A Look Inside the Learning Process of Neural Networks

K. BERTELS, L. NEUBERG, S. VASSILIADIS, AND G. PECHANEK

Koen Bertels is a Professor in Computer Information Systems at the Universities of Namur and Antwerp, Belgium. L. Neuberg, originally trained as an engineer, also holds a Ph.D. in Management from the University of Namur. He now works for a French bank. S. Vassiliadis is a Chair Professor in TU Delft (EE dept.) He has worked previously with IBM, SUNY, and Cornell University. Dr. Pechanek, with 67 publications, 32 patents, and 28 years in the electronics industry, is one of the founders of BOPS, Inc. where he is their Chief Scientist and Director of Architecture.

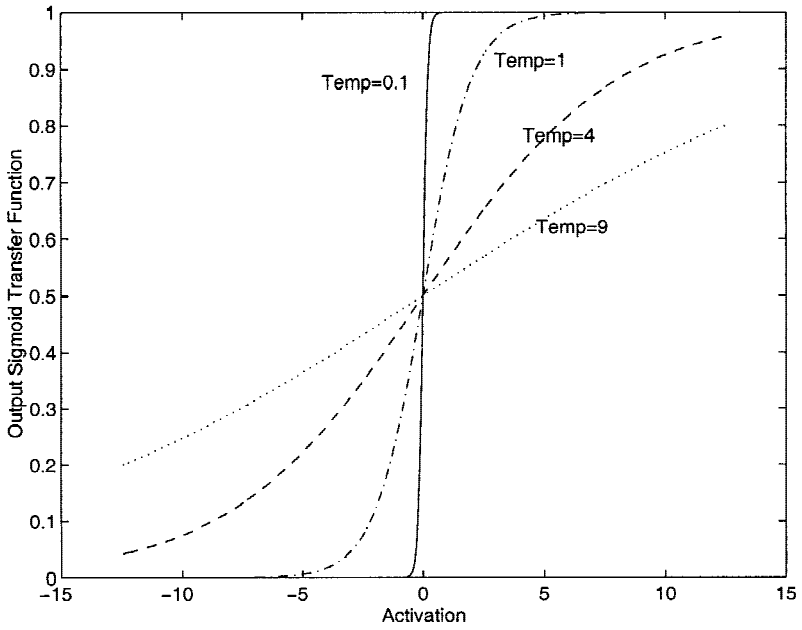
One of the best-known results of the sciences of complexity is that complex systems learn on the edge of chaos, by which is meant that both chaotic and orderly states coexist and that the system remains close to this borderline and may switch from one state to the other. In this article, we take a look inside the learning process of neural networks, and we more specifically focus on the role of chaos for learning a nonlinearly separable function, the XOR Boolean function. It is clear that neural networks are complex adaptive systems (CAS) in the sense of Casti's definition of CAS [1]. The neurones are individual components that can make some kind of computation. These neurones interact as they send the result of their computation to neighboring neurones, and they only use local information as they do not receive information from all the neurones in the network. For a detailed discussion of neural networks, we refer to Ref. [2], which previously appeared in *Complexity,* and for an extensive discussion of the results reported here we refer to Refs. [3], [4], and [5]. The following restrictions apply to the research reported on in this article. We only look at a relatively simple problem. Even though it could be argued that a simple problem such as the XOR problem is not representative of more realistic problems, the simplicity of the network allows us to better focus on the phenomenon we want to study, namely the occurrence of chaos during the learning process. We also restrict ourselves to multilayer neural networks that are trained, using the backpropagation algorithm. Similar analyses for other learning algorithms could be the subject of further research.

The article is structured as follows. We first briefly introduce the basic equations of the backpropagation algorithm (BPA). We then show that chaos occurs during the learning process and investigate in more detail the role of two parameters, the learning rate and the temperature. We finally address the question whether chaos increases learning speed and could therefore be considered necessary or beneficial for efficient learning.

