ANNALI DI GEOFISICA, VOL. XXXVII, N. 3, June 1994

Seismic signals detection and classification using artificial neural networks

Giovanni Romeo
Istituto Nazionale di Geofisica, Roma, Italy

Abstract
Pattern recognition belongs to a class of problems which are easily solved by humans, but difficult for computers. It is sometimes difficult to formalize a problem which a human operator can easily understand by using examples. Neural networks are useful in solving this kind of problem. A neural network may, under certain conditions, simulate a well trained human operator in recognizing different types of earthquakes or in detecting the presence of a seismic event. It is then shown how a fully connected multi layer perceptron may perform a recognition task. It is shown how a self-training auto-associative neural network may detect an earthquake occurrence analysing the change in signal characteristics.

Key words *seismology – detection – neural network – auto-associative neural network – classification*

1. Principles of classical pattern recognition

Pattern recognition (Tou and Gonzales, 1981) is a simple task for humans, but a hard task for computers. Maybe it depends on the different architecture of the two different thinking machines, brain and calculator. The most relevant difference in the architectures is that the natural brain is essentially a parallel machine, while an ordinary computer is a sequential machine. The parallel architecture intrinsic in neural networks may help in building pattern recognition engines with superior performance.

A pattern recognition machine operates in a world (the measurements space) defined by the variable describing the phenomena under study. A pattern of these variables is represented by a point in the measurement space. The recognition goal is to divide the possible input pattern in classes, and decide if an input pattern belongs to a specific class. A class can be considered like a complex volume in the measurements space defined by a decision boundary.

The simplest pattern recognition machine is illustrated in fig 1. It operates over a one-dimension world, and the recognition task is extremely simple: it divides the measurement space into two half spaces, using a threshold. The threshold level is decided by the value of the multiplying block *a* and by the value of the bias *b*. The threshold level is the decision boundary in this simple example.

Things become more interesting if we add another input to the summing node (fig. 2). Now the recognition device operates in a two-dimensional plane, and it is able to separate this into two half-planes, using a straight line. In general if we use *n*

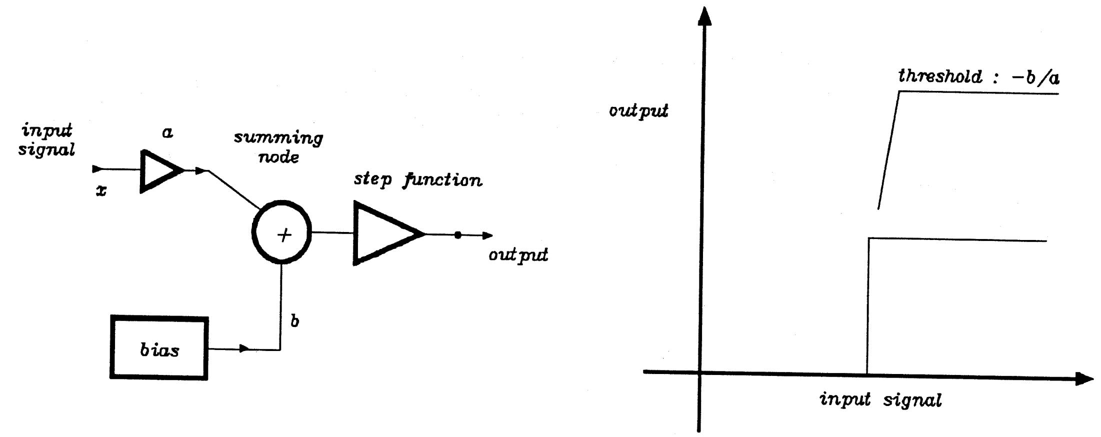

Fig. 1. The simplest pattern recognition machine, with one input and one output, can separate the mono-dimensional space of the input variable into two classes. The separation element may be tuned moving a or b.

