVNN can converge to smaller errors and can produce a bit better 1 step prediction (see table 1 and 2).

Thus, complex valued neural network is of the same quality as real valued neural network (if to compare the absolute part of the CVNN output and the RVNN output). The complex back propagation algorithm is inconsistent due to the error function type but fortunately one can use the Wirtinger calculus to avoid the problems. CVNN is very sensitive to the initialization. In order to avoid bad initialization of weights with random numbers RSA was introduced. After RSA is applied CVGD always converges to local minimum.

Since there is no difference in approximation quality and training between RVNN and CVNN we can think about extension of the application area of the NN in industry. For example in electrical engineering modeling dealing with complex valued inputs is much more convenient.

References

[1] R. Neuneier, H.G. Zimmermann, How to Train Neural Networks, Neural Networks: Tricks of the Trade, Springer 1998, pp. 373-423.

[2] D. H. Brandwood. A complex gradient operator and its application in adaptive array theory. IEE Proceedings, F: Communications, Radar and Signal Processing, 130(1):1116, 1983.

[3] H. Leung, S. Haykin, *The Complex Back Propagation*, IEEE Transactions on Signal Processing., Vol.39, No.9, September 1991., pp. 2101 – 2104.

[4] T. Kim, T. Adali, *Fully Complex Multi-Layered Perceptron Network for Nonlinear Signal Processing*, VLSI Signal Processing 32, pp. 29-43, 2002.

[5] A. Hirose, *Continuous Complex-Valued Back-propagation Learning*, Electronics Letters, Vo1.28, No. 20, September 24, pp. 1854 -1855, 1992.

[6] T. Nitta, *An Extension of the Back-Propagation Algorithm to Complex Numbers*, Neural Networks, Vol.10, No.8, pp.1391-1415, 1997.

[7] Yu. Sushkov and A. Abakarov, The algorithm of adaptive random search for discrete-continuous optimization, *proceedings of the 5th St. Petersburg Workshop on Simulation*, pp. 11–17, June 26-July 2, 2005.

[8] V. Kusherbaeva and Yu. Sushkov, Statistical investigation of Random Search, *Stochastic optimization in information science*, 3:pp. 21-36, S.-Petersburg: SPbSU, 2007.