A Qualitative Version of Backpropagation Learning

Bernardo Morcego Seix

Automatic Control and Computer Engineering Deptartment
Univ. Politècnica de Catalunya
c/ Pau Gargallo, 5
Barcelona, Spain
fax +343 401 70 40
bernardo@esaii.upc.es

Andreu Català Mallofré

Automatic Control and Computer Engineering Deptartment
Univ. Politècnica de Catalunya
c/ Pau Gargallo, 5
Barcelona, Spain
fax +343 401 70 40
andreu@esaii.upc.es

Núria Piera Carreté

Applied Mathematics II
Univ. Politècnica de Catalunya
c/ Pau Gargallo, 5
Barcelona, Spain
fax +343 401 70 40
nuria@ma2.upc.es

Abstract

Neural networks are suitable models for qualitative techniques to be applied. We explore how qualitative reasoning could deal with the well known backpropagation learning algorithm. Qualitative models are based on the discretization of their parameters and the use of closed operators on the sets induced by the discretization. Henceforth, a qualitative version of backpropagation is an algorithm in which the variables involved in it belong to one among the finite classes defined. It can be very useful either to realize a physical implementation of the algorithm or as a starting point to develop new reinforcement learning algorithms for neural networks.

Introduction

This work is concerned with the establishment of a relation between the fields of qualitative reasoning and neural networks. It is quite difficult to do so at first glance because the treatment data receives in most neural networks, from beginning to end, is purely numerical: weights interconnecting neurons are real numbers, activation functions are (usually) bounded real numbers, and in general all the variables involved in the process of learning and test are real numbers.

Artificial neural networks were first conceived as a very simple model of the processing carried out by natural ones. The first artificial neurons were binary and their interconnecting links had +1 or -1 values [1]. Further models included real valued weights [2] and subsequent ones inherited this level of complexity and even increased it adding non-binary activation functions, recurrent links [3], pulse activated neurons [4], etc. And it was not done in vain; the first models were only capable of mapping separable boolean functions and current ones are able, for example, to recognize people faces [5] or control a complex non-linear process [6].

In spite of this, there have been three remarkable neural network approaches using qualitative reasoning techniques. The first one is concerning the use of qualitative data as input or output of neural networks, the second one is the developement of reinforcement learning algorithms [7], and the third one is the use discretized weights [8].

Neural networks have proven successful on mapping qualitative input data to real valued output data, or vice versa. A well-known application of this kind is the recognition of speech sounds using as input articulatory features of letters, which is done in [9].

A more complicated use of qualitative data is reinforcement learning. This is a supervised method in which there is no teacher but a critic. The error signal is therefore a reward or a penalty rather than a real valued quantity. For instance, Barto, Sutton, and Anderson [10] had been working with a reinforcement learning unit that learned associations from the states of a dynamic system to control

actions. The dynamic system involved a pole hinged to a cart, and the learned control actions were forces upon the cart. The learning system consistently learned to maintain the pole's balance. The goal of this task was expressed solely through a reinforcement value of -1 whenever the pole fell past a designated angle from vertical or the cart hit the end of its track, with zero-valued reinforcement all other times. As we can see, in that case we are working with a qualitative description of the available information on the possible variations of angle's values.

There have been different attempts to discretize weights for digital implementations, allowing binary and ternary values {+1, -1, 0} for the weights, [11]. At the same time many efforts have been put in developing learning algorithms for discrete weights [12].

On the other hand, it seems feasible and justifiable to study the learning algorithms used to configure neural networks from a qualitative point of view. It seems feasible because some attemps have already been taken (as discussed previously) and justifiable because it might lead to faster or easier to implement schemes. Take, for instance, gradient algorithms. They intend to follow the steepest gradient descent to reach a global optimum. The main objective of an algorithm of this kind is to compute the direction (positive or negative) a weight must follow to minimize the difference between the output and the desired output. Some of these algorithms (backpropagation, for example) do not even try to find the size of the increment but they just add the weight a quantity proportional to the gradient. One can then say that, to some extent, backpropagation follows a qualitative scheme and we will try to prove it throughout this article.

After this brief introduction the outline of the article is as follows: the next section describes schematically the backpropagation learning algorithm [13] for multilayer perceptron networks. This algorithm is probably the most popular among neural networks practitioners, and it is proved to obtain very good results on many different applications. After the study of backpropagation various proposals of qualitative versions are given in the following section. Next is an experimental validation of the different backpropagation qualitative algorithms. Finally, we conclude and sketch our future lines of work.

