Comparison of Counter-propagation and Spatio-temporal Nets in the Detection of Sequences of Numbers

Juan Moreno[1], Gabriel Sebastian[1], Miguel A. Fernandez[2] and A. Fernandez Caballero[2]

Departamento de Informatica, Escuela Politecnica Superior de Albacete
Universidad de Castilla-La Mancha
Albacete, 02071, Spain
[1]{jmoreno,gsebas}@sancho.info-ab.uclm.es [2]{miki,caballer}@info-ab.uclm.es

Abstract. Due to the growing interest of many applications in detecting sequences of numbers (communications, artificial vision,...) where some of them may appear faulty, we thought about the possibility to compare two types of adequate neural nets to resolve this problem. These are the counter-propagation and the spatio-temporal nets. To see the effectiveness of each of them, our team has implemented two simulators of a system whose function is to correct a telephone number (as example of a sequence of numbers). To do this, the system reads the telephone number and, using the counter-propagation and the spatio-temporal nets, selects a telephone number of among all the learned ones.

1 Introduction

This paper seeks to check the efficiency of the counter-propagation nets (CPNs) and of the spatio-temporal nets (STNs) in the detection of sequences of numbers containing errors. To check each of the described nets, two simulators of a system have been implemented. These simulators memorise a calendar of telephone numbers, and when they receive as input a telephone number, either a complete one, or where some digits are missing (this is simulated introducing an '* ' called generic digit), or where some numbers appear to be exchanged, the system corrects the telephone number (if it is erroneous) with another one selected among all the previously learned ones.

Hecht-Nielsen [5] synthesised the CPN by combining a structure known as competitive net with Grossberg's outstar structure [2][3][4], obtaining this way the so called counter-propagation net. This net's functionality is shown in figure 1. Given a group of vectors (x_1,y_1), (x_2,y_2), ...,(x_L,y_L), the net learns how to associate one vector X in the input layer with one vector Y in the output layer. If the relationship between X and Y may be defined by means of a continuous function Ω, such that $Y=\Omega(X)$, then the net will learn how to approach that correspondence for any X value in the interval specified by the training vectors set.

As you may appreciate on figure 1, the architecture consists of three layers. An input vector is offered to all layer 1 units. Each unit of layer 2 calculates its net input value, and a competition takes place to calculate which unit possesses a greater net input. This unit will be the only one to send a value to the output units.

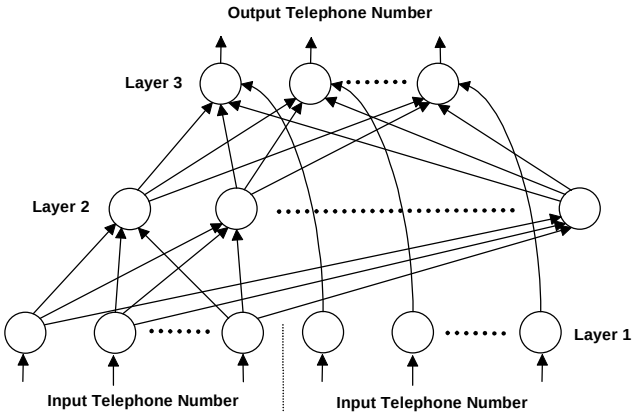

Figure 1. The CPN structure

To test the STN we have designed a three layer architecture. This architecture incorporates an input layer whose output is a normalised vector calculated according to a mathematical expression that we will see later on. Moreover, the architecture consists of a hidden layer that receives the output of the input layer. This layer is precisely a spatio-temporal net. The architecture has also an output layer whose function is to choose the output of among all the learned ones. Spatio-temporal nets incorporate the peculiarity that the presented input vectors have a correlation in time. This architecture is based in the structure of Grossberg's formal avalanche [6]; instead of carrying out a recognition operation, the avalanche allows to learn and to remember a sequence of spatio-temporal patterns. Let $\{Q_{11}, Q_{12}, ..., Q_{1n}\}$ be the sequence of numbers that feeds our net. Then, the recognition avalanche is the one represented in figure 2.

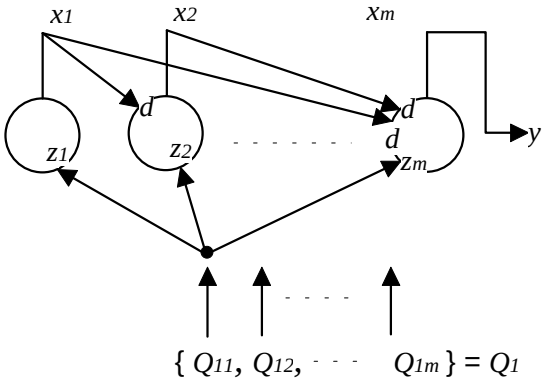

Figure 2. The STN structure

Each one of the Q_{1i} becomes a normalised vector (by means of a function carried out by the input layer), and is applied to the inputs of all STN units; they are allowed to remain there during a time t; later, the following vector Q_{1i+1} is applied, and so forth. During that time t, each unit dynamically adjusts its activation and its output values. After the processing

of all the digits for the input and STN layers, the outstar chooses a sequence of numbers of among the learned ones.

