The Influence of the Sigmoid Function Parameters on the Speed of Backpropagation Learning

Jun Han Claudio Moraga

Research Group Computational Intelligence
Dept. of Computer Science, University of Dortmund
D-44221 Dortmund, Germany

Abstract

Sigmoid function is the most commonly known function used in feed forward neural networks because of its nonlinearity and the computational simplicity of its derivative. In this paper we discuss a variant sigmoid function with three parameters that denote the dynamic range, symmetry and slope of the function respectively. We illustrate how these parameters influence the speed of backpropagation learning and introduce a hybrid sigmoidal network with different parameter configuration in different layers. By regulating and modifying the sigmoid function parameter configuration in different layers the error signal problem, oscillation problem and asymmetrical input problem can be reduced. To compare the learning capabilities and the learning rate of the hybrid sigmoidal networks with the conventional networks we have tested the two-spirals benchmark that is known to be a very difficult task for backpropagation and their relatives.

1 Introduction

A sigmoid function is a bounded differentiable real function that is defined for all real input values and that has a positive derivative everywhere. It shows a sufficient degree of smoothness and is also a suitable extension of the soft limiting nonlinearities used previously in neural networks. The backpropagation algorithm also focused attention on it due to the existence of theorems which guarantee the so called "universal approximation" property for neural networks. Hornik and White showed that feedforward sigmoidal architectures can in principle represent any Borel-measurable function to any desired accuracy, if the network contains enough "hidden" neurons between the input and output neuronal fields [1] [2]. However, backpropagation learning is too slow for many applications and it scales up poorly as tasks become larger and more complex. The factors governing learning speed are poorly understood [3]. A number of people have analyzed the reasons why backpropagation learning is so slow, such as step size problem, moving target problem and all the problems that occur with gradient descent (See e.g. [3],[4],[5]). In this paper we introduce a variant sigmoid function with three parameters that denote the dynamic range, symmetry and slope of the function respectively and a hybrid sigmoidal network: *a fully feedforward network with the variant sigmoid function and with different parameters configuration in different layers*. By regulating these three parameters in different layers we can reduce the attenuation and dilution of the error signal problem, the oscillation problem and the asymmetrical input problem that contribute also to the slowness of backpropagation learning. Results on the two-spirals benchmark are presented which are better than any results under backpropagation feed forward nets using monotone activation functions published previously.

2 Three Parameters of the Variant Sigmoid Function

A standard choice of sigmoidal function, depicted in Fig. 2.1, is given by [6],[7]

$$f(h) = \frac{1}{1+\exp(-2\beta h)}$$

which satisfies the condition $f(h) + f(-h) = 1$. Further, a sigmoidal function is assumed to rapidly approach a fixed finite upper limit asymptotically as its argument gets large, and to rapidly approach a fixed finite lower limit asymptotically as its argument gets small. The central portion of the sigmoid (whether it is near 0 or displaced) is assumed to be roughly linear. Among physicists this function is

commonly called the Fermi function, because it describes the thermal energy distribution in a system of identical fermions. In this case the parameter β has the meaning of an inverse *temperature*, $T = \beta^{-1}$ [8].
The form of the variant sigmoid function is given as

$$f(x, c_1, c_2, c_3) = \frac{c_1}{1+\exp(-c_2 x)} - c_3$$

where

- c_1: Parameter for the dynamic range of the function; (See Fig. 2.2)
- c_2: Parameter for the slope of the function; (See Fig. 2.3)
- c_3: Parameter for the symmetry (or bias) of the function. (See Fig. 2.4)