ALGORITHMIC BASICS

We will not explain the entire algorithm here as it has already been discussed previously, but the remainder of the article centers around the so-called Delta rule. As we know, backpropagation neural networks belong to the category of supervised learning neural networks, implying that it is known what output the network has to produce for a given set of inputs. The following equations allow us to compute the output of any node (hidden or output) [6]:

FIGURE 1



Influence of temperature on the sigmoid transfer function.

As we know, backpropagation neural networks belong to the category of supervised learning neural networks, implying that it is known what output the network has to produce for a given set of inputs.

$$A_{pj} = \sum_{i=1}^{m} W_{ji} O_{pi} - U_j \qquad (1)$$

$$O_{pj} = f_j(A_{pj}) = \frac{1}{1 + e^{-A_{pj}/t}}, \qquad (2)$$

where A_{pj} is the activation for the input pattern p presented to node j, W_{ji} is the weight of the connections between nodes i and j, O_{pi} is the output of node i presented to node j as input, and U_j is the threshold of node j. The transfer function O_{pj} is also called the sigmoid transfer function. The parameter T is the temperature. As can be seen from Figure 1, the smaller its value, the more the transfer function looks like a step-function and the higher the temperature, the flatter it becomes.

The Delta rule computes the weight changes of the connections in function of the error in the computed output in the following way:

$$\Delta_p W_{ji} = \beta \delta_{pj} O_{pi} \qquad (3)$$

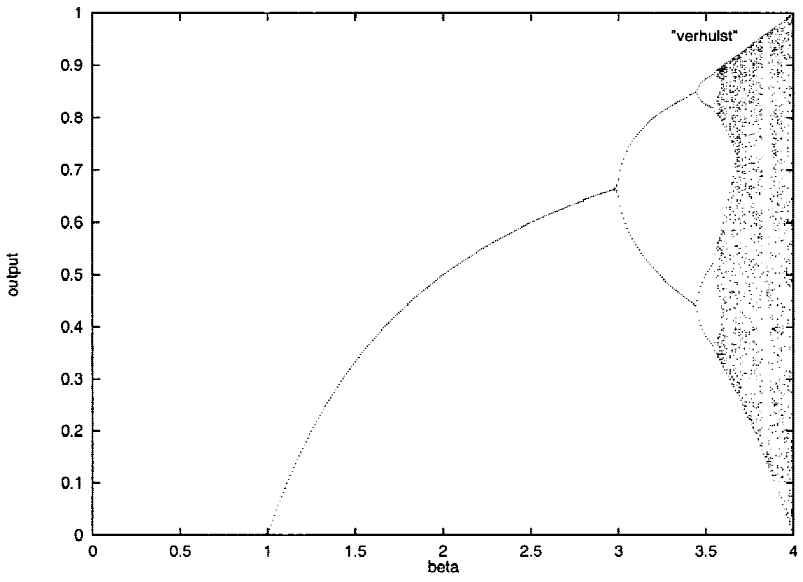
where O_{pi} is the value of the ith incoming connection, β is the learning rate, and δ_{pj} is the error at the jth node for the input pattern p. The error δ_{pj} needs to be calculated separately for the hidden and the output nodes.

For the output nodes, the error can be computed in the following way:

$$\delta_{pj} = (t_{pj} - O_{pj}) f'(A_{pj}), \qquad (4)$$

where the term $(t_{pj} - O_{pj})$ computes the difference between the expected and actual output of the net. This error is then multiplied by the first derivative of the transfer function. This derivative follows from the gradient descent equa-

FIGURE 2



Bifurcation diagram for the logistic equation.

TABLE 1.

Training set for the XOR problem

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

tions characterizing the BPA, but it can be interpreted in the following way: It allows to sanction nodes that generate an uncertain output (an activation value close to 0) because there the first derivative will be high and consequently the weight change will be larger. The inverse is true for nodes that generate large activation values.