The Backpropagation Algorithm

Backpropagation is a learning algorithm for multilayer feedforward neural networks (MFNN) and it is based on gradient descent on the hypersurface of errors. An MFNN is a neural network where neurons are arranged in layers of processing elements (neurons). Two layers are always present: the output and the input layers, and between them there is usually at least one hidden layer. The word feedforward means that activation always flows from the input to the output, i.e. there are no recurrent connections. Fig. 1 shows an example of an MFNN and Fig. 2 describes the notation used throughout the article for layers' indexes in an MFNN. Following, there is a synthetic derivation of the backpropagation algorithm.

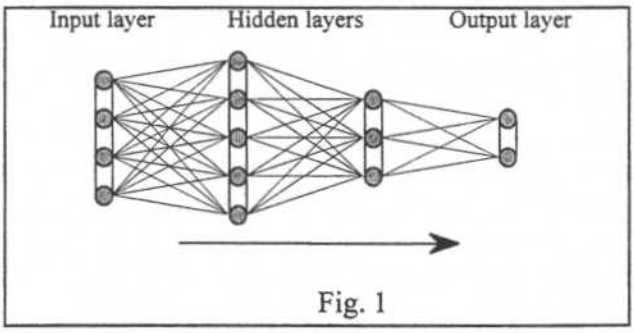


Fig. 1

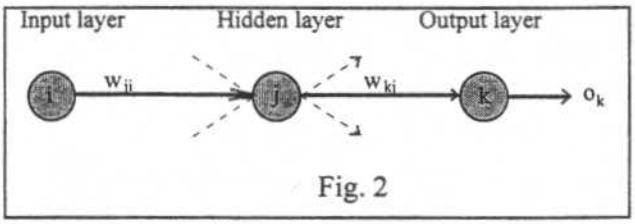


Fig. 2

The input to node j in the hidden layer is $s_j = \Sigma o_i w_{ji}$ and its output is $o_j = f(s_j)$ where f is the activation function (usually a sigmoidal function, but sometimes also linear or gaussian).

A gradient search of the minimum system error is based on the minimization of the sum of squared errors of the patterns presented to the network:

$$E = \frac{1}{2}\sum_k (d_k - o_k)^2 = \frac{1}{2}\sum_k e_k^2 \qquad (1.1)$$

Weights are updated according to:

$$\Delta w = -\eta \frac{\partial E}{\partial w} \qquad (1.2)$$

where η is a user supplied constant: the learning rate. The derivatives of weights have different expressions for weights of the hidden layer and for those of the output layer. The final update formulae is the following:

$$\frac{\partial E}{\partial w_{kj}} = e_k \cdot f_k'(s_k) \cdot o_j = \delta_k \cdot o_j \qquad (1.3)$$

for weights w_{kj}, i.e. weights connecting neurons with the output layer and

$$\frac{\partial E}{\partial w_{ji}} = f_j'(s_j) \cdot \sum_k (\delta_k w_{kj}) \cdot o_i = \delta_j \cdot o_i \qquad (1.4)$$

for weights w_{ji}, i.e. weights connecting neurons with the hidden layers.

Backpropagation, the name of the algorithm, comes from the way derivatives of the errors are calculated: they are first obtained for output neurons and propagated backward through previous layers. It can be observed how this is done in Fig. 2.

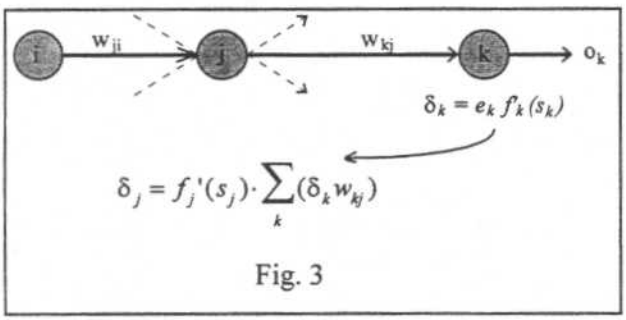


Fig. 3

In the context of this article, the most remarkable characteristic of backpropagation is that it can *only* obtain the direction in which a weight must be altered to decrease the errors made by the network (1.1). This means that after processing the derivatives of the errors respect to all weights, the only *true* information is their sign. The amount of weight increment results from the heuristic that when error decreases are steep, the step size must be big, and vice versa.

This heuristic is the main drawback of the algorithm and many authors tried to reduce its effect adding second order derivative terms. These approaches usually lead to much faster and accurate learning at a higher computational cost. The question of when will second order learning algorithms globally outperform first order ones still remains open.