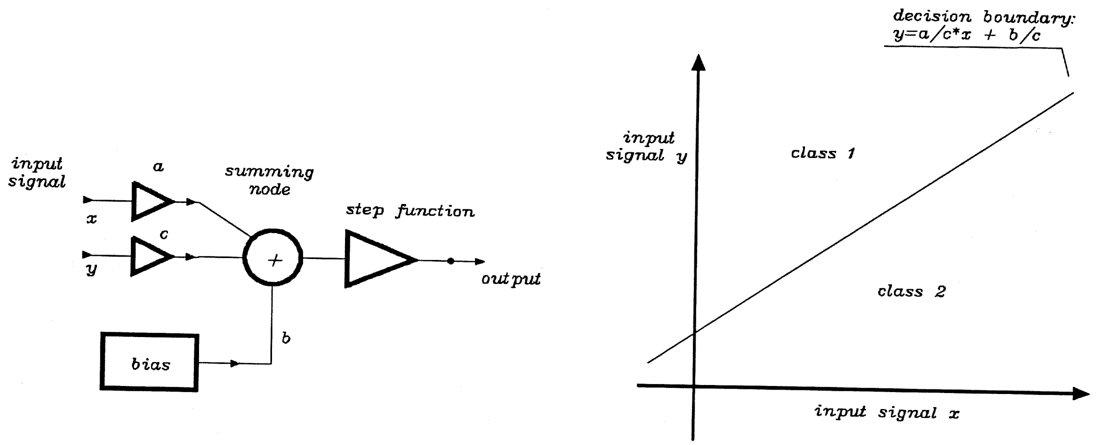

Fig. 2. A two input weighted sum can separate the input plane into two half planes. In general a weighted sum of n variables can separate the n-dimensional input hyperspace with an (n-1)-dimensional hyper plane.

inputs we obtain a device able to operate in an n-dimensional hyper space, separating it with an (n-1)-dimensional hyper plane.

Things become even more interesting if we combine several decision blocks in a tree structure (fig. 3, left). Each of the decision blocks at the input still divides the input space into two half spaces; the decision block at the output combines the output of the previous input block in order to obtain a closed shape in the measurements space. Adding one more layer, and increasing the number of decision blocks, we may obtain all kinds of complex decision boundaries in the measurements space.

2. Pattern recognition and neural nets

The mathematical neuron (fig. 4b) was formalized by McCulloch and Pitts (1943) and tries to reproduce the behavior of the physiological neuron (fig. 4a). The biological neuron activates its outputs depending on the input signals. The input signals are connected to the nerve fiber through membranes of different thickness (synapses). This allows the neuron to give a different weight to signals coming from different inputs. The mathematical neuron feeds the weighted sum of the inputs through a non linear function (activation function). Although several kinds of functions are used as an activation function, the most common

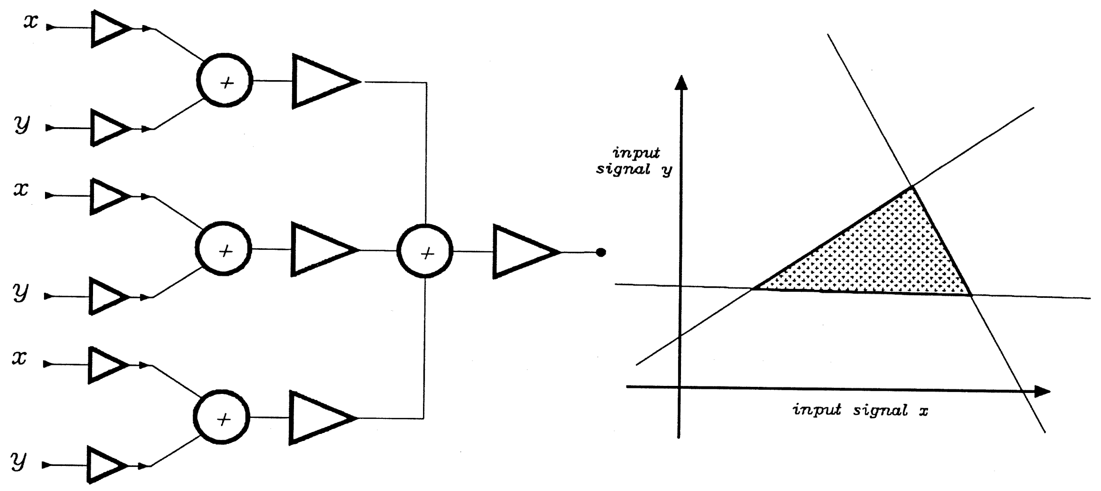

Fig. 3. By connecting several pattern recognition blocks we may obtain different shapes in the variable space.

