Neural networks with periodic and monotonic activation functions: a comparative study in classification problems

Josep M. Sopena
Lab. Neurocomputació
Universitat de Barcelona
jsopena@psi.ub.es

Enrique Romero
Departament de LSI
Univ. Politècnica Catalunya
eromero@lsi.upc.es

René Alquézar
Departament de LSI
Univ. Politècnica Catalunya
alquezar@lsi.upc.es

Abstract

This article discusses a number of reasons why the use of non-monotonic functions as activation functions can lead to a marked improvement in the performance of a neural network. Using a wide range of benchmarks we show that a multilayer feed-forward network using sine activation functions (and an appropriate choice of initial parameters) learns much faster than one incorporating sigmoid functions - as much as 150-500 times faster - when both types are trained with backpropagation. Learning speed also compares favorably with speeds reported using modified versions of the backpropagation algorithm. In addition, computational and generalization capacity increases.

1 Introduction

The term "classification" covers a wide range of problems in Artificial Intelligence. In general, given a finite number of variables which are assumed to characterize the problem in question, and a finite number of classes, a classification system assigns a particular class to each set of values of the variables. Hence, classification can be understood as the application of a function with a domain in the space of variables and image in the space of classes. In many real problems, these functions are unknown, and they must be approximated from a set of given points.

Neural networks offer a method for approximating functions. Goodness of fit is calculated on the basis of a measurement of the approximation error, known as the error function. The quadratic error function is the most usual one.

A learning algorithm aims to lead the system from an initial state (in which the values of the parameters are randomly assigned) to a minimum of error. Once this is achieved, the network's generalization capacity is measured by its ability to calculate correctly the values of the function at points that do not belong to the training set. The most frequently used learning algorithm is backpropagation (BP).

A good learning algorithm must have the following properties (among others): fast learning speed, computational capacity (approximation to complex functions) and generalization capacity.

[Fahlman and Labiere, 1990] states that BP's lack of speed is one of the main reasons why neural networks have not been more widely used. In addition, although in theory a neural network can approximate any function, in practice finding the right set of parameters may be extremely difficult. Finally, in certain problems, even if the training set is learnt, the generalization is unsatisfactory. Several modifications of BP have been proposed in an attempt to improve some of these properties: "Quickprop" [Fahlman, 1988], "Supersab" [Tollenare, 1990], "Cascade Correlation" [Fahlman and Labiere, 1990], "RPROP" [Riedmiller and Braun, 1993], etc.

None of these proposals have considered modifying the activation functions. The most frequently used activation functions are monotonic, such as the sigmoid function and the hyperbolic tangent. Recent years have seen the introduction of Radial Basis Functions Networks (RBFN), which use non-monotonic functions such as Gaussian functions.

This study compared the behavior of multilayer perceptrons with periodic activation functions and those with monotonic activation functions. In our experiments, the sine was the periodic activation function. A wide range of benchmarks indicated that the behavior of periodic functions was clearly superior, in terms of learning speed, computational capacity and generalization.

2 Periodic vs. monotonic activation functions

[Yoshizawa *et al.*, 1993] demonstrated formally that the storage capacity of an associative memory with non-monotonic activation functions can be several times greater than the capacity of the same architecture with monotonic activation functions. One might think - although this has not been formally proven - that the same property may hold for multilayer perceptrons. As multilayer perceptrons are highly non-linear, demonstrations of this type are difficult to obtain. However, there are a number of reasons for thinking that the choice of non-monotonic functions as activation functions - in particular, sine functions - can produce a notable improvement in the performance of a neural network.

A classifier acts by dividing the space of variables or input space into regions, so that in each region there are only representatives of one class. In the case of a neural network, each unit contributes to dividing the input space in a particular way. It is well known that units with monotonic activation functions generate a single hyperplane which divides the input space into two regions. The combination of several of these hyperplanes can give rise to complex classifications. A unit with a non-monotonic activation function can divide the space into more than two regions. If the function is periodic, the number of regions is infinite, and it can be interpreted as a wave front spreading through the space of variables.