2 The CPN functionality

As it has already been said in the introduction, the architecture consists of three layers. Each of the layer 1 units normalises the received input number, passing it to layer 2. Layer 2 is known as a competitive net [5]. It is formed by a series of process elements called instars [1][2]. The mission of an instar is to memorise a normalised input vector, providing a greater output intensity the more it resembles the learned vector. The competitive net consists of a set of instars that classify any input vector. The one instar offering the greatest output is the winner of the competition, and it will be the only one that will have a non null output value. The winner will send value 1 to the outstar, and the rest of instars of the competitive net will send a 0 value. Finally, layer 3 consists of some process elements called outstars. Each of these process elements memorises a number of the sequence. The outstar's function is to provide the sequence of output numbers.

The CPN learning phase is carried out by means of a series of equations [8], although, to simulation effects, learning may be performed in the process elements by assigning the sequences of normalised numbers to each instar's weight vectors. With regard to the outstar, and in our simulations, learning is carried out in a similar way: the numbers to be learned are assigned to the weight vectors.

The CPN needs the intensity of each one of the numbers of the sequence to be defined. In our tests the selected input intensities for each of the digits are shown in table 1.

Table 1. Input intensities for each digit

Digit	Intensity
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10
*****	0 or 5 or 10

As you may observe on table 1, to each digit i we assign an input intensity i+1, since the normalised input intensity doesn't influence too much in the obtained results. For the generic digit, the chosen input intensities for the tests were 0, 5 and 10. An intensity value of 5 has been chosen because it is the one that best approaches the intensity of the rest of the digits. We have also tested with intensity values 0 and 10.

3 The STN functionality

STN works in the way described in the introduction. To perform the tests we have designed a three layer architecture, like the one shown in figure 3. Next to the STN we have added two layers (an input and an output layer). The input layer [9] has a unique process element that transforms the input number into a normalised vector. The hidden layer, a spatio-temporal net, memorises a sequence of normalised vectors. And, the output layer associates the winning sub-net with a telephone number of among the learned ones. The sequence of input numbers is sequentially offered to the unique input layer unit. Then, this layer spreads its output simultaneously to all the units of the hidden layer. Once the whole sequence of digits has been processed in the input and hidden layers, the output layer associates the hidden layer winning group with a telephone number shown as output.

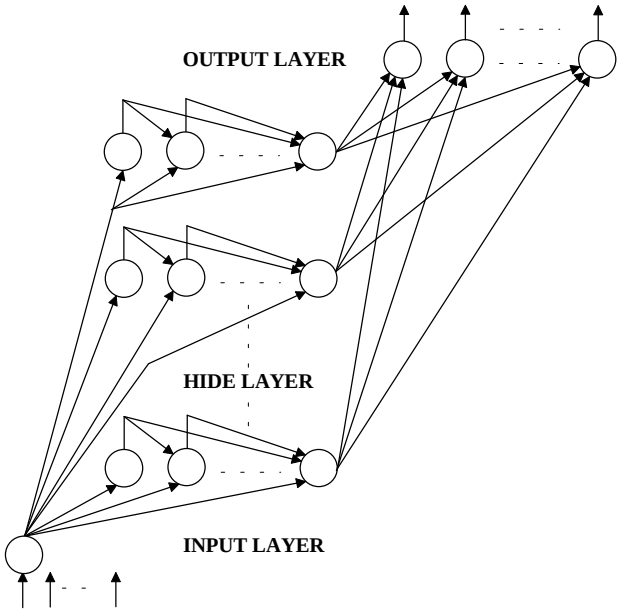

Figure 3. Our STN architecture

Our input layer has the mission of transforming an input number into a normalised vector. Since the STNs [1][5][7] require their input to be normalised, our architecture contains a layer for it. What the input layer really does is to associate to an input number a vector of those presented in figure 4. These vectors have the property of being unitary (they have module one). The output of the input layer is obtained by using the following mathematical expression:

$$\begin{aligned} &\text{if } j = \text{`*'} \\ &\quad \text{then } N(j) = (1, 0, 0) \\ &\quad \text{else } N(j) = (\cos \alpha, \sin (j * \pi/5), \cos (j * \pi/5) * \sin \alpha) \end{aligned} \qquad (1)$$

where j is the input digit to the layer, and α is the angle between the vector and the X-axis. By enlarging the value of the angle, the STN confuses less the vectors received through the input layer. This is due to the fact that a greater value of α means that the vectors are far away from each other. So the scalar product of the input vector by the STN unit learned vector will give a smaller value when erroneous. The angle $\pi/5$ is obtained by dividing the length of a circumference (2π) by 10 (the number of possible input digits, 0 to 9 in our simulations). Expression (1) may be generalised to other applications where the input number varies from 0 to n.