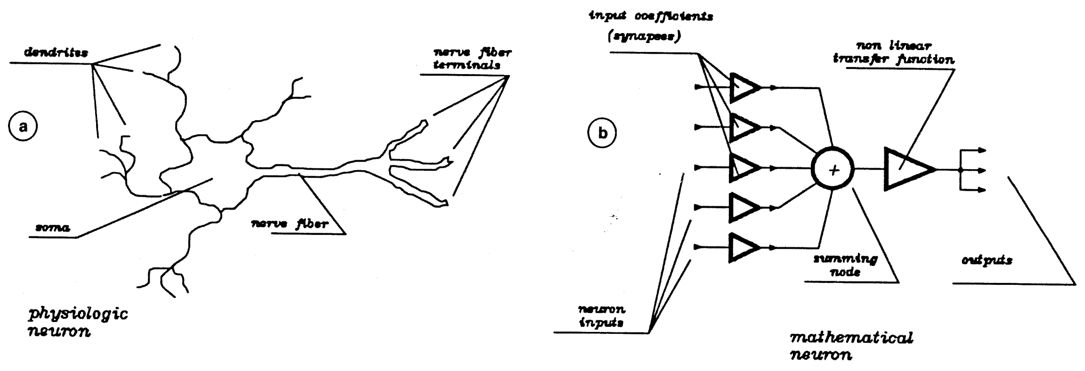

Fig. 4a,b. a) Physiological neuron; b) mathematical neuron.

are the non-decreasing functions (like the sigmoid, which recalls a smooth step function). It results the mathematical neuron looks similar to the block diagram of the classical pattern recognition machine (fig. 2) choosing a step-like activation function. The choice of a non linear activation function even makes a tree structure of neurons (fig. 3) performing a complex recognition job. A tree structure, built using linear activation functions, may be easily conducted to a single block, which can only split the measurements space into two half spaces (fig. 2).

The transfer function of the entire neuron is:

$$f(\mathbf{x}) = \lambda \left(\sum_{i=1}^{I} w_i x_i \right) \tag{2.1}$$

where the sum with weights w_i is extended to all the I elements of the input vector $\mathbf{x}$. The non-linear function λ is represented by a sigmoid:

$$\lambda(y) = \frac{1}{1+e^{-ay}} \tag{2.2}$$

Connecting several neurons (fig. 5) we obtain a neural network (sometimes called multi-layer perceptron) that can be used for pattern recognition. It looks very similar to the classic pattern recognition machine we described in section 1. A neural network, with a right number of neurons, may recognize all kinds of patterns.

The values of the connections between neurons (synapses in the physiological neuron, weights $\mathbf{w}_i$ in the mathematical one) have to be determined to make the network to perform its recognition job correctly. This task is called *training of the neural net*. This is performed using an iterative method which tries to minimize the errors made by the net in recognizing real examples. It is necessary to have a sufficient number of examples, representative of the phenomena. We define an error function as the sum of the squared values of the difference between the untrained net response and the response we would like to obtain:

$$E = \sum_{n=1}^{N} \sum_{i=1}^{I} (T_{ni} - C_{ni})^2 \tag{2.3}$$

where $\mathbf{C}_{ni}$ is the value (computed by the network) of the $\mathbf{i}$-th output port obtained for the n-th input pattern, and $\mathbf{T}_{ni}$ is the corresponding desired value.

Our goal is to minimize this function. This may be obtained using the iterative rule:

$$\mathbf{w}_k \leftarrow \mathbf{w}_{k-1} - \mu \cdot \mathrm{grad}E(\mathbf{w}_{k-1}) \tag{2.4}$$

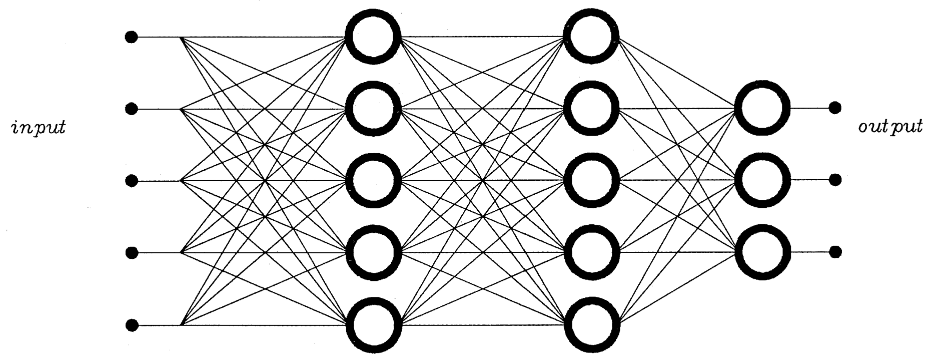

Fig. 5. A five-input, three-output multi-layer perceptron: by connecting several layers of neurons we can recognize a pattern with an arbitrary complex shape in the decision space.

(Ciaccia and Maio 1989; Lippman 1987), where k is a progressive number identifying every synapses, regardless of the layer; k-1 indicates the value before the upgrade of the synapse k and μ is a coefficient which defines the learning speed of the network. The learning speed coefficient must be carefully determined. The use of the gradient descent algorithm assumes the error surface in the $\mathbf{w}_k$ space is linear in the interval of the learning coefficient. This may be false at points of high curvature, thus making a small μ coefficient desirable, which, unfortunately slows down the learning process. Several strategies were conceived to improve the learning process (NeuralWare, 1991; Rumelhart, and McClennand, 1986) but all of them are modifications of the basic gradient descent algorithm.

