A tool for system monitoring based on artificial neural networks

MARIA GRAZIA DI BONO, GABRIELE PIERI, OVIDIO SALVETTI
Istituto di Scienza e Tecnologia dell'Informazione
Consiglio nazionale delle Ricerche
ISTI – CNR
Via G. Moruzzi 1, 56100 Pisa
ITALY

Abstract: - A research has been carried out finalized to the definition of a methodology useful for the diagnosis and prediction of the correct evolution state of physical systems. In this paper we present a related model and a specific network topology for the considered problem. In particular, the prediction procedure is based on a "Self Organizing Map"(SOM) and an "Error Back-Propagation"(EBP) networks combined to form a hierarchical architecture. The network system has been developed and tested using data furnished by Alenia and consisting in sensorial data (FBG, Fiber Bragg Grating) and multi-format descriptive data regarding evaluation (SB). The obtained results have shown that the developed methodology is a promising tool for the diagnosis activity.

Key-Words: - Artificial Neural Networks, Self-Organizing Maps, Prediction Systems, Life cycle monitoring

1 Introduction

The traditional monitoring models applied to the study of the correct evolution of physical systems are often based on prediction approaches in which, under the hypothesis of a correct functioning, on the basis of a set of measurements acquired on field, reference values are computed according to a descriptive mathematical model. These values are then compared with the real-time recalculated measurements in order to verify whether the difference between every corresponding values does not exceed a prefixed threshold; on the contrary, the system alerts the user for a potential malfunctioning. Then, a set of rules integrated in these models can derive some type of prediction for the system (*expert systems*) [6,7]. These models are very rigid and do not easily adapt themselves to the functioning variability of the systems under different and aleatory conditions. Moreover, being critical the time in the initial data pre-processing phase, some measures could not be recalculated because of a possible system time out. Other diagnosis approaches need accurate models of systems and require a fixed number of diagnoses classes (*model-based*) [6], decreasing the flexibility of the models themselves. In other cases, the expert's knowledge is stored in a library of cases (*case-based*) [6,8], even if the search for the best matching case can be computational expensive. Also used is inductive learning that includes decision trees, statistical classifiers or neural networks [6].

In this paper, we propose a novel model for the realization of a valid standalone diagnosis methodology which is also able to provide a final prediction about the state of the monitored system, or its components. On the basis of the received inputs and without requiring further elaboration, we define a structured artificial neural network architecture to describe the state of the system and to propose a final prediction. This model has been applied to study the life cycle prediction of aeronautical components demonstrating to be flexible, fast and reliable.

2 System monitoring and prediction

Monitoring a physical system, or its component *objects*, can be realised according to a specific model that follows two phases:

1. *Pre-processing phase,* dedicated to input data filtering and validation, necessary to eliminate eventual incongruence and to define a consistency control of the measures obtained from specific sensors.
2. *Classification and prediction phase*, performed on the previous data also using historical-statistical data.

We assume that the sensors belong to different categories and that we can identify correlations that

allow grouping into suitable clusters those sensors whose outputs mutually influence themselves. When this hypothesis is valid the first phase (pre-processing) can be implemented as a set of networks each dedicated to a particular group of correlated sensors. The output of these networks defines the input to the next phase.
A second network architectural paradigm has been considered to model a classification and prediction of different cases of the same problem instantiation by adopting a two level hierarchy of independent specialised networks (i.e. hierarchical neural network, HNN) for some type of prediction, the first one suitable to map the sensors sets, the second one capable to perform a final prediction on the basis of the output given by the previous net and other specific parameters [1,3].
The implementation of the global network by means of two independent network levels implies a rapid and efficient training of each individual level. The prediction phase mainly consists of two levels. The lower level (clustering level) is composed of a single network based on a SOM model, and the training is performed with the aim to cluster each input value into crisp classes, without using any information related to the object's classification class which the input value belongs to. Each specific feature is the input to the clustering level. The higher level (prediction level) is composed of a single final classifier based on an EBP model. The architecture of the model is shown in Fig. 1.

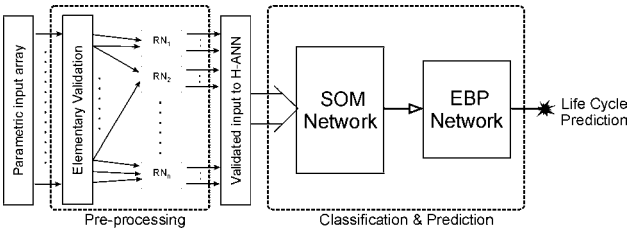

Fig. 1 Architecture of the model.