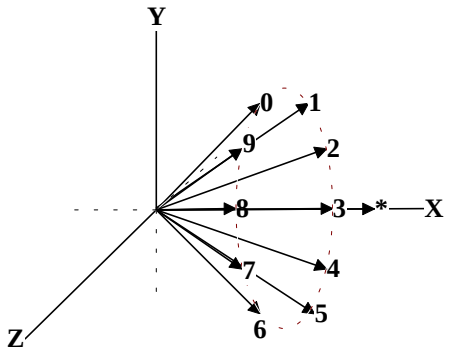

Figure 4. The vector's graphical representation

The hidden layer corresponds to a spatio-temporal net, made up of n*m units, where n is the number of sequences of numbers to be learned, and m is the number of elements a sequence consists of. The net's units receive an input value equal to the scalar product of the input vector Q by the weight vector Z of that unit. Also, each one receives an input signal coming from the outputs of all the units that precede them. Precede means at the left of a given unit, as represented in figure 2. The hidden layer is divided in groups of m units, each one taking charge of learning and recognising one sequence of numbers. The output of the last unit of each group will be the group's output value. The output unit containing the greatest answer will be the only one that has a non null value (exactly the same as for the CPN).

Finally, the output layer performs the same function than the CPN, that is to say, it selects the output sequence.

Learning in STNs is carried out by means of a series of equations [6][9], although to simulation effects, each unit of the STN is able to learn by assigning the normalised vectors to each weight vector of each instar. With regard to the outstar, learning is carried out in the same way than in the STN outstar.

4 Results

To verify the results, two simulators have been implemented, one for each kind of net. Telephone numbers have been chosen as sequences of numbers. Both simulators carry out learning of the diverse layers firstly. They next send to both net's input layers, one by one, all the telephone numbers. A telephone number consists of 9 components, since in Spain they are composed of 9 digits. The tests have been carried out randomly with a calendar of 100 elected telephone numbers, since we consider that this is a significant sample to check the validity of both types of nets. With the purpose of carrying out a statistic of results we have introduced all the telephone numbers with some error. The studied cases are the following ones: the correct telephone number (C), the phone number with a generic digit in a random position (1 *), with two generic digits in random positions (2 *), with three generic digits in random positions (3 *), with an error in a random position (E1), with two errors in random positions (E2), with three errors in random positions (E3), with two digits randomly exchanged (I2), the phone number displaced one position to the right from the initial position (D1), and the phone number displaced one position to the right from a random position (DX). By generic digit (*) we understand the digit that is to be used in case the user doesn't remember a particular digit of the telephone number. Four simulations have been performed for each studied case. Later, their mean values have been calculated and shown as results in tables 2 and 3.

Table 2 presents the results obtained for the CPN working with the different input intensities for the generic digit i=0, i=5 and i=10. This table shows the percentage of guesses for this net. The first quality to highlight for the CPN is that it learns quickly and offers an excellent response time. Observing table 2, one can affirm that better results are obtained for input intensity of 5, although the percentages are practically the same except for cases 1 *, 2 * and 3 *. In cases with erroneous digits (E1, E2 and E3), observe that the solution is very good for any i. In a generic digit analysis, excellent results are obtained for input intensity 5 (91.25%). Nevertheless, results are quite bad for intensities 0 and 10. This is because with i=5, the error that takes place between the intensity of the real digit of that position and the input value is never greater than 5. For the displacements in the input sequence, the CPN offers poor results. The results for D1 are quite worse than those for DX because all the numbers are incorrect, while in case of DX at least a part fits with the correct sequence. Finally for the exchanges the answer is acceptable, overcoming a 50% of guesses.

Table 3 presents the percentage of guesses for the STN for each input α. The greater the angle α the better the obtained results (see table 3). This is because the system confuses less the digits to each other, as its corresponding vectors are more distant. The designed architecture works better in the case of introducing one or several generic digits than in the case of introducing erroneous digits. This is due to the fact that function (1) assigns a vector to the generic digit equidistant to the rest of vectors. We highlight the results obtained in case 3 *, with guesses over a 40% for all α and near to 50% when $\alpha = \pi/3$, even more when we have other possible correct outputs when generic digits exist. For the case of error detection, the STN obtains an exceptional solution, since a percentage of 85.5% and 60.25% are given for cases E1 and E2, with $\alpha = 2\pi/5$. The results for input vector displacements are excellent too, being better in case D1 than in DX, because in the STN the inputs of the units coming from the previous units provide more weight to their output. Finally the case of exchange is better than in the CPN. We obtain a very good result of approximately a 70% of guesses.