The Qualitative Version of Backpropagation

Before beginning with the derivation of the qualitative version of backpropagation two questions need to be answered: is it feasible to derive a qualitative version of backpropagation? and, what do we understand by a qualitative version of backpropagation?

The answer to the first question was partially explained in the previous sections and is clearly yes. The principal reason to say so is because the only *true* information available after a training cycle is the sign of Δw. Therefore, if by using the algebra of signs we are able to obtain the sign of the error derivative (without obtaining a [?] result) and we modify the constant η to counterbalance the effect of working with products of ± 1 we will have obtained what we looked for.

A remark must be done before answering the second question. We do not want to alter either the structure or the operation of the network, so we can only treat the variables involved in the learning process, which are e, f' and w. Therefore, different algorithms can be considered as qualitative versions of back-propagation depending on which of these variables are transformed into qualitative, i.e. classified into SD = {[+], [0], [-]}.

Now we can proceed with the derivation of the qualitative version of backpropagation. The main objective of such a derivation will be to obtain δ_{oj} and δ_{oj} in SD by using the algebra of signs.

In expression (1.3) one can easily notice that if the values e and $f' \in$ SD the resulting δ_{oj} will always be determinate, i.e. will also belong to SD. This is because the product is a closed operation in SD.

Equation (1.4) needs to be analysed more carefully. On the one hand, addition is not a closed operation in SD (for example, [+] + [-] = [?]) and on the other hand, some attention must be paid if we want to

convert w to a qualitative value because this parameter is intrinsically real valued and making such a conversion may be meaningless.

In order to solve the first problem and not to alter the behaviour of backpropagation we used the 'majority' operator, which is a modification of the qualitative addition. This operator carries out a sort of voting between the addends and the result is the winning sign, or [0] if there is no winner.

					maj1	maj2
1	0	-1	-1	1	0	[0]
1	1	1	-1	-1	1	[+]
1	1	1	1	0	4	[+]
-2	0	-1	-1	2	-2	[-]

Table 1

There are two versions of the majority operator and each can take different forms depending on the addends. Version 1 quantifies the voting and its result is a number (positive or negative) indicating the difference between [+] and [-]. Version 2 normalizes this result giving only a sign. It is also noticeable that when the addends are signs (i.e. both δ and w are qualitative) the voting is equitable, while when either δ or w is real valued the voting is not really fair because voters have different number of votes. Table 1 shows examples of the use of both operators.

The conversion of real valued weights to qualitative terms (in order to have a fair voting) is controversial. On one side, having $w \in$ SD would normalize the contribution of all weights and δ would be the most influential learning factor. On the other side, the error being backpropagated should somehow be proportional to the degree with which a certain neuron contributed to generate it: the weight.

As one can easily see, there is not only *one* qualitative version of backpropagation but quite a few. There are two operators and three variables to convert to qualitative, which results in 14 versions of the algorithm. A brief description of the meaning of transforming each variable into qualitative follows:

- e: if the error is a sign (belongs to SD) the network only knows if it gets closer or further to the minimum rather than how closer or further. Approaches similar to this one are the basis for most reinforcement learning algorithms.

- f': this variable is used to obtain the direction pointing to a minimum. It is, therefore, very reasonable to choose only the sign of the activation function derivative. On the other hand, transforming it to qualitative could allow the use of different activation functions with hard to calculate derivatives.

- w: it was explained previously.

Experimental Results

In order to validate and compare the different versions of the obtained learning algorithm, we set a test bed consisting of 4 problems, each one representing a general class of problem. They are:

XOR: $\{ 0,1 \}^2 \longrightarrow \{ -1,1 \}$.

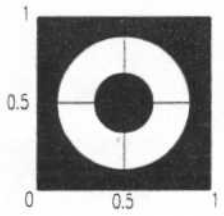
This is a combinatorial function with two binary inputs and one output. It is the most frequently used non-linear separable function to test any learning algorithm, and its truth table is shown.

in1	in2	xor
0	0	-1
0	1	1
1	0	1
1	1	-1

DECODE: $\{ 0,1 \}^n \longrightarrow \{ -1,1 \}$. This is a combinatorial function with n binary inputs and n binary outputs. Each input vector contains all zeros and a single 1, and the network must be able to codify in the hidden layer (of $\log_2 n$ neurons) the n different inputs.

DONUTS: $\Re^2 \longrightarrow \{ 0,1 \}$. This function classifies (x,y) real pairs into two groups: the ones inside a donut and the ones outside it. When this task is simplified by reducing the internal circle radius to zero, it can be

solved using a small network of 2 inputs, 2 hidden neurons and 1 output. To solve the DONUTS task one can compose two of these networks, obtaining a 4 layer network.