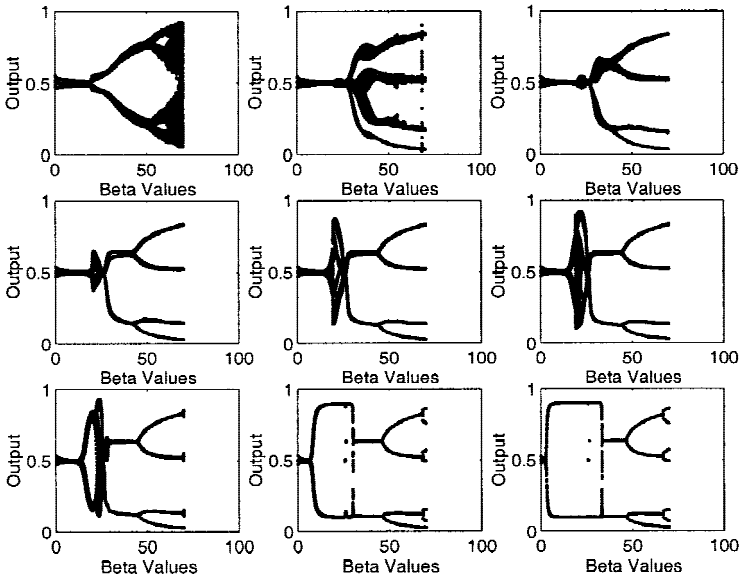
For the hidden nodes, the error can be computed in the following way:

$$\delta_{pj} = \left(\sum_{k=1}^{n} \delta_{pk} W_{kj} \right) f'(A_{pj}), \qquad (5)$$

where $\delta_{pk} W_{kj}$ represents the error of the connected neurons of the above layer, multiplied by the corresponding weights, which is propagated throughout the net. For exactly the same reasons as mentioned above, the first derivative of the transfer function is included.

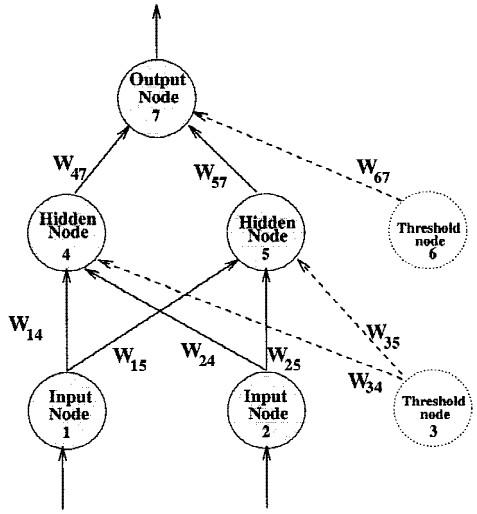
The two parameters that are important in these equations are β and *T*. β is the learning rate and specifies how much of the computed error will be taken into account for learning. Values for β can be any positive number. Typical values of β normally never exceed the value of 1 even though certain conditions may require higher values. In this article, we will explore a much wider range of β values. This is required because we want to study under what conditions chaos appears and disappears and what its influence is on the learning process. In Ref. [4] it was shown not only that chaos occurs for different values of β but also that the critical point, called β_c, at which chaos occurs differs in function of the value of the temperature parameter: The higher the temperature, the higher the value of β_c. We will now first explain why chaos occurs. We then discuss more in detail the role of two parameters, β and *T*.

FIGURE 4



A dynamic view on learning the XOR function (Reprinted with permission from Ref. [4]).

FIGURE 3

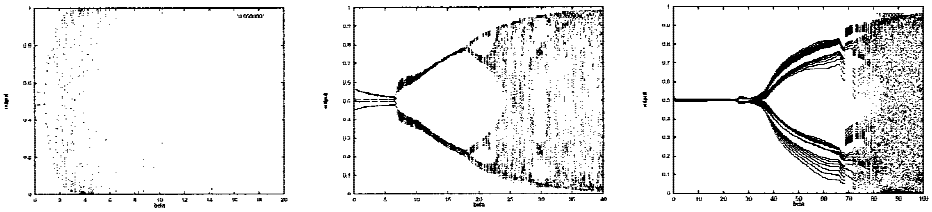


Architecture of the neural network.