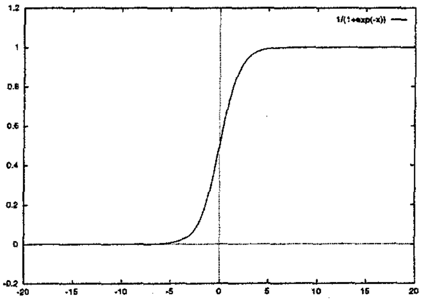


Figure 2.1

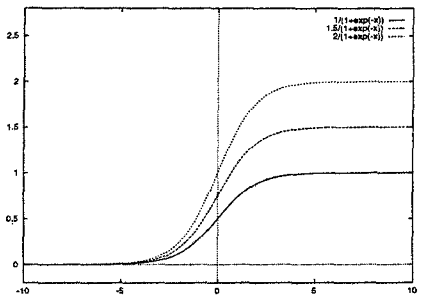


Figure 2.2

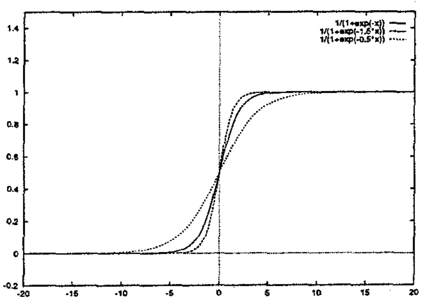


Figure 2.3

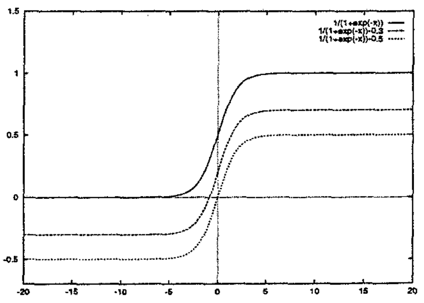


Figure 2.4

Notice that if $c_3 = \frac{c_1}{2}$,

$$f(x, c_1, c_2, c_3) = \frac{c_1}{1+\exp(-c_2 x)} - \frac{c_1}{2}$$

becomes a symmetric (hyperbolic) function. Note that the hyperbolic tangent is just the standard sigmoid function biased and rescaled, as shown by

$$f(x, c_1, c_2, c_3) = \frac{c_1}{1+\exp(-c_2 x)} - \frac{c_1}{2} = \frac{c_1}{2}\left[\frac{1-\exp(-c_2 x)}{1+\exp(-c_2 x)}\right]$$

$$= \frac{c_1}{2}\left[\frac{\exp(+\frac{c_2}{2}x)-\exp(-\frac{c_2}{2}x)}{\exp(+\frac{c_2}{2}x)+\exp(-\frac{c_2}{2}x)}\right] = \frac{c_1}{2}\tanh(\frac{c_2}{2}x)$$

3 The Influence on the Speed of Backpropagation Learning

Many experiments have reported that backpropagation learning slows down dramatically (perhaps exponentially) as we increase the number of hidden layers in a network. This slowdown is partly due to an *attenuation and dilution* of the error signals as they propagate backwards through the layers of the network. The other reason is the so called "moving target effect" and it is well solved through the Cascade-correlation learning architecture [4]. One way to reduce the attenuation and dilution of the error signal problem is to decrease the number of hidden layers and increase the number of hidden units in a given hidden layer of the net, because in a given layer of the net the hidden units cannot communicate with one another directly. The other way is to enlarge the error signals in different hidden layers respectively. That is: *the more a hidden layer is far from the output layer, the more we have to enlarge the error signals.*

First, let us consider the standard backpropagation algorithms [9]. In general, with an arbitrary number of layers, the backpropagation update rule always has the form

$$\Delta w_{pq} = \eta \sum_{patterns} \delta_{output} \times \Gamma_{input}$$

where *output* and *input* refer to the two ends p and q of the connection under consideration and Γ stands for the approriate input-end activation from a hidden unit or a real input. The meaning of δ depends on the corresponding layer. For the last layer of connections δ is given as

$$\delta_i^\mu = f'(h_i^\mu)[\xi_i^\mu - O_i^\mu] \qquad (3.1)$$

while for all other layers it is given as shown below

$$\delta_j^\mu = f'(h_j^\mu)\sum_i W_{ij}\delta_i^\mu \qquad (3.2)$$

Equation (3.1) allows us to determine δ for a given hidden unit Γ_j in terms of the δ's of the units O_i that it feeds. Meanwhile in equation (3.2) the coefficients are just the usual "forward" W_{ij}'s, but here they are propagating errors (δ's) backwards instead of signals forwards. Both in (3.1) and (3.2) $f'(h_i^\mu)$ denotes the derivative of the activation function of the corresponding unit. We can simply change the parameter c_1 or c_2 of different hidden layers to enlarge the error signals respectively.

C_2 can also be used to reduce the oscillation problem. Because the output layer tends to have larger local gradients than hidden layers, the learning step size should be smaller in the output layer than in the hidden layers. From (3.1) we can see that by changing parameter c_2 in the output layer we can control the learning step size and therefore reduce the oscillation effectively.

C_3 can displace the function up or down along the Y axis and can therefore eliminate the problems arised by the asymmetrical way in which the two states (0's and 1's) of the input are treated.