3. Rules for building a neural net

The most important characteristic of a neural net is its capability to learn from examples. Therefore an important task is to collect a data set representative of the phenomena under study. This data set will be split into two parts: we will use one to train the net, the other to test the net behavior. Defining the net configuration is an empirical task. The more complex the law which ties the data to the desired response, the more complex the net is. Using a large net regardless of the law complexity may yield to unwanted results. We may compare the performances of a large and small network trained with the same data set. The large net simply memorizes the data set: it is a kind of look-up table. Its performance is very good with the data used for the training, but it gives bad results with unknown data. A small net does not memorize the data, but finds the law (if it exists) which ties the input patterns to the desired output patterns. There are no hard rules for numbers of neuron, only rules of thumb. One of them (NeuralWare, 1991), useful for networks with one hidden layer, connects the number of training patterns t, the number of the net input i, the number of net output $\boldsymbol{o}$ and the number of neurons in the hidden layer h:

$$\frac{t}{10*(i+o)} = h \qquad (3.1)$$

The first operation is to perform a training using the first set. A net may be considered trained when the difference between the output and the wanted value approaches 0 or does not change significantly. Sometimes, when using data from physical measurements, this error level can be very far from 0. This may depend on the measurement errors. If the net does not fit a few patterns, maybe those patterns are wrong. This has to be verified to decide if a new training is to be performed without them. To decide if the training is successful the net must be tested using the second data set. If the errors on the test set are comparable with the error on the training set, the net was well trained.

There are several reasons why a net may not work.

Like every hill climbing algorithm, the net training suffers from the problem of the relative minima. The error function in the weight space is very complex with several relative minima. The training can be trapped in one of them without reaching the absolute minimum. If the net does not even fit the training set, maybe we are caught in this problem. Repeating the training with a different starting point (to initialize the weight with random numbers) may be useful.

The net configuration may not be adequate to the law we want to reproduce. This case requires the design of a new net.

The data set may be inadequate to ensure the convergence (too many wrong patterns, or a bad choice of good patterns to ensure the generalization). This case requires the collection of a new data set.

4. Event classification using a perceptron

Earthquake detection and classification belong to a class of problems where artificial neural networks may be useful (Dowla *et al.*, 1990).

A simple application in earthquake classification is illustrated in fig. 6. Earthquakes recorded at the Bardonecchia (North Italy) very broad band station (Romeo, Mele and Morelli, 1991) were empirically classified into five classes: regional, local, sausage, tele, spikes. Normalized spectra were used to train the multilayer net showed in fig. 6. The output «sausage» refers to a peculiar disturbance (a burst of monochromatic waves).

The data set used for the training included 6 local events, 2 regional earthquakes, 4 teleseisms, 5 spikes, 14 sausages. After the training, the network could recognize all the patterns of the training set, and about 90% of all the other analysed and coming from the same station. The same principle was used to improve the performance of the Italian National Seismic Network. This system performs a simple trigger based on a frequency analysis (detection) which allows discarding about the 70% of incoming potential events and then a more accurate analysis is needed to pick up the true events (picking). This second step is very CPU consuming. A network shaped like the one in fig. 6 (but with 10 additional inputs containing the normalized envelope of the first ten seconds of seismograms) was trained to distinguish between true and false events coming from the de-