In the following, a detailed description of each step is given: pre-processing and two-level HNN description.

2.1 Input data pre-processing

The pre-processing phase regards only the input data set relative to the measurements performed on the examined object using different sensors. Since on field measurements might be incongruent owing to some bad functioning, we have introduced a data validation process, based on mathematical models, to obtain a set of validated data, also congruent.
In traditional diagnosis models, a main problem regards the duration of the validation and the reconcilement of the revealed measures. In fact, the validation of each measure could require some time with the consequence that sometimes the measure might not be validated owing to a time out of the system, not allowable for real-time analysis.
As a consequence the validation of the sensorial data needs to be improved as much as possible, and because of this a system of neural network has been studied for on-line validation.
The pre-processing phase is composed of:
1. *Elementary validation*

Initially a filtering process is applied on the acquired data in order to identify sensors temporarily or permanently out of service. During the filtering process some values of the input signals can be corrected or rejected dependently on the possibility to recover correct data. The values obtained for each sensor are normalised on a prefixed range (e.g. [-1,1]); then the normalised data are processed in order to substitute those values that are out of range with the most probable value for that sensor in that time by extrapolating it over the previous measurements.
2. *Measurements reconcilement*

The process of verification of the measurements congruence and reconcilement is performed using supervised neural networks (RNs), which are trained with the measurements acquired directly from the sensors and with the validate measurements: these last can be obtained according to a mathematical model derived from the knowledge of an expert. The number of neural networks to be trained corresponds to the number of correlated quantity groups. To each network a certain weight is associated so that in the case in which more networks calculate the same measure the reconciled data is obtained as a weighted average of the previous calculated values. The data reconcilement process allows individuating eventually incongruent measurements and, by recalculating the reliable values of the respective quantities, provides a set of measurements which are all each other congruent. The network is composed of the input and output layers and an additional hidden layer whose number of units can be found empirically. Even the choice of the number of the hidden layers can be done in the same way, based on observations regarding the activation of the specific layer. At the end of the training phase, the network can be reutilised to filter the input data acquired from the sensors. The output of this network is composed of a set of validated and congruent

variables which define, together with other parameters, a training set for the hierarchical networks *system*.

2.2 System monitoring and prediction through HNN

In order to properly train the HNN network, the set of stimuli (training set) should be chosen in such a way to be representative of all the classes that the network should recognize: more stimuli of a certain class are chosen for the training, more specialised will be the network to recognize the elements of that specific class and the number of excited neurons by such input will be greater. In particular, for the SOM, those parts of the network that during the training phase are tuned on the specific stimuli class are composed of a number of neurons as higher as the number of elements of that class in the training set (i.e., they have a higher resolution).

The training set is defined by a set of values, initially pre-processed, acquired on field and the use conditions of the examined object.

Aim of our study is then to design and implement a hierarchical neural network system able to work automatically in the context above mentioned.

The network used in the first phase of classification is a Self-Organizing map for which the steps of the training algorithm are the following [2]:

For each input pattern

1. *the neurons of the network compute their respective value of the discriminating function;*
2. *the neuron having the biggest value of such function wins the competition(the so called winner);*
3. *the winner determines the spatial location of the neurons with a certain topologic neighbourhood;*
4. *the winner apply a modification to his synaptic weights to increase his activation after the presentation of a similar successive input pattern. The weights of a generic neuron "n" in the range of the winner influence are also modified inversely proportional with the distance from the winner.*

The output of the clustering level is used to form the input pattern, together with known historical-statistical information, for the prediction level, which refines the classification and gives a final response, that is a some type of prediction on the examined object.

The network used is a feed-forward back-propagation network. The dimension of the input stimuli corresponds to the number of different historical-statistical features taken into account, plus the SOM output for a total of $N_{EBP.}$

In Fig. 2 the architecture of the prediction level is shown. The input layer is composed of n_I neurons, the output layer is composed of n_O neurons that correspond to the number of different classes of prediction to be identified. The network is used with no hidden layer; experimental tests have shown that hidden layers do not improve the performance nor the quality of the results.