For a clearer idea of how sine functions behave, let us imagine a two-dimensional space such as the one in figure 1. In this space a function $sin\,(\vec{\omega}\vec{x} + bias)$ represents a sinusoidal surface whose contours of equal height are given by $\vec{\omega}\vec{x} = constant$. The separation between two consecutive contours is related to the wavelength $2\pi/\sqrt{\vec{\omega}^2}$. The greater the weights, the lesser the distance between two levels of the same height, and therefore the input space will be divided into smaller regions. The bias causes a shift of the surface.

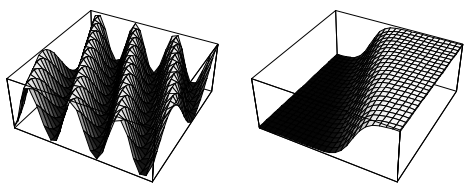

Figure 1: Comparison between the sine and sigmoid functions in a two-dimensional space for the same argument ($\vec{\omega}\vec{x} = 5x + 2y$).

Let us imagine a feed-forward network with a single layer of hidden units with sine activation functions and a linear output unit. This network will approximate a function by effecting a superposition of sinusoidal functions:

$$g\,(\vec{x}) = bias_o + \sum_{i=1}^{nhid} \alpha_i \sin\,(\vec{x}\vec{\omega}_i + bias_i)\,.$$

We will have as many sinusoidal surfaces as there are units in the hidden layer ($nhid$). The characteristics of each of these surfaces are given by the weights of the connections that the corresponding hidden units receive. We will call these weights non-linear parameters ($\vec{\omega}_i$). These different sinusoidal surfaces combine linearly to produce the output. The parameters that adjust this linear combination are the weights of the connections between the hidden units and the output units. We will call these parameters linear parameters (α_i). The biases can also be considered as weights with a simple transformation.

Intuitively, it appears that by combining several of these units with periodic functions we should obtain more capacity to divide the space into differentiated regions than with sigmoidal functions (see figure 1); it should therefore be possible to approximate more complex functions. Training a neural network with sine functions is, in essence, constructing a Fourier series ([Lapedes and Farber, 1987], [Barron, 1993]). However, in contrast to a standard Fourier representation, the network has the ability to adjust the values of the non-linear parameters (the frequencies).

However, networks with sine functions can yield a much larger number of local min-

ima in the error surface than networks with sigmoid functions. This is because the activation function is non-monotonic, as in the case of the RBFN. An algorithm based on the correction of error, such as BP, can easily be trapped in a local minimum before it reaches a global minimum. Consequently, with non-monotonic activation functions it is difficult to make the weights vary sufficiently from their initial position. A solution proposed for the RBFN [Lowe, 1991] is to assign an initial set of weights that is appropriate for the problem in question. In this way the region of the parameter space to be explored is delimited. With appropriate initial assignations, the probability of falling in a local minimum is substantially diminished.

Another important factor is the different time scales in which the network's parameters evolve. Non-linear parameters are difficult to adjust, and evolve at a much slower pace than linear parameters, which adjust much more easily [Lowe, 1991]. This means that, in order to obtain satisfactory performance, it could be convenient to assign different values of learning rates for each layer: large in the first layer, and small in the second (see "Sonar" problem).

In the benchmarks that follow we will check empirically the aspects of the behavior of the networks with sine activation functions. We will show that when the sine function initially has a wide, appropriate range of values of weights, its behavior is clearly superior to that of the sigmoid function. We will also show that in general the behavior of the sine function is not very different from that of the sigmoid function if the initial values of the weights are not suitable.

3 Experiments

The benchmarks we used have been frequently studied elsewhere. Most of them are taken from the *NN-Bench Collection* of the University of Carnegie Mellon.

3.1 General method

We now describe the training conditions, and comment on some of the criteria used in the experiments.

The learning speed is measured as the number of epochs that the network needs to learn the training set. Training is considered complete (the stop criterion) when all the examples have been learnt up to a given tolerance. In some cases a different criterion is used, because learning all the examples is impossible, or even not convenient for generalization purposes. We used the pattern learning mode (weights are modified after the presentation of each example). Each experiment was conducted several times until ten successful runs were achieved for each of the configurations of initial parameters. In the result tables, the column '%sr' indicates the percentage of successful runs as regards the total number of executions, the column "epochs" indicates the average number of learning epochs for successful runs and the column "gener." shows the average percentage of generalization for successful runs.