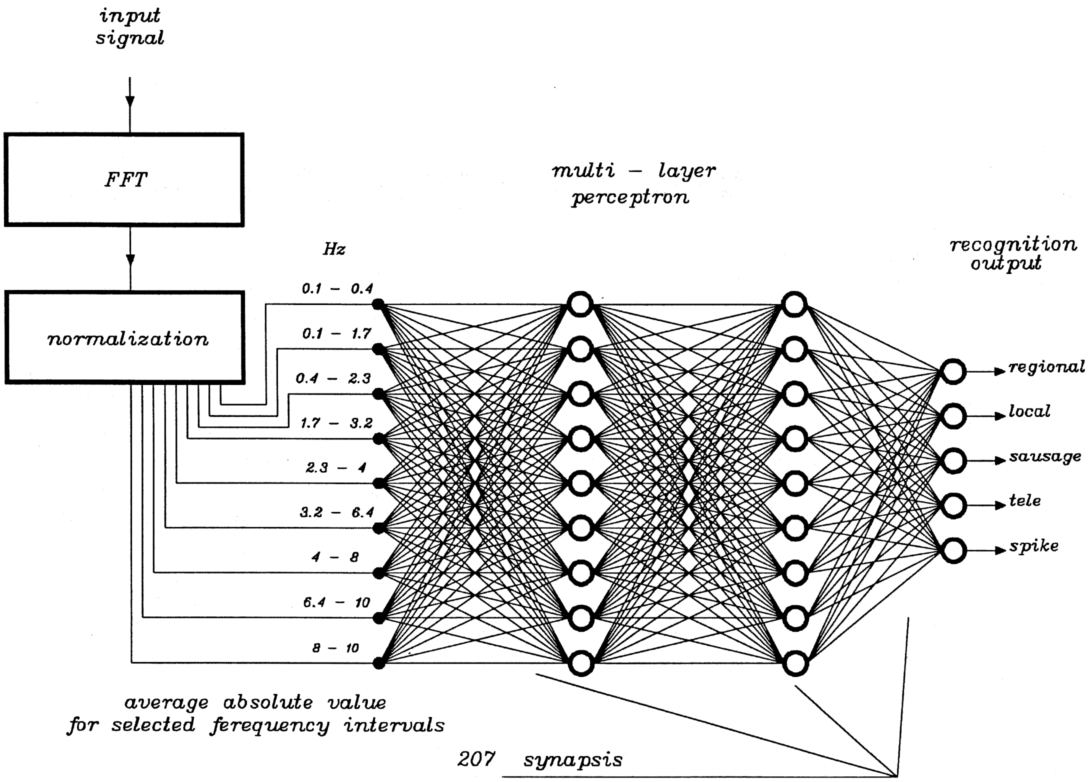

Fig 6. Neural network used in seismic event classifying (Romeo, Mele and Morelli 1991).

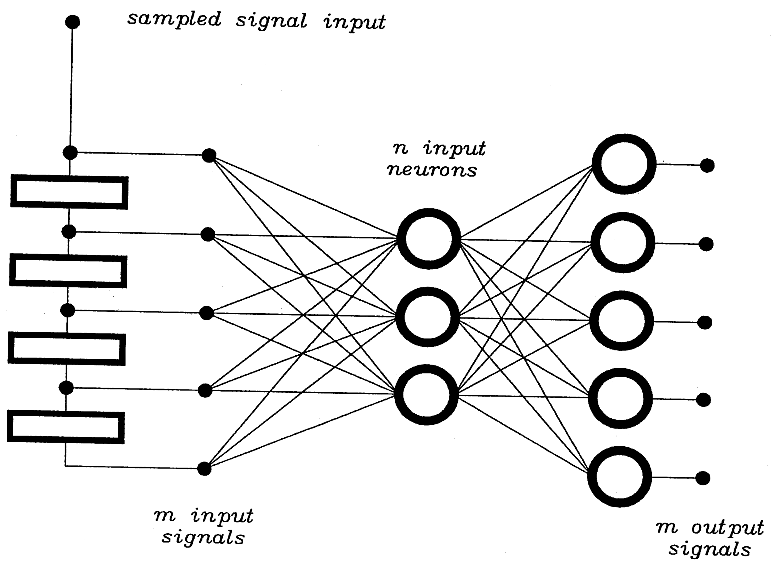

Fig. 7. Principle of an auto associative network. The discrete time series at the input is scaled through some delay blocks (rectangles at the left of the picture) and presented at the net input.

tection stage. For this purpose 2599 patterns coming from all the network stations were used. The perceptron was tested off line with the data produced by a 15 day observation period. Unfortunately the perceptron was not able to recognize all the seismic signals: 0.05% of the incoming data was discarded by the perceptron although generated by true earthquakes. This means that about 10% of all seismic signals are erroneously discarded by the perceptron. However this test was not completely unsatisfactory. The perceptron avoided useless data to be processed at the picking stage. This resulted in a CPU time reduction of about 64%.

5. Event detection using an auto associative network

The principle of an auto associative neural network is illustrated in fig. 7. The four blocks on the left represent a series of delay cells used to obtain a series of time-scaled samples at the net input. The goal of the net is to reproduce the input signal at the output (NeuralWare, 1991).

This is a very difficult task for the net. First of all because the simple operation of reproducing the input signal at the output is a linear operation, but the use of a non linear activation function (like a sigmoid) makes the net non linear. Secondly because there are only 3 neurons in the hidden layer, compared to 5 input-output. This means that the net, to perform the simple transfer job, must compress and decompress the signal. This is possible only if the net knows something about the signal. If a signal which did not appear in the training data is placed at the input, the net can not reproduce it. Figure 8a-c illustrates the response of an auto-associative net trained with a sine wave.