WHY CHAOS OCCURS

Feigenbaum [7] defines the mathematical conditions for chaos to occur, which are that the dynamical system needs to be nonlinear and recursive. He also points out that the latter property is far more important than the specific operation being performed. In the BPA, the Delta rule contains the first derivative of the logistic or sigmoid transfer

FIGURE 5



Bifurcation diagram for T = 0.065, 0.35, and 1.25 (Reprinted with permission from Ref. [4]).

function. It is easy to see that $f'(A) = t^{-1}O_{pj}(1 - O_{pj})$. The recursive nature of the learning algorithm can be shown by rewriting

$$O_{pj} = f_j(A_{pj}) = \frac{1}{1 + e^{-\sum_{i=0}^{m} W_{ji}O_{pi}}} \tag{6}$$

using a Taylor series developed around 0, as

$$O_{pj}^{\text{new}} = [1/2 + 1/4 \sum_{i=0}^{n} W_{ji}O_{pi}] + 1/4\left(\sum_{i=0}^{n} O_{pi}^2\right)\beta\delta_{pj}O_{pj}(1 - O_{pj}) + \Theta(O_{pj}^6). \tag{7}$$

Even though the dependence of the right-hand side of this equation is more complicated than a simple quadratic map, we can see that Equation 7 is structurally related to the Verhulst equation known for its chaotic behavior, which is as follows:

$$x_{t+1} = \lambda x_t(1 - x_t). \tag{8}$$

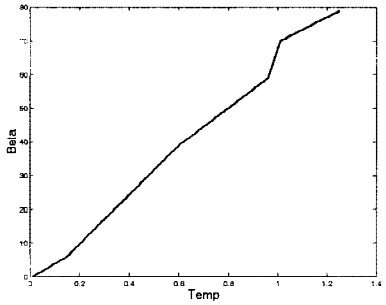
This Verhulst equation has different solutions, depending on the value of the one parameter in the equation, λ. These states are (1) a fixed point for values of λ between 0 and 3 and (2) a limit cycle for λ values between 3 and 3.5699999. This means that there are a limited number of states in which the system can be and through which the system iterates. If we have 4 different states, the system will start in state 1. The next time period, it will be in state 2 and then states 3 and 4, after which this iteration starts again. For any value beyond this critical value of λ, chaos occurs. The behavior of the equation can be visualized by means of the Feigenbaum bifurcation diagram, given in Figure 2. We explore this issue in the next section.

The Role of BETA

We conducted numerical experiments in which the network had to learn the XOR function (see Table 1 for the input sets and the corresponding target output). As depicted in Figure 3, the network has two input nodes: one hidden layer with two nodes and one node in the output layer. The two supplementary nodes represent the thresholds U_j. In order to establish whether bifurcations and chaos also appeared during the learning process, we ran different learning processes where each time the β value was changed. Each bifurcation diagram plots 200 computed outputs for each of the four input sets.

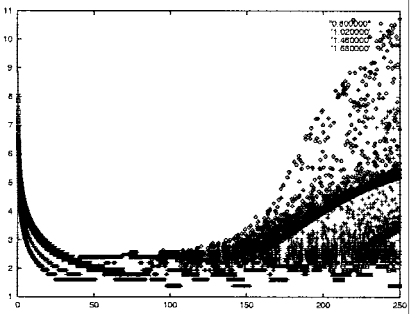
Figure 4 shows the bifurcation diagram where the different learning rates are plotted on the X-axis and the Y-axis represents the computed output. As we can observe from this figure (going from upper left to lower right), not only fixed points, limit cycles, bifurcations, and

FIGURE 6



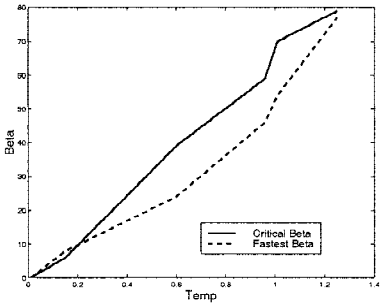
Different temperatures and the corresponding β_c (Reprinted with permission from Ref. [4]).