The architecture used was feed-forward with a layer of hidden units in all cases. The number of units in each layer is indicated by the expression I-H-O, where I, H and O are the number of units in the input, hidden and output layers respectively. Since the functions to be approximated in all the experiments have only two possible values, a sigmoidal activation function in the neurons of the output layer was used in all of them. The weights were initially assigned randomly to a particular interval ($-wrange/2$ and $wrange/2$). As we said above, the value of this range can vary for each layer of weights. This is shown in the results tables as follows: I-H:*wrange* refers to the weights in the first layer, and H-O:*wrange* to those of the second.

To simplify the initial parameter fit, momentum was set at 0. With this value, adequate performance was achieved in all the problems. Bias in the hidden layer was only used in the case of the "Quadrants". If we have no bias in a hidden layer of sinusoidal units, the network can only generate odd-like functions. One way of converting the target function in an odd function is to rescale all the variables in the input space between 0 and 1. We use this transformation in all cases except the "Two spirals" problem and the "Quadrants".

3.2 Two spirals problem

The "Two spirals" problem has been perhaps the most widely used benchmark in order to test the effectiveness of new learning algorithms, or proposed modifications of BP. It consists in identifying the points of

two interlocking spirals. The training set comprises 194 points.

Standard BP with a simple architecture has not found a solution to this problem (see [Fahlman and Labiere, 1990]). [Lang and Witbrock, 1988] solved it in 20,000 epochs using standard BP with a complex architecture (2-5-5-5-1 with shortcuts). "Supersab" needed an average of 3,500 epochs, "Quickprop" 8,000, "RPROP" 6,000, and "Cascade Correlation" 1,700. To solve this problem we constructed a network with architecture 2-16-1, using the sine as the activation function in the hidden layer, and the hyperbolic tangent in the output layer. This architecture is the simplest of those used to date to deal with this problem. Results are shown in table 1. The importance of the range of initial weights is clear. With small ranges for the non-linear parameters learning is not possible (see two first rows in table 1).

wrange	%sr	epochs	gener.
I-H:1 H-O:1	0(51.6)	-	-
I-H:3 H-O:1	0(79.9)	-	-
I-H:7 H-O:1	46(96.3)	46.0	97.6%
I-H:7 H-O:3	15(96.3)	237.5	94.7%
I-H:7 H-O:0.001	72(98.5)	42.3	98.3%

Table 1: Results for the "Two spirals" problem, with different ranges of initial weights and learning rate 0.006. In the column '%sr' the second figure shows the average percentage of examples learnt for unsuccessful runs.

The supposition that the linear parameters have a much faster temporal evolution than non-linear parameters and should therefore be initialized in a different way is also confirmed by the results shown in table 1. With the correct range, the learning speed obtained with sine functions (and with a simpler architecture) is 40 times greater than the fastest current method.

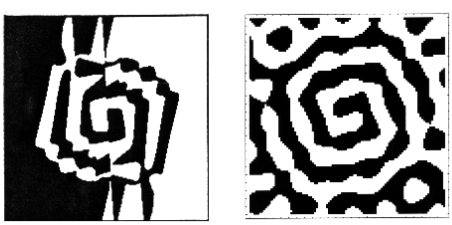

Figure 2: Generalization obtained by [Fahlman and Labiere, 1990] (a) and generalization obtained with sine functions (b).

The generalization obtained by "Cascade Correlation" can be seen in figure 2a. It is clear from figure 2b that the generalization obtained by a multilayer perceptron with sine functions is better.

3.3 Negation N

In this problem the input patterns have N+1 bits (binary coded). The last bit indicates whether the output is to be a copy of the N first bits of the input or their negation (1-complement).

In [Rumelhart *et al.*, 1986], with a 4-3-3 architecture with sigmoids, the network required some 5,000 epochs to learn the function. We constructed the same architecture using the sine as the activation function of the hidden layer and the hyperbolic tangent in the output layer. The results are shown in table 2. This case confirms again the importance of the ranges of the initial weights.

wrange	%sr	epochs
I-H:1 H-O:1	57	161.0
I-H:3 H-O:1	66	33.3
I-H:5 H-O:1	81	58.6
I-H:3 H-O:0.05	90	29.3
I-H:5 H-O:0.05	95	30.0

Table 2: Learning results for the "Negation N" problem with different ranges of initial weights and learning rate 0.04.