When we place a corrupted signal b) at the input, the net responds with an uncorrupted signal c). The possibility offered by this comportment may be used to build an adaptive neural trigger (Romeo and Taccetti, 1993) (fig. 9). An auto-associative net

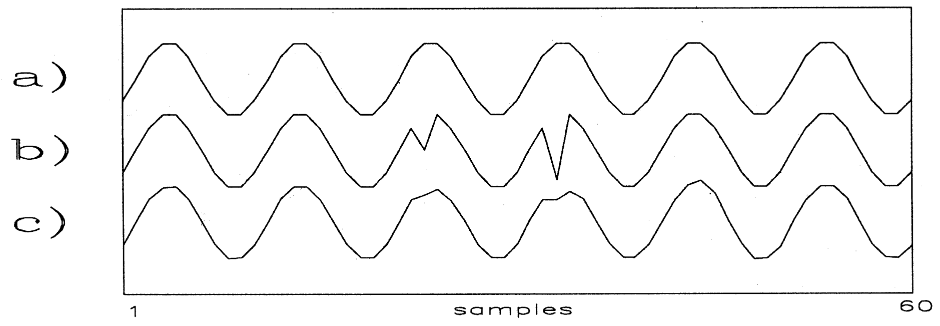

Fig. 8a-c. Waveforms for training and testing an auto-associative neural net. a) Training signal; b) corrupted signal; c) net output when excited by b) after being trained by a) (Romeo and Taccetti, 1993).

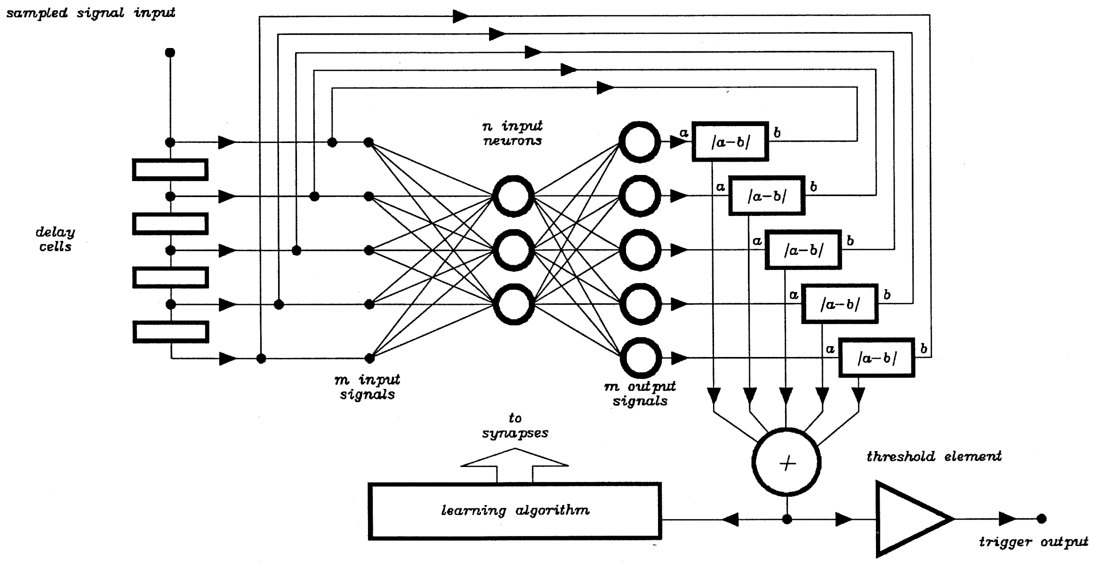

Fig. 9. Adaptive neural trigger (Romeo and Taccetti, 1993).

is real-time trained in order to minimize the error at the output. The rules used to do that are still based on the minimization of an error function:

$$E(w_k) = \sum \left| I_i - O_i \right| \qquad (5.1)$$
$$w_k \leftarrow w_k + a \cdot \frac{d}{dw_k} E(w_k) \cdot E_0$$

In the formulas above the ***k*** index indicates all the weights, regardless of their position in the net and the ***i*** index indicates the input-output couples. The value ***E***$_0$ indicates the error made for a single input pattern.

The block diagram in fig. 9 represents the real time trained auto-associative net. A training step is performed every time a new sample is presented to the input. The $E(w_k)$ error function is present at the summing of the outputs, and represents the rate of change of the signal characteristics.

The behavior of the net output during the training with a sine wave is shown in fig. 10. The learning arrow indicates the progress of the training procedure. At the beginning the untrained net output was a straight line. At the end the output curve is a sinus.

In fig. 11 we see the net error using a corrupted sine.