Table 2. Results for the CPN

Studied cases	i=0	i=5	i=10
Correct telephone number	100%	100%	100%
One error	75.75%	76.75%	76%
Two errors	48.5%	48%	52.75%
Three errors	34%	31.75%	33.75%
One generic digit	40.75%	91.25%	48.5%
Two generic digits	23.75%	66.25%	27.25%
Three generic digits	16.25%	45.5%	16.75%
Displacement from the first position	4%	4%	4%
Displacement from a random position	35.75%	36.5%	38.75%
Exchange of two digits	54%	56.75%	55.5%

Table 3. Results for the STN

Studied cases	$\alpha=\pi/4$	$\alpha=\pi/3$	$\alpha=2\pi/5$	$\alpha=\pi/2$
Correct phone number	100%	100%	100%	100%
One error	83.25%	82%	85.5%	81.5%
Two errors	59.25%	59%	60.25%	59%
Three errors	34.25%	33.5%	35%	35.75%
One generic digit	81%	87%	88.75%	85.5%
Two generic digits	64.75%	64.75%	67.75%	68%
Three generic digits	41%	48.25%	44%	44.75%
Displacement from the first position	83%	83%	84%	83%
Displacement from a random position	69.75%	73.75%	73.5%	74.25%
Exchange of two digits	63.25%	72.25%	67.25%	70.5%

Finally, we want to highlight that the STN has a superior percentage of success than the CPN in most of the studied cases. In the case of the presence of a generic digit in the sequence with input intensity i=5, a better percentage is obtained for the CPN.

5 Conclusions

The CPN has a good behaviour to detect sequences of numbers with errors. For the treatment of the generic digits it provides a good behaviour when the input intensity is appropriate for the concrete case. And it is not appropriate for displaced sequences of numbers.

The STN is more appropriate than the CPN to detect sequences of numbers with errors. As it receives the outputs of the preceding units, the STN knows the information of the incoming sequences, being activated when they are correct, even if the input intensity is not the expected one. That's to say, sequencing is a cue in STN. They provide acceptable values for all the tested cases. They have an order of inferior complexity to any other system analysing the same cases [9]. Also, any other case to be analysed doesn't suppose an increment of the order of complexity. In order to deal with problems without temporary component, the input layer is primordial, because it has to find a normalised vector that has to identify the received number with no possible confusion.

As a main conclusion note that both nets are good in dealing with sequences of numbers, since the obtained results confirm it so.

6 References

1. R. Hecht-Nielsen. Neurocomputing, Addison-Wesley, Reading MA. (1990).
2. S. Grossberg. Studies of Mind and Brain, *Boston Studies in the Philosophy of Science, vol. 70*, D. Reidel Publishing Company, Boston. (1982).
3. R. Hecht-Nielsen. Counterpropagation Networks, *Applied Optics, vol. 26, no. 23***,** pp. 4979-4984. (1987).
4. R. Hecht-Nielsen. Counterpropagation Networks, In *Proceedings of the IEEE First International Conference on Neural Networks*, Piscataway, NJ, pp. II-19 - II-32, IEEE. (1987).
5. J.A. Freeman and D.M. Skapura. Neural Networks: Algorithms, Applications, and Programming Techniques, Addison-Wesley, Reading, MA. (1991).
6. S. Grossberg. Learning by Neural Networks. Published by Stephen Grossberg in *Studies of Mind and Brain*. D. Reidel Publishing, Boston, MA, 65-156. (1982).
7. R. Hecht-Nielsen. Nearest matched filter classification of spatio-temporal patterns. Technical Report, *Hecht-Nielsen Neurocomputer Corporation*, San Diego, CA. (1986).

8. Juan Moreno, Gabriel Sebastian, Miguel A. Fernandez and A. Fernandez Caballero. A Telephone Number Corrector using a Counterpropagation Network. In *Proceedings of the Fifth International Conference on Neural Information Processing ICONIP´98*, Kitakyushu, Japan. (1998).
9. Juan Moreno, Gabriel Sebastian, Miguel A. Fernandez and A. Fernandez Caballero, A Neural Architecture for the Identification of Number Sequences. In *Proceedings of the Fifth Brazilian Symposium of Neural Networks SBRN´98*, Belo Horizonte, Brazil, (1998).