3.4 Separation problem

[Yang *et al.*, 1993] presented a classification problem that poses extreme difficulties for standard BP. In this problem, as in the "Two spirals" situation, standard BP is trapped in local minima. To avoid this, [Yang *et al.*, 1993] apply a method (BP with homotopy) that reaches a solution after 5,172 epochs. With the same architecture as in [Yang *et al.*, 1993] (2-12-1) but using sines instead of sigmoids, we achieved learning around 180 times faster (see table 3). Again, the assumption concerning the ranges of the weights appears to be correct.

wrange	%sr	epochs
I-H:8 H-O:1	0	-
I-H:16 H-O:1	71	2504.9
I-H:16 H-O:0.1	59	1431.9
I-H:32 H-O:0.1	43	66.0
I-H:64 H-O:0.01	91	27.5

Table 3: Learning results for the "Separation problem" with different ranges of initial weights and learning rate 0.02.

3.5 Quadrants

It might be thought that not all functions are adequately approximated using sines as activation functions. Might there not be functions for which the sigmoids may be better suited than sines?.

We designed a function that we call "Quadrants". In this problem the points of the first and third quadrants have target 1, and those of the second and fourth quadrants target 0. This problem intuitively seems to be tailor-made for sigmoids, as in order to obtain the correct classification it is only necessary to generate a few number of hyperplanes. An activation function such as the sine, which can generate an infinite number of hyperplanes, does not seem suited to solving this type of problem.

The results are shown in table 4. Using the sine as activation function in the hidden layer and the sigmoid in the output layer, the training set was learnt with an architecture 2-2-1. For sigmoid functions, the training set was learnt with 4 hidden units (last row in table 4). Besides the advantage in terms of the number of hidden units, with the sine function learning is much faster than with the sigmoid function. Generalization is also better.

arch.	wrange	%sr	epochs	gener.
2-2-1	I-H:1 H-O:1	91	433.4	100%
2-2-1	I-H:3 H-O:1	62	187.4	100%
2-2-1	I-H:7 H-O:1	77	95.3	100%
2-4-1	I-H:1 H-O:1	83	826.5	97.5%

Table 4: Learning results for the "Quadrants" with different ranges of initial weights and learning rate 0.1 (sines-sigmoids) and 0.4 (sigmoids). The initial range of bias is 1.

3.6 Diabetes

We performed another comparison with a problem of medical diagnosis, using the "Pima Indians Diabetes Database" produced in the Applied Physics Laboratory, Johns Hopkins University, 1988.

To approach this problem we constructed a network with an 8-12-1 architecture. Three separate partitions of the set of examples were made, to construct three different training sets and their corresponding test sets. In each of the partitions the test sets contained 25% of the examples. The first partition was used to set the optimal learning rate and stop criterion. The stop criterion was fixed at successful learning of 80% of the training set. Once these parameters were set, the network was trained and a test conducted with the sets of the other two partitions. Using sigmoid functions the generalization performance obtained was 72.3%, and with sine functions 78.8%. In further experiments to compare computational capacity, sigmoid functions were unable to learn more than 82% of the learning set correctly after 70,000 epochs, while the sine functions learnt more than 90% in around 1,500 epochs.

3.7 Sonar

In this problem the task is to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock, as described in [Gorman and Sejnowski, 1988].

[Gorman and Sejnowski, 1988], using sigmoids and standard BP with a 60-24-2 architecture, learnt the whole training set and obtained a 89.2% generalization for the "aspect angle dependent" set. Their stop criterion was 300 epochs. We constructed the same architecture with sine activation functions, and it was trained by 400 epochs over the entire training set. The range of weights was 3 and 0.1 for the first and second layer respectively. The learning rate for the first layer was 0.1, and for the second 0.0001. The whole training set was learnt and a 92.9% generalization was obtained.

4 Other problems

In a learning task with strings generated by finite state automata (prediction of the next symbol) [Sopena and Alquézar, 1994] found learning to be ten times faster using sines than using sigmoids.

In a classification task with examples generated with a biconditional grammar, [Moliner and Tubau, 1997] found that a network with sigmoid activation functions was unable to learn the training set. The same network with sine functions learnt the set extremely fast, and obtained a generalization higher than 98%.

Finally, in a task involving the generation of fixed length sequences, a generalization of 98.2% was obtained, while the same architecture using sigmoid functions was unable to learn the task [Moliner and Sopena, 1993].