The corruption (a couple of wrong samples, like in fig. 8) appears roughly at sample 480. The net needs about 200 samples to learn the signal's characteristics. After sample 250 the error is almost 0. The error increases when the net encounters the corruption (near sample 480). This adaptive trigger was tested for earthquake detection.

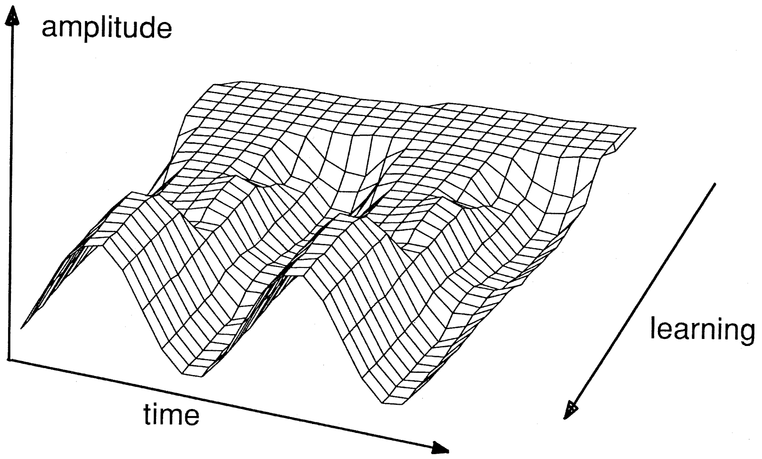

Fig. 10. Behavior of the real-time-trained auto-associative net output, excited using a sinus wave (Romeo and Taccetti, 1993).

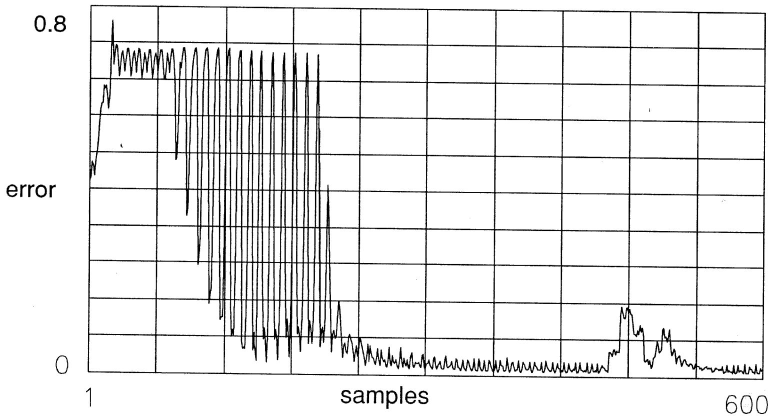

Fig. 11. Behavior of the normalized error output using a waveform containing a corruption (Romeo and Taccetti, 1993).

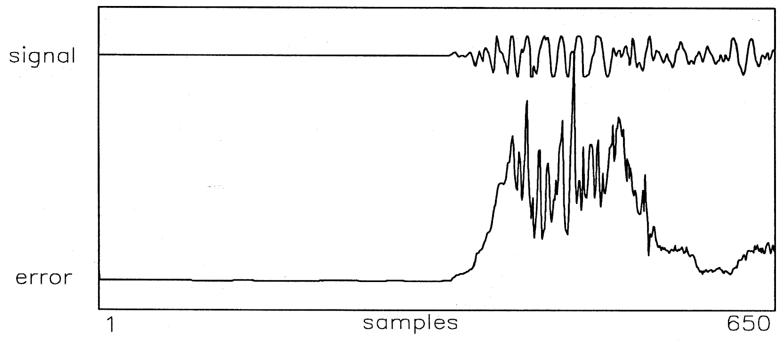

Fig. 12. Error output exciting the net with a seismic event (Romeo and Taccetti, 1993).

We used sampled data from the Italian National Seismic Network; it uses about 100 vertical short period sensors (Teledyne Geotech S13) sampled at 50 sps. Figure 12 illustrates the error signal obtained by feeding a seismic signal to the trigger.

To test the trigger we used a collection of 50 seismograms from the seismic network data base, taking care to select small events. The net shape was determined empirically and the choice of 40 input, 12 hidden, 40 output realized a good compromise between computing speed and performance.

Unfortunately the only events available for our test were those already detected by the seismic network event detector (which uses a filtered sta/lta algorithm).

In order to have seismic events more difficult to detect, we used a simple algorithm to decrease the signal-noise ratio: each sample of the signal was multiplied by a factor depending on the average of the absolute value of the signal near the sample itself. The signals produced in this way are much more difficult to detect with a classical unfiltered sta/lta method.