SQRT: $\Re^3 \longrightarrow \Re$. The last function is an analytical function: it is the square root of the division between two different linear combinations of the inputs. This is the saturation factor of an image's pixel calculated with the red, green and blue levels of the pixel. Generally speaking, analytical functions that map real numbers to real numbers can be solved, but the accuracy depends on the number of hidden neurons. Sometimes, though, it is possible to find non-standard configurations that work accurate enough, like the one in Fig 4.

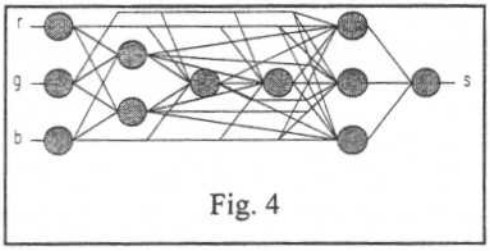


Fig. 4

A remark must be done before proceeding with the results of the tests: the implementation of the qualitative version of backpropagation induces new parameters to be incorporated to the algorithm. These parameters determine the intervals of discretization of each variable (e, f, and w). Parameter setting allowed all possible tests for XOR and SQRT but some tests could not be done for DECODE and DONUTS problems. The complete statistic analysis was, therefore, only performed for those problems which allowed it.

Each test consisted of 45 runs of a network learning the task at hand. Tests were realized with backpropagation and the 14 different qualitative versions. In each run the network had to attain a previously fixed degree of precission for the function to be learned.

In figures 3a and 3b there is a graph of the means obtained for problems XOR and SQRT respectively, and each column plots:

col 1:	normal backpropagation
col 2:	majority v1, qualitative e
col 3:	majority v1, qualitative f
col 4:	majority v1, qualitative w
col 5:	majority v1, qualitative e and f
col 6:	majority v1, qualitative e and w
col 7:	majority v1, qualitative f and w
col 8:	majority v1, qualitative e, f and w
col 9:	majority v2, qualitative e
col 10:	majority v2, qualitative f
col 11:	majority v2, qualitative w
col 12:	majority v2, qualitative e and f
col 13:	majority v2, qualitative e and w
col 14:	majority v2, qualitative f and w
col 15:	majority v2, qualitative e, f and w

Figures 3a and 3b show there is an evident difference between the learning time (computed in learning cycles) of the tested algorithms.

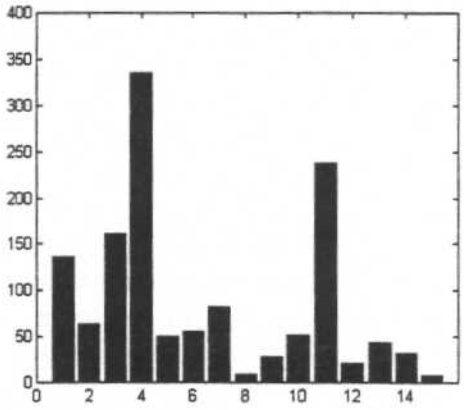


Fig. 3a

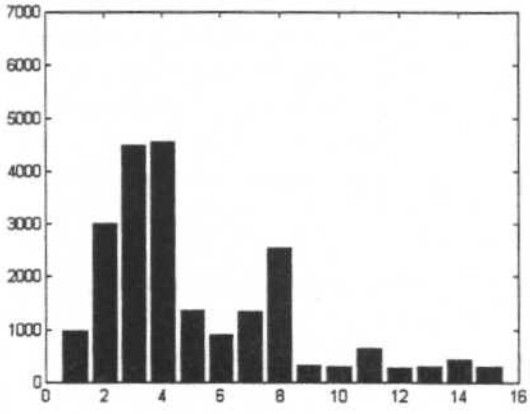


Fig. 3b

	XOR	DECODE	DONUTS	SQRT
Backpropagation	89.2 ± 51.6	106.68 ± 7.88	783.5 ± 260.8	984.2 ± 235.5
v7: Maj. v1 + *e*, *f'*, *w*	9.22 ± 1.26	90.4 ± 25.6	747 ± 326	2534 ± 682
v8: Maj. v2 + *e*	28.2 ± 2.26	149.89 ± 21.79	1688 ± 528	316 ± 86.3
v14: Maj. v2 + *e*, *f'*, *w*	7.76 ± 1.06	49.1 ± 41.2	605.9 ± 220.7	300 ± 120

Table 2

After concluding the tests, only versions 7, 8, and 14 of the qualitative algorithm were comparable with backpropagation on each of the four problems. Algorithms 8 and 14 use the majority operator version 2 while algorithm 7 uses version 1 of the operator. The qualitative variable of algorithm 8 is *e* and they are *e*, *f'* and *w* for algorithms 7 and 14. Table 2 shows 99% confidence intervals for the mean of the learning time (computed in learning cycles) and those shaded indicate the mean can be considered statistically smaller than that of backpropagation with more than 1% of probability.