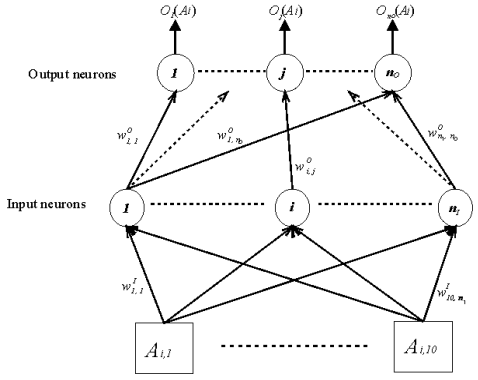

Fig. 2 The architecture of the prediction level.

The training algorithm is composed of the following steps:

For each input vector A_i do

1. *Get the input from the clustering level;*
2. *Append historical –statistical data,*
3. *Compute the output of the network propagating the input through weights;*
4. *Compare the output with the target class $C_{i,T}$ and evaluate the error $\delta_j(A_i)$;*
5. *Back-propagate the error and updates the weights up to the input layer;*

After training, the network can be used as classifier. The algorithm for this prediction phase consist of the following steps:

For each input vector A_i to be classified do:

1. *Get the input from the classification level;*
2. *Append of the historical-statistical data;*
3. *Compute the output of the network propagating the input;*
4. *Take the most excited neuron i of the output level as the finale classification of A_i;*

The output of this network is the final prediction for the processed object, in terms of which of the n_O different output classes of prediction has been associated with the object examined.

3 Materials and methods

The implemented hierarchical neural network has been tested on data obtained from measurements furnished by Alenia. In this context, the developed algorithms and the networks can be easily integrated inside a single aircraft Life Cycle Monitoring (LCM) system, so that an interpretation of the data examined can be presented friendly and quickly also providing an evidence of the significant classes for the diagnosis and prediction.

Regarding the examined data, three different test conditions have been chosen, maintaining a fixed temperature of 100°C (Fig. 3):

1. a load charge on two jacks A and B with 112% of the limit load
2. a load charge on five jacks A, B, K, L, and M with 112% of the limit load
3. a load charge of five jacks A, B, K, L, and M with 100% of the maximum load.

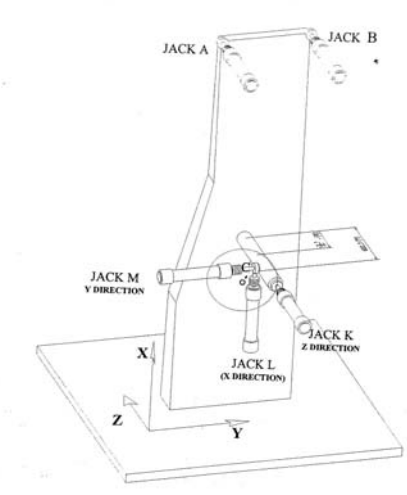

Fig. 3 Drawing of the component with jacks.

The deformations of the material under solicitation were measured using strain gages, having one or three channels, with respect to the main direction.

Each set of measures consisted of:

1. the progressive load charge percentage with respect to the limit one
2. the load charge on each jack
3. the deformations of each strain gage for every channel
4. the temperatures revealed from two thermocouples.

The training of the networks for the measurement reconcilement has been performed using the Stuttgart Neural Network Simulator (SNNS) [5]. Each network can be created using the SNNS graphic interface so that different tests can be easily performed to individuate the most promising parameters for a rapid and correct learning of the network. Once the best parameters are individuated, we can write a program in "batchlanguage" language (something between Pascal and C) to perform the training in a batch way. At the end of the training phase, the network can be reutilised to filter the input data acquired from the sensors. The output of this network is composed of a set of validated and congruent variables which define together with other parameters a training set for the hierarchical networks system.

The HNN architecture above described has been realized using SOM and EBP networks implemented under MATLAB™ software package. In particular for the clustering level (SOM) four typologies of networks have been trained having 5×8, 10×10, 20×20 e 20×30 neurons, respectively, distributed over a single layer and with hexagonal connection topology, for the first three, and a rectangular one, for the last. Each network has been implemented using a *distant function* based on both the connections and the real Euclidean distance among neurons; the results obtained in the two cases have not shown appreciable differences.