5 Conclusions

All the benchmarks confirmed the suppositions we presented in section 2 concerning the respective performance of the sine and sigmoid functions. A multilayer perceptron with sine activation functions learns much more quickly than one using sigmoid functions. Using standard BP, learning speed can be between 150 times faster (viz. "Negation N") and 500 times faster (viz. "Two spirals"). The speed is also greater than that obtained with the modified versions of BP proposed up to date. The computational capacity of the networks with sine functions is also greater. All the functions that could not be learnt with sigmoid functions ("Two spirals","Separation problem") were learnt easily with sine functions. Finally the generalization in all cases is better, and in some cases considerably so.

The modification described here appears to be compatible with most modifications for BP suggested to date. In future studies we aim to examine possible combinations with some of these modifications. It would also be interesting to analyze formally the properties of these types of network. Their similarities with Fourier series open up a range of interesting possibilities: interpretation of weights in terms of frequencies; a simple growing network algorithm (adding a hidden unit to a network in the same way a term of a Fourier series can be added to a nth-degree trigonometrical polynomial). We are currently dealing with one of the problems that the use of sine functions presents: the difficulty to find an initial set of weights in high-dimensional inputs.

References

[Barron, 1993] Barron, A. Universal Approximation Bounds for Superposition of a Sigmoidal Function. *IEEE Transactions on Information Theory, 39*, 930-945.

[Fahlman, 1988] Fahlman, S.E. Faster Learning Variations: An empirical study. In Touretzky, D. et al. (ed.), *Proc. of the NN Model Summer School*. Morgan Kaufmann, San Mateo, Ca.

[Fahlman and Labiere, 1990] Fahlman, S.E. & Labiere, C. The Cascade Correlation Learning Architecture. In Touretzky, D. (ed.), *Advances in Neural Information Processing Systems 2*. Morgan Kaufmann, San Mateo, Ca.

[Gorman and Sejnowski, 1988] Gorman, R.P. & Sejnowski, T.J. Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets. *Neural Networks, 1*, 75-89.

[Lang and Witbrock, 1988] Lang, K.J. & Witbrock, M.J. Learning to Tell Two Spirals Apart. *Proc. 1988 Connectionist Models Summer School*, Morgan Kaufmann.

[Lapedes and Farber, 1987] Lapedes, A. & Farber, R. Nonlinear signal processing using neural networks: prediction and system modelling. Tech. Rep. LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos NM.

[Lowe, 1991] Lowe, D. What have Neural Networks to Offer Statistical Pattern Processing?. *Proc. SPIE Conf. on Adaptative Signal Processing*, 460-471, San Diego, Ca.

[Moliner and Sopena, 1993] Moliner, J.L. & Sopena, J.M. Variable binding using serial order in recurrent neural networks. In Mira, J. et al. (ed). *New Trends in Neural Computation*. LNCS-686. Springer-Verlag.

[Moliner and Tubau, 1997] Moliner, J.L. & Tubau, E. Learning complex rules. *Proc. of the ASSC*, Pomona, Ca.

[Riedmiller and Braun, 1993] Riedmiller, M. & Braun, H. A direct adaptive method for faster backpropagation learning. The RPROP algorithm. In Ruspini H. (ed.), *Proc. IEEE Int. Conf. on Neural Networks*, 586-591. San Francisco.

[Rumelhart *et al.*, 1986] Rumelhart, D. et al. Parallel Distributed Processing, Vol. 1. MIT Press.

[Sopena and Alquézar, 1994] Sopena, J.M. & Alquézar, R. Improvement of learning in recurrent networks by substituting the sigmoid activation function. *Proc. of the ICANN, vol. 1*, 417,420. Sorrento, Italy.

[Tollenare, 1990] Tollenare, T. Supersab: Fast adaptive backpropagation with good scaling properties. *Neural Networks, 2* (5).

[Yang *et al.*, 1993] Yang, L. & Wanzhen, Y. Backpropagation with Homotopy. *Neural Computation, 5*, 363-366.

[Yoshizawa *et al.*, 1993] Yoshizawa, S., Morita, M., & Amari, S. Capacity of Associative Memory Using a Nonmonotonic Neuron Model. *Neural Networks, 6*, 167-176.