Two facts can be observed when examining the previous table: first is that version 14 of the qualitative algorithm performed better than backpropagation in 3 of the four problems and second, problem DONUTS was very hard for all qualitative versions.

It should be noticed that version 14 of the qualitative algorithm is the "most" qualitative of all and its performance shows that it cannot be stated that backpropagation makes better use of the information provided by the error than it does.

On the other hand, DONUTS is a difficult task because it needs accuracy, in the sense that the boundaries of the donut can be assigned either class if the network is not accurate enough. We observed that qualitative versions of backpropagation got very close to a solution but failed to attain it, i.e. the learning rate needed to be adaptive.

Conclusions

This work is a first attempt to deal with the intrinsic qualitative aspects in the learning processes employed by neural networks models. The results obtained using qualitative operators and qualitative parameters modifying the backpropagation algorithm have shown better or similar performance in the chosen examples. We selected these four tasks because they cover a broad range of simple real problems. They are strongly non-linear, their input and output sets cover from { 0,1 } to $\Re^n$, and DECODE problem presents compaction capabilities.

The measure of performance is training time computed in learning cycles and not only did some versions of the qualitative backpropagation outperform the classical one but they also present the additional advantage of easier implementation. This is due to the use of qualitative variables and operators that make real number products become sign products and real number additions become integer comparations. This leads to a shorter and easier to calculate algorithm.

This work is just an initial test for the use of qualitative techniques applied to neural networks. Different lines seem interesting to follow: for instance, a more sophisticated qualitative description of parameters could be done and might lead to more efficient algorithms, although such finer partitions of the real line raise new problems to account for. Another possibility is to extend the methodology applied here to other learning algorithms, such as counterpropagation, Hebb rule, etc. Maybe the most urgent task to take care of is to improve the algorithm in order to automatize the setting of parameters, specially those parameters that are not present in backpropagation and appear because of classifying variables in SD.

Bibliography

[1] McCulloch, W.S. and W.H. Pitts. A logical calculus for the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics 5:115-133,* 1943.

[2] Rossenblatt, F. The Perceptron, a probabilistic model for information storage and organization in the brain. *Psycological Review 62*, 1958.

[3] Hopfield, J.J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Science (USA) 81:3088-3092,* 1984.

[4] Dress, W.B. Frequency-coded artificial neural networks: an approach to self-organizing systems. *Proceedings of the First Annual International Conference on Neural Networks, vol 2,* 1987.

[5] Fuchs A., Haken H. Pattern Recognition and Associative Memory as Dynamical Processes in Non-Linear Systems. *Proceedings of the IEEE International Conference on neural Networks, San Diego, pp 217-224.* 1988

[6] Steven C. Suddarth. A Symbolic-Neural Method for Solving Control Problems. *Proceedings of the IEEE International Conference on neural Networks, San Diego,* 1988

[7] Barto A.G., Sutton R.S.,and Brower. Associative Search Network; A Reinforcement Learning Associative Memory. *Biological Cybernetics, 40, pp.201-211,* 1981.

[8] Carrabina, Jordi. Xarxes Neuronals VLSI d'alta velocitat. *Doctoral disertation. Universitat Autònoma de Barcelona,*1991.

[9] Jeffrey L. Elman. Finding structure in time. CRL *Technical Report 8801*, 1988.

[10] Barto A.G., Sutton R.S., Anderson C.W. Neuronlike elements that can solve difficult learning control problems. *IEEE Trans. on Systems, Man, and Cybernetics, 13, 835-846,* 1983.

[11] Sivilotti M.A., Emerling M.R., Mead C.A. VLSI Architectures for Implementation of Neural Networks. *American Institute of Physics conference Proceedings 151, Neural Networks for Computing, pp408-413,* 1986.

[12] Perez Vicente C., Learning algorithm for binary synapsis. *Statistical Mechanics of Neural Networks, Springer Verlag* 1990.

[13] Rumelhart D.E., Hinton G.E., Williams R.J. Learning internal representations by error propagation. *Parallel Distributed Processing: Exploration in the Microstructures of Cognition, vol.1, pp.318-362, Mit Press,* 1986.