From (3.2) we can see

$$\delta_i^\mu = f'(h_i^\mu)\sum_i W_{ij}\delta_i^\mu = f(h_i^\mu)[1 - f(h_i^\mu)]\sum_i W_{ij}\delta_i^\mu$$

when $f(h_i^\mu) = 1$ or $f(h_i^\mu) = 0$, we have $f(h_i^\mu)[1 - f(h_i^\mu)] = 0$ and $\delta = 0$.

Thus, for some typical pattern only half of the weights from the input to the hidden layer will be modified, if $c_3 = 0$.

A hybrid sigmoidal hetwork is therefore formed by using different parameter configuration of sigmoid functions in different layers. Let us consider an N-layers hybrid net with n=1,2,...,N.

We use Γ_i^n for the output of the ith unit in the nth layer. Γ_i^0 will be a synonym for ξ_i, the ith input. Note that superscript n denotes layers, not patterns. Let w_{ij}^n mean the connection from Γ_j^{n-1} to Γ_i^n. Then the backpropagation learning procedure is the following:

1. Initialize the weights with random values to avoid the symmetry breaking.

2. Choose a patten ξ_m^μ and apply it to the input layer (n=0) so that

$$\Gamma_m^o = \xi_m^\mu \qquad \text{for all } m \qquad (3.3)$$

3. Propagate the signal forwards through the network using
$$\Gamma_i^n = f(h_i^n) = f\left(\sum_j w_{ij}^n \Gamma_j^{n-1}\right) \quad (3.4)$$
for each i and n until the final outputs Γ_i^N have all been calculated.

4. Compute the deltas for the output layer
$$\delta_i^N = f'(h_I^{n-1})[\xi_i^\mu - \Gamma_i^N] \quad (3.5)$$
by comparing the actual outputs Γ_i^N with the desired ones ξ_i^μ for the pattern μ being considered.

5. Compute the deltas for the preceding layers by propagation the errors backwards
$$\xi_i^{n-1} = f'(h_i^{n-1}) \textstyle\sum_j w_{ji}^n \delta_j^n \quad (3.6)$$
for $n = N, N-1, ..., 2$ until a delta has been calculated for every unit.

6. Use
$$\Delta w_{ij}^n = \eta \delta_i^n V_j^{n-1} \quad (3.7)$$
to update all connections according to $w_{ij}^{new} = w_{ij}^{old} + \Delta w_{ij}$.

7. Go back to step 2 and repeat for the next pattern.

From (3.6) it is straightforward shown that by adopting different parameters in different hidden layers (i.e. tuning the slopes or dynamic ranges of the functions) we can enlarge the deltas for every hidden layer respectively. Similarly, we can also see from (3.5) that in the same way we can adjust the deltas for the output layer and regulate the step size further.

4 Two-Spirals Benchmark Test

The two-spirals problem is a benchmark to test the behavior of a network. The task is to learn to discriminate between two sets of training points (194 points) which lie on two distinct spirals in the x-y plane. These spirals coil three times around the origin and around one another. (See Fig. 4.4a) This benchmark was chosen as the primary benchmark for this study because it is an extremely hard problem to solve for algorithms of the backpropagation family [4].
To show the behavior of the hybrid net clearly, we illustrate a small two-spirals problem that consists of 100 points. (See Fig. 4.1a)

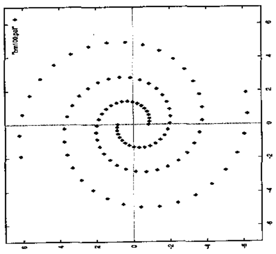


Figure 4.1a

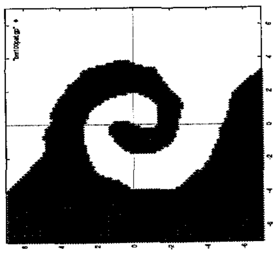


Figure 4.1b

(Figure 4.1: (a) Training points for the small two-spirals problem
(b) Output pattern for one trained hybrid net)