The parameters to control are the following: the name of the variable that physically contains the network data structure; the type of network, in this case a SOM; the variation range of each element of the input array; the network dimension, defined by a couple of values : e.g. [10,10] points out a 10×10 neurons network; the connection topology: hexagonal (HEXTOP), rectangular (GRIDTOP) or random (RANDTOP); the distance function: based on connection arcs (LINKDIST), based on the effective Euclidean distance between neurons (DIST) or based on the Manhattan distance (MANDIST); the initial value of the learning factor for the ordering phase; the number of training cycles of the ordering phase; the initial value of the learning factor for the tuning phase; the minimum width value that neighbourhood influenced by the winner can assume.

The input data for the SOM are arrays. In particular during the training a 2D input array (9x74000) is passed to the network; it represents a set of 74000 patterns each of them is composed of a 9-dimension features array. The elements of each array correspond to 9 different reconciled values of sensor.

Fixed the weights of the arcs being input to a generic neuron "*n*", $P_n(t)=(p_{n1},.....,p_{n9})$, at time *t*, and established a stimulus $v=(v_1,...,v_9)$, the winner "*w*" is defined by the following function:

$$w=\Phi_p(v):V\rightarrow SOM\ ,\ \text{where}$$

$$\sum_{i=1}^{9}p_{wi}v_i=\max_{n\in SOM}\sum_{i=1}^{9}p_{ni}v_i$$

where *V* is the stimuli space and *SOM* is the neurons space. The function $H(n,w)$ that updates the weights of the winner and the neurons inside a prefixed neighbourhood has a gaussian-like behaviour in such a way to follow the descending behaviour of the modification, with the distance from the winner:

$$H(n,w)=\exp\left(-\frac{(n-w)^2}{2\sigma^2}\right)$$

where "σ" controls the amplitude of the interval influenced by the winner.

The weight variation of a generic neuron "*n*" in the range of the winner influence is defined by:

$$P_n(i+1)=\xi\cdot H(n,w)\left(v-P_n(i)\right)$$

where "ξ" is a numeric constant that controls the entity of the modification during the training step of the network.

The Neural Network Toolbox allows to load from the Matlab workspace values arrays as training set.

After the network has been trained it can be exported in the Matlab workspace and then used to classify new input values. Then an implemented module for the SOM simulation can be used to facilitate the use of the trained SOM. Since the response of the network to each single input is related to the neuron excited by the input itself, we have also implemented the possibility to display a 3D plot of the network output for each stimulus.

In the simulation phase, the input patterns are passed one by one to the previously trained SOM and in order to have a graphic display of the network output, a tool has been developed that shows which is the neuron more excited by the input, on a lattice with the same dimension of the network.

The output of the clustering level, composed of a value pair identifying the neuron more excited, is used to form the input pattern, together with known historical-statistical information, for the prediction level implemented using the feed-forward network.

As an example, considering a 10x10 SOM, the result obtained are highlighted in Fig 4: regarding the three different test conditions examined: condition 1 is represented by red bullets, condition 2 by blue bullets and condition 3 by the yellow ones. Then, each of the conditions can be recognised and grouped as a specific cluster over the network.

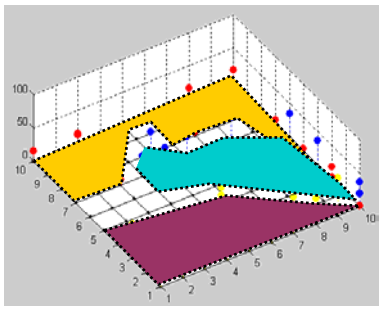

Fig. 4 Clusters of the different three conditions.

The first step is to create the network object. A function is implemented to create a feed-forward network. It requires four inputs and returns the network object. The first input is a 10x2 matrix of minimum and maximum values for each of the 10 elements of the input vector; the second input is an array containing the sizes of each layer. In particular, our network is composed of two layers, the first one by 15 neurons and the second one by 5 neurons. The other two input are a cell array containing the names of the transfer functions to be used in each layer and the name of the training function to be used. The function for the network creation automatically initialises the weights. Once the network weights and biases have been initialised, the network is ready for training.

The training process requires a set of examples, network inputs A_i and target outputs $C_{i,T}$. The A_i input structure is a 10-dimension features array:

$$A_i=[o_1,o_2,p_1,p_2,....,p_8]$$

This input is composed of a first part that contains the output of the SOM ($[o_1,o_2]$) and of a remaining part that contains historical-statistical parameters ($[p_1,p_2,....,p_8]$) of the examined object.