All the signals were fed to the neural trigger and produced a significant increase of the error signal at the beginning of the seismic event. Some of them are shown in fig. 13a-c.

Figure 13a illustrates a regional earthquake. Though SNR is low, the net does

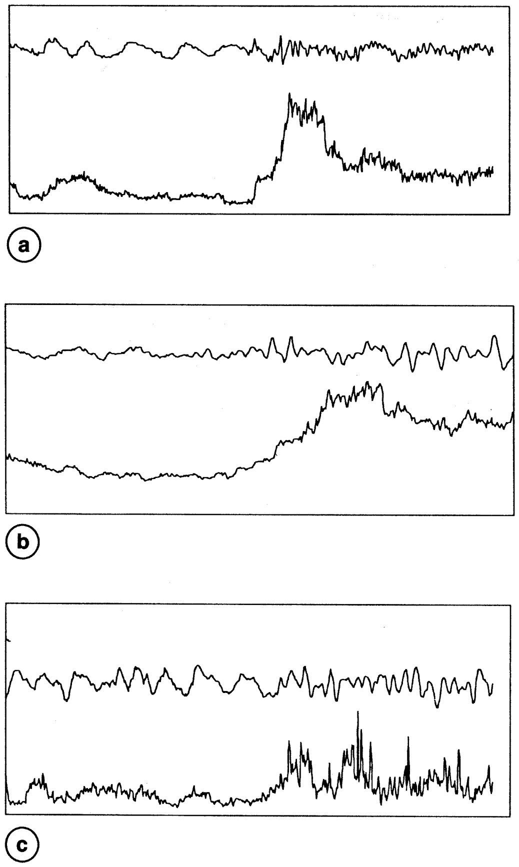

Fig. 13a-c. Three examples obtained using the net of fig. 9 with different kinds of seismic signals (Romeo and Taccetti, 1993).

not have difficulty in detecting the earthquake. Figure 13b,c illustrates teleseisms which appear hidden by the noise. The net still gives information about the presence of a seismic event.

6. Conclusions

Neural nets are powerful tools in approaching problems that can not be described using a classic algorithmic strategy. The potentiality offered by neural networks to be trained by examples is useful in building event detectors and classifiers without formalizing classes of events. The simple applications reported show this possibility. A fully connected multi layer network can be applied to event classification and proved to be rather faster in comparison with other methods. This may help saving CPU time. A neural network trained to recognize true seismic events allowed to save about 70% of computing time in an automatic event detector. The auto associative nets may be useful in building triggers. The possibility offered by the adaptive neuronal trigger may be used when the modification of the signal characteristic is significant in detecting an event of interest. A real time trained auto associative net builds an internal model of the input signal and the error made in reproducing the signal increases when some modification of the input signal prevents it from matching the internal model. This method was successfully tested using 50 seismograms from the Italian National Seismic Network. In this experiment the auto-associative trigger offers a superior performance, if compared with an unfiltered sta/lta trigger, and almost the same if compared to a well designed filtered sta/lta. The advantage is that it does not require a preliminary examination of the signal thanks to its auto associative behavior. The disadvantage is the computing time, which is higher if compared to the other types of trigger.

REFERENCES

CIACCIA P. and D. MAIO (1989): Reti Neuronali: proprietà e problematiche aperte, *Alta Frequenza*, **4**, 59-84.

DOWLA F.U., S.R. TAYLOR and R.W. ANDERSON (1990): Seismic discrimination with artificial neural networks: preliminary results with regional spectral data, *Bull. Seismol. Soc. Am.*, **80**, 5, 1346-1373.

LIPPMAN R.P. (1987): An introduction to computing with neural nets, *IEEE Transactions ASSP*, **4**, 4-22.

MCCULLOCH W.S. and W. PITTS (1943): A logical calculus of the ideas immanent in nervous activity, *Bull. Math. Biophys.*, **5**, 115-133.

NEURALWARE (1991): Neural Computing, *NeuralWare Technical Publications Group.*

ROMEO G., F. MELE and A. MORELLI (1991): Reti neuronali e riconoscimento di segnali sismici: *Atti del 10º convegno annuale del gruppo nazionale di geofisica della terra solida*, 277-283.

ROMEO G. and Q. TACCETTI (1993): Un approccio neuronale al trigger di segnali geofisici, in press on *Atti del IV Workshop Informatica e Scienze della Terra*.

RUMELHART, D.E. and J.L. MCCLENNAND (Editors) (1986): Parallel distribute processing: exploration in the microstructure of cognition, *Foundations*, vol. 1 (MIT Press).

TOU J. and R.C. GONZALES (1981): *Pattern recognition principles* (Addison-Wesley Publishing Company).