To solve this problem, we build a sigmoidal hybrid net. The structure and the activation functions are given as follows:

$$2 - 65_{N-2} - 65_{N-1} - 1_N$$

where

$$f(h^{N-2}) = \frac{2}{1+exp(-4h)} - 1 \qquad (c_1{=}2,\ c_2{=}4,\ c_3{=}1)$$

$$f(h^{N-1}) = \frac{1}{1+exp(-1.8h)} \qquad (c_1{=}1,\ c_2{=}1.8,\ c_3{=}0)$$

$$f(h^{N}) = \frac{1}{1+exp(-0.8h)} \qquad (c_1{=}1,\ c_2{=}0.8,\ c_3{=}0)$$

The derivatives of the activation functions for different layers are given as follows:

$$f'(h^{N-2}) = 8\left(\frac{1}{1+exp(-4h)}\right)\left(1 - \frac{1}{1+exp(-4h)}\right)$$

$$f'(h^{N-1}) = 1.8\left(\frac{1}{1+exp(-1.8h)}\right)\left(1 - \frac{1}{1+exp(-1.8h)}\right)$$

$$f'(h^{N}) = 0.8\left(\frac{1}{1+exp(-0.8h)}\right)\left(1 - \frac{1}{1+exp(-0.8h)}\right)$$

Fig. 4.2 shows that the ranges and slopes of the derivatives by these functions are enlarged correspondingly. This net solves the small two-spiral problem in less than 50 epochs. (See Fig. 4.1b and Fig. 4.3(e)) Now we compare it with another four conventional nets:

- $2 - 65_{N-2} - 65_{N-1} - 1_N$ with $f(h^{N-2}) = \tanh(2h)$ for every layer
- $2 - 65_{N-2} - 65_{N-1} - 1_N$ with $f(h^{N-1}) = \frac{1}{1+exp(-1.8h)}$ for every layer
- $2 - 65_{N-2} - 65_{N-1} - 1_N$ with $f(h^{N}) = \frac{1}{1+exp(-0.8h)}$ for every layer
- $2 - 65_{N-2} - 65_{N-1} - 1_N$ with standard sigmoid function for every layer

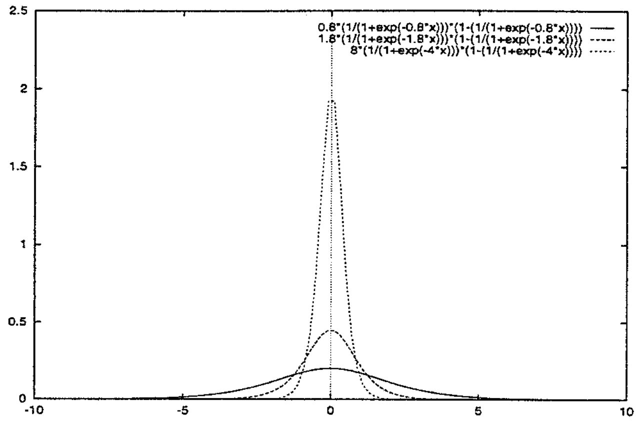
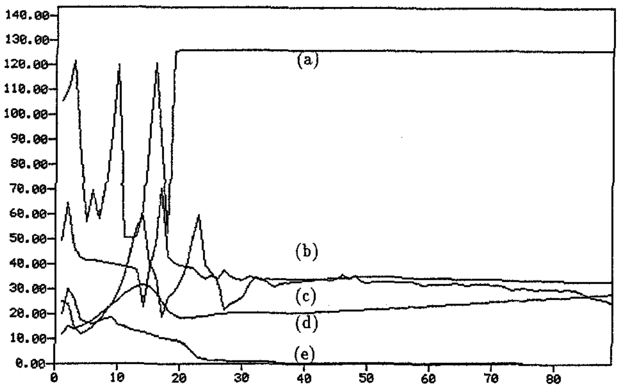


Figure 4.2
(Figure 4.2: Derivative of the activation functions for different layers)

Fig. (4.3) shows the comparison clearly:



(Figure 4.3: Comparative results for small two-spirals problem from top to bottom: (a) Feed forward net using tanh as activation function (b) Feed forward net using a sigmoid activation function with maximum slope 1.8 (c) Feed forward net using the standard sigmoid function (d) Feed forward net using a sigmoid activation function with maximum slope 0.8 (e) The hybrid net)

Finaly, we build a hybrid net to solve the original two-spirals problem:

$$2 - 130_{N-2} - 130_{N-1} - 1_N$$

where

$$f(h^{N-2}) = \frac{3.2}{1+exp(-2h)} - 1.6 \qquad (c_1{=}3.2,\ c_2{=}2,\ c_3{=}1.6)$$

$$f(h^{N-1}) = \frac{1}{1+exp(-1.2h)} \qquad (c_1{=}1,\ c_2{=}1.2,\ c_3{=}0)$$

$$f(h^{N}) = \frac{1}{1+exp(-0.7h)} \qquad (c_1{=}1,\ c_2{=}0.7,\ c_3{=}0)$$

This net solves the two-spirals problem in 900 epochs (average over 8 trials by shuffle, all successful) (See Fig. 4.4b and Fig. 4.5). Table (4.1) shows training epochs necessary to solve the two-spiral problem by backpropagation networks and their relatives. It is fair to mention, that the comparison does not include the computational effort of a genetic algorithm to optimize the parameters c_1, c_2 and c_3 for each layer.