The O_i output structure is a 5-dimension features array:

$$O_i=[o_1,o_2,o_3,o_4,o_5]$$

Each element of the output array corresponds to a well defined reference class. In particular the maximum value element represents the class corresponding to the final prediction. During training the weights and biases of the network are iteratively adjusted to minimize the network performance function. Having an input A_i, the

weights from neuron i to neuron j (i.e. $w_{i,j}$) are updated by a value $\Delta_{i,j}(A_i)$ computed as follows:

$$\Delta_{i,j}(A_i) = \eta \delta_j(A_i) O_i(A_i)$$

where η represents the *learning rate* and the $\delta_j(A_i)$ error is calculated as a function of the neurons output and the difference between the network output and the target class for the specific input A_i.

$$\delta_j(A_i) = \begin{cases} f_j'(net_j(A)) \cdot (C_{A,T} - O_j(A)) & \text{if } j \text{ is an output neuron} \\ f_j'(net_j(A)) \cdot \sum_{k \in \{\text{output neurons}\}} (\delta_k(A) \cdot w_{j,k}) & \text{if } j \text{ is an input neuron} \end{cases}$$

where $net_j(A_i)$ is the weighted sum of the ingoing signals to the neuron j for the input A_i and $f_j'(net_j(A_i))$ is the derivative of an activation function $f(\cdot)$ used to compute the output. The default performance function for feed-forward networks is mean square error MSE (i.e. the average squared error between the network outputs a and the target outputs *t*) computed as follow:

$$MSE = \frac{\sum_{i=1}^{N} \delta_i^2}{N}$$

The output of the training process is the updated network itself and all the information about the progress of training. The low number of inputs and the absence of intermediate neurons allow the network to be train-able in a very short time.
After training, the network can be used as classifier. The function implemented to simulate a network takes the network input A_i, and the network object *net*, and returns the network outputs that is the class which identifies the final output of our system, i.e. the life cycle prediction.

4 Discussion and conclusions

We have assessed that the use of neural networks, in particular SOM networks, with an increasing neurons number has shown a very interesting asymptotic behaviour. In fact, after the training, the nodes of the network with a low neuron number (2×8 or 10×10) are all tuned on a particular stimuli class (clustered). On the contrary, in neural networks with a higher neuron number we have noted more and more numerous clusters of nodes that have not be excited by any of the values configuration stored in the database of the input data. These neurons have been considered non-clustered, that is not distributed in clusters, since the stimuli to which they react were not present in the input database. In other words, it is possible that they belong to both a unique cluster and a variable number of clusters. Therefore, we can deduct that independently from the number of parameters used for the monitoring of the life cycle of an object, it is possible to find a SOM with a sufficiently high neurons number able to recognize all the possible states individuated by the parameter set. The use of SOM does not need to retrain the network when some change occurs in the problem domain (for instance, the addition of new diagnosis classes), showing that the developed methodology is a promising tool for the diagnosis activity.

Acknowledgements

The authors would like to thank the Alenia group, in particular Dr. Giovanni Cavaccini for his contribution in the preparation of the study case.

References

[1] Di Bona S, Pieri G, Salvetti O, A multilevel neural network model for density volumes classification, *Proceedings of the IEEE 2nd International Symposium on Image and Signal Processing and Analysis-ISPA'01*,Zagreb, Croazia: University of Zagreb, 2001, pp. 213-8.

[2] Kohonen T, Self-Organizing Maps, *Springer Series in Information Sciences*, second ed. Vol.30, Berlin Springer, 1997.

[3] Mavrovouniotis ML, Chang S, A hierarchical neural networks, *Comput Chem Eng*, Vol.16, No.4, 1992, pp. 347-69.

[4] Ersoy OK, Deng SW, Parallel, self-organizing, hierarchical neural networks with continuos inputs and outputs, *IEEE Trans Neural Networks*, Vol.6, No.5, 1995, pp. 1037-44.

[5] SNNS (Struttgart Neural Network Simulator) http://www-ra.informatik.uni-ebingen.de/SNNS/)

[6] Balakrishnan K., Honavar V, Intelligent Diagnosis Systems, *Journal of Intelligent Systems* ,Vol. 8, No.3/4, 1998, pp. 239-290.

[7] Puppe F., *Systematic Introduction to Expert Systems-Knowledge Representations and Problem Solving Methods*, Springer-Verlag, 1993.

[8] Kolodner J., *Case-Based Reasoning*, Morgan Kaufrnann Publishers, 1993.