Advances in Engineering Software 40 (2009) 725–730

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

# An incremental learning algorithm for function approximation ☆

Jacques M. Bahi [a], Sylvain Contassot-Vivier [b,*], Marc Sauget [a]

[a] LIFC, University of Franche-Comté, Belfort, France
[b] LORIA, University Henri Poincaré, Nancy, France

ARTICLE INFO

*Article history:*
Received 23 September 2008
Accepted 1 December 2008
Available online 24 February 2009

*Keywords:*
Neural networks
Function approximation
Incremental learning

ABSTRACT

This paper presents an incremental learning algorithm for feed-forward neural networks used as approximators of real world data. This algorithm allows neural networks of limited size to be obtained, providing better performances. The algorithm is compared to two of the main incremental algorithms (Dunkin and cascade correlation) in the respective contexts of synthetic data and of real data consisting of radiation doses in homogeneous environments.



## 1. Introduction

It has already been proved that a neural network can be used as a universal approximator [2,7]. However, the way to obtain a good neural network architecture is not given. The two most important problems about the use of a neural network as a function approximator are the choice of the network architecture and the learning algorithm. The higher the number of hidden units in the neural network, the longer the computation time of every epoch in the learning process. Moreover, if the size of the network is too small, the learning is impossible and if the training is too important, the network may over learn and loose its generalization capability. This problem is intensified when the function to interpolate is a high frequency function containing very sharp variations. The main objective of our work is to use a neural network as a universal approximator in the context of radiotherapy. The work presented in this paper takes place in the context of a larger project in collaboration with the IRMA team, directed by Makovicka, of the FEMTO-ST institute. The goal of that project, called Neurad [1], is to rapidly and accurately evaluate radiation doses in heterogeneous environments in order to be able to optimize the treatment planning of cancerous tumors. Thus, this work deals with a better way to perform the learning of a neural network in order to obtain a high accuracy level with a smaller number of neurons. Therefore, requiring shorter computation times during the exploitation phase.

In the following section, a brief state of the art of incremental algorithms is presented and two representative algorithms are detailed, the cascade correlation algorithm and the algorithm described in [3]. Then, our incremental learning algorithm is presented in Section 3. Finally, our algorithm is qualitatively and quantitatively compared in Section 4 to the two existing learning algorithms detailed in Section 2.

## 2. Classical incremental learning algorithms

Since the first developments of neural networks [10], one of the major problems has been their building and learning. Some algorithms have had an important success such as the very well-known back-propagation algorithm [9,13]. However, they present important drawbacks as they are quite slow at learning and they do not give any information on the optimal number of hidden units to be used. In order to overcome those problems, new learning techniques have emerged such as the incremental ones [11,3,8,5,4]. The main idea is to begin the learning with very few neurons (typically one), and to add new neurons (one by one) when the learning does not evolve any longer, until the whole process stabilizes. This method tends to produce networks with fewer hidden units and thus with shorter learning times. Nonetheless, most of the incremental algorithms have been developed for classification problems and not for function approximation ones. This may induce important differences since the constraints on these two kinds of networks are not the same. We describe in the following paragraphs two representative incremental learning techniques for multilayer approximator networks.

☆ This work was partially supported by the French LCC (Ligue Contre le Cancer) and the CAPM (Communauté Urbaine du Pays de Montbéliard, France).

* Corresponding author.
*E-mail addresses:* jacques.bahi@iut-bm.univ-fcomte.fr (J.M. Bahi), sylvain.contassotvivier@loria.fr (S. Contassot-Vivier), marc.sauget@iut-bm.univ-fcomte.fr (M. Sauget).

0965-9978/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2008.12.018

### 2.1. Cascade correlation

One of the most popular incremental algorithms is the Cascade correlation architecture proposed by Fahlman and Lebiere [4]. This algorithm combines two rules. The former is the cascade architecture, in which a new hidden neuron is added in a new hidden layer after its own learning. Once a new hidden neuron has been added to the network, its weights do not change any more. The latter is the learning algorithm which creates and installs new hidden neurons. For each new node, the algorithm tries to maximize the magnitude of the correlation between the output of the new node and the residual error signal of the network. The particular architecture of this technique is presented in Fig. 1.

### 2.2. Incremental algorithm proposed by Dunkin et al.

As in the cascade correlation learning technique, this algorithm tries to minimize the number of nodes to train at each step of the learning process. However, it does not generate multiple hidden layers but conserves only one hidden layer. At the beginning, a minimal network architecture with only one hidden node on the hidden layer is trained. That training is stopped when the reduction of the squared error is lower than a given threshold. Then, the algorithm successively adds new nodes to the hidden layer as long as needed. At each addition of a new node, the previously added neurons are unchanged and only the new neuron is trained to match the difference between the objective output and the output of the previous network. That construction principle is depicted in Fig. 2.

The main problem with these algorithms is that, although they give good results in approximating synthetic data such as analytic functions, they do not always give satisfactory results with real world data obtained from physical phenomena, as shown in Section 4. Effectively, such real data may contain variable noise and very high frequency variations. In the context of the Neurad project, the radiation doses sharply vary according to the position in the environment and are perturbed by noise increasing with the distance from the surface of the environment. However, all that information is necessary to accurately evaluate the doses in heterogeneous environments. Furthermore, since the neural network we construct with our incremental method is to be used in an optimization process, its activation time must be as fast as possible. Thus, it is necessary to design a learning algorithm allowing to accurately approximate those real data while minimizing its learning process and its activation time, implying a minimal number of hidden neurons. This is why we decided to design a generic incremental learning algorithm.