FIGURE 7



Required number of iterations for different T's and β's.

FIGURE 8



Different temperatures and the corresponding critical and fastest β values.

chaos appear for certain values of the learning parameter, but also that these phenomena disappear as learning progresses (which is of course to be expected). For a detailed account of these simulations, we refer to Ref. [4].

The Role of Temperature

We can now look at the second parameter in our equations, the temperature, in order to establish its influence on the above behavior. We ran another series of simulations in which the learning rate β was varied for different values of the temperature. The same neural network was used as above, which again had to learn the XOR function.

What we can observe from Figure 5 is that as the temperature increases, the bifurcation diagram shifts to the right. This implies that chaos will occur only for larger β values, and it seems to indicate a way to control the occurrence of chaos during the learning process. Analogously to the Verhulst equation, we can compute the value of the learning rate for which chaos first occurs, called the critical beta, β_c. When we plot these values in function of the different temperature values, we obtain a straight line, as can be seen in Figure 6. This seems to indicate that we can control the occurrence of chaos during the learning process by choosing an appropriate value of the temperature. This will be addressed in the remainder of the article.

DOES CHAOS INCREASE LEARNING SPEED?

The next logical step in our analysis is to find out whether chaos plays a positive role for learning and whether a parameterization strategy can be proposed. Do neural networks learn faster when using chaotic learning rates, and do we therefore need to push the network in a chaotic regime? This final series of simulations involved again the XOR function, and we computed the number of iterations the network needed to converge for any given pair of β and *T*. The result is shown in Figure 7. The β values are plotted on the abscissa, and the number of iterations required for convergence is plotted on the ordinate (which is expressed in thousands). The following can be concluded:

1. The network converges for a wide range of learning rates, going from 0.1 to 68.
2. As the temperature increases, the average number of iterations needed to converge increases also.
3. For any given temperature, the U-shaped curve indicates that for very low as well as for very high learning rates, a higher number of iterations is required.
4. There is a high variability in the different convergence rates when using very high βs. For these β values, there is always chaos.
5. In Figure 8, we plotted the curve of those tuples (β, *T*) for which there was fastest convergence together with the critical β values of Figure 6. Even though many more simulations are required to draw statistically sound conclusions, we can see that fastest convergence is obtained for β values corresponding almost always to the chaotic regime. For very low temperature values, the critical learning rates are close to the fastest ones. Only there could we say that learning takes place on the edge of chaos.

Do neural networks learn faster when using chaotic learning rates, and do we therefore need to push the network in a chaotic regime?

CONCLUSIONS

Even though we do not claim to advance a general statement on neural networks with respect to chaos and learning, the following seems to hold as suggested by our numerical experiments.

As the temperature increases, the β value has to be increased proportionally for chaos to occur during the learning process. Our simulations suggest that fastest learning appears more often for "chaotic" β values. However, the irregularity in the chaotic regime does not guarantee that fastest convergence will always be achieved.

REFERENCES

1. Casti, J. Complexity 1996, 1(5), 7.
2. Picton, P.; Johnson, J. Complexity 1996, 1, 13.
3. Bertels, K.; Neuberg, L.; Vassiliadis, S.; Pechanek, G. Xor and backpropagation: In and out of chaos? In European Symposium on Artificial Neural Networks, 1995, 69–74.
4. Bertels, K.; Neuberg, L.; Vassiliadis, S.; Pechanek, G. Neural Processing Lett. 1998, 7, 69.
5. Bertels, K.; Neuberg, L.; Vassiliadis, S.; Pechanek, G. Artificial Intelligence Rev, to appear.
6. McClelland, R. Parallel Distributed Processing: Explorations in the microstructure of cognition. MIT Press: Cambridge, MA, 1988.
7. Feigenbaum. Los Alamos Sci. 1980, 4.