network model	number of epochs	reported in
Backpropagation	20000	Lang and Witbrock (1989)
Cross Entropy BP	11000	Lang and Witbrock (1989)
Cascade-Correlation	1700	Fahlman and Lebiere (1990)
Hybrid net under BP	900	This paper

Table 4.1

(Tab.4.1: Training epochs necessary for the original two-spiral problem with backpropagation networks and their relatives)

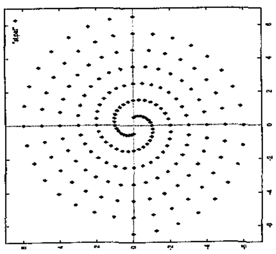


Figure 4.4a

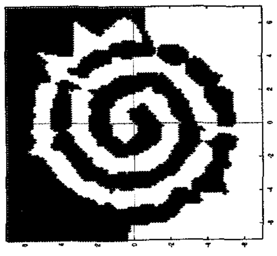


Figure 4.4b

(Figure 4.4: (a) Training points for the original two-spirals problem
(b) Output pattern for one trained hybrid net)

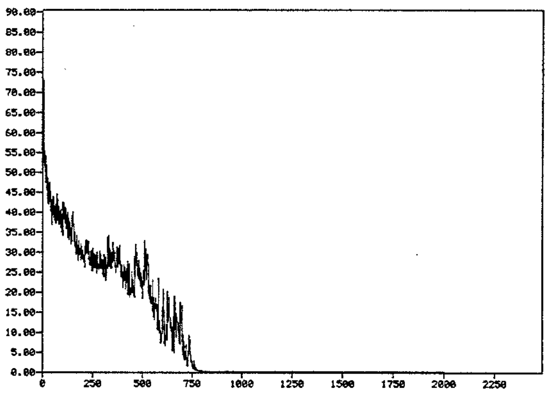


Figure 4.5
(Figure 4.5: Test result of the original two-spirals problem for one hybrid net)

5 Closing Remarks

In this paper we have illustrated how the sigmoid function parameters influence the speed of backpropagation learning and introduced a hybrid sigmoidal network with different parameters configuration in differente layers. In this way it is possible to reduce the attenuation and dilution of the error signal problem, the oscillation problem and the asymmetrical input problem.
Finally it should be mentioned that the goal of this study was the improvement of the learning speed in backpropagation nets with monotone activation functions and not the reduction of the number of nodes. Further improvements are however expected by extending the former strategy from the layer level to the node level. It may be seen that a good parameter configuration may provide a basis for the achievement of greater performance of fast learning under backpropagation.

6 References

[1] Hornik, K., Stinchcombe, M., and White, H.: Multilayer Feedforward Networks are Universal Approximators. Neural Networks, Vol. 2, no. 5, 359-366, (1989)

[2] Kosko, B.: Neural Networks and Fuzzy Systems. Prentice-Hall, INC. (1992)

[3] Fahlman, S.E.: An Empirical Study of Learning Speed in Back-Propagation Networks. Technical Report CMU-CS-88-162, CMU, (1988)

[4] Fahlman, S.E., Lebiere, C.: The Cascade-Correlation Learning Architecture. in Touretzky (ed.) Advances in Neural Information Processing Systems 2, Morgan-Kaufmann, (1990)

[5] Stornetta, W.S., Huberman, B.A.: An improved Three-Layer, Back Propagation Algorithm. IEEE First Int. Conf. on Neural Networks, (1987)

[6] Little, W.A.: The Existence of Persistent States in the Brain. Math. Biosci. 19, 101, (1974)

[7] Little, W.A., Shaw, G.L.: Analytic Study of the Memory Capacity of a Neural Network. Math. Biosci. 39, 281, (1978)

[8] Müller, B., Reinhardt. J.: Neural Networks. Springer-Verlag, (1990)

[9] Hertz, J., Krogh, A., Palmer, R.G.: Introduction to the theory of neural computation. Addison-Wesley publishing Company, (1991)