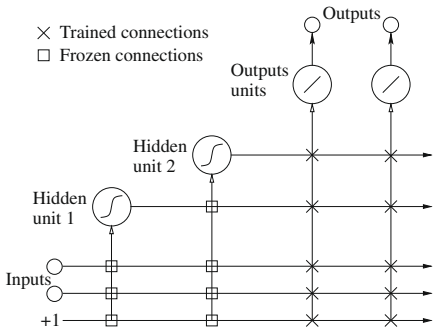


**Fig. 1.** The cascade correlation network architecture.

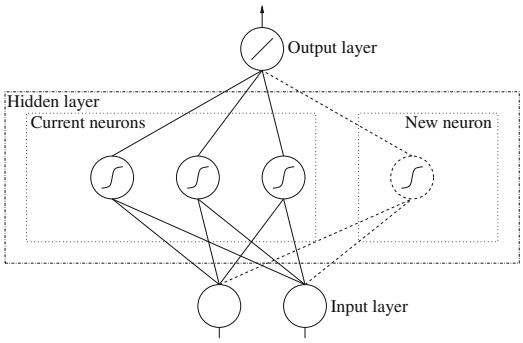


**Fig. 2.** The incremental algorithm by Dunkin et al.

## 3. Incremental learning algorithm used in Neurad

This section details our new incremental learning algorithm. A first part is devoted to the presentation of the network structure, we have chosen in order to get an efficient approximator and the second part details our learning algorithm.

### 3.1. Network structure

Many results have shown that a multilayer neural network can be used as a universal approximator [2,7]. Generally, three layers (input, hidden and output) are sufficient to obtain the desired results. The number of neurons in the input layer is determined by the number of parameters of the objective function. In the same way, the number of neurons in the output layer is directly dictated by the number of outputs of the objective function. In the case of radiation doses evaluation, the number of neurons in the output layer is reduced to one neuron which delivers the dose. Finally, the number of neurons in the hidden layer is the most difficult one to determine. It does not directly depend on the number of inputs and outputs of the problem and there is no precise rule to compute it. In fact, this number of neurons rather influences the ability of the network to approximate high degree functions. However, it is not a good idea to coarsely overestimate that number since that sharply increases the learning time and may thus make the network unusable. Thus, to bypass that problem, we have designed an incremental learning method which automatically sets up the number of hidden neurons.

In addition to the learning algorithm, some slight modifications of the classical structure of the network can enhance its capacity to approximate high degree functions with fewer neurons, and thus, to be trained faster. In our context of accurate approximation, the HPU (Higher-order Processing Unit) structure [6] allows us to obtain better results by artificially increasing the number of inputs of the network. In fact, an HPU neural network has additional inputs which are polynomial combinations of the original inputs up to a maximal degree (referred to as the order of the network). For example, the inputs of an HPU network of order 3 corresponding to an original network with two inputs $(x_1, x_2)$ are $(x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3)$.

This modification of the original network tends to facilitate the approximation of high order areas in the original function, such as sharp variations, and thus to speed up the learning phase. However, although it is strongly suggested to use this HPU structure to enhance the approximation efficiency, it is not indispensable.

Another structural modification which can enhance the results of our neural network for some specific data sets is to replace the linear output neurons by sigmoid ones. That is the case with the sets of radiation doses.

### 3.2. Learning

The classical learning method used with multilayer networks is the back-propagation. Nevertheless, although this method gives good results, the learning process remains slow, even with HPU networks. Among all the possible optimizations of that process (see [14] for a survey of the existing optimizations), the Resilient back-propagation (RPROP) [12] is one of the most efficient ones.

Contrary to the classical back-propagation, the RPROP algorithm only uses the sign of the error derivative to update the weights and the updatings are performed, for each weight, with a distinct value independent from the error. That modification value is respectively increased or decreased, similarly to an acceleration or a deceleration, whether the error evolves in the same direction or not.

However, even if the combination of the HPU structure and the RPROP learning gives better results than the classical approaches, it does not avoid the problem of fixing the number of hidden neurons. Effectively, there is no a priori information which may indicate what is the best suited number of hidden neurons to accurately approximate a given function. Thus, we have designed an incremental learning algorithm which starts with a given number of hidden neurons and which automatically adds hidden neurons when needed during the learning process.

The principle of our algorithm is to perform a RPROP learning over the current HPU neural network until the error either reaches the required accuracy or does not sensibly evolve anymore according to a given threshold (Fig. 3).

In the first case, the neural network has the desired accuracy and the learning process stops. In the second case, the learning limit of the current neural network is considered to be reached. Then, a neuron is added to the hidden layer in order to increase the approximation ability of the network. So that the initial network does not deviate from its optimization path, the addition of the new hidden neuron is performed without modifying the other neurons and links and the added neuron is initialized with null weights and threshold. After that, the learning process is resumed with that new configuration of neural network while allowing the modification of any weight and threshold in the network.

That fact of allowing the modification of any component in the network after the addition of a new neuron is quite different from the previous incremental algorithms. Most of the previous approaches tend to apply the learning phases to only one neuron in order to converge faster. Unfortunately, that gain of time is achieved at the cost of an important loss of accuracy as it is far more difficult to adapt only one neuron to a residual signal than the whole set of neurons.

So, once the new neuron is added, the learning process starts again and the links and threshold of all the neurons (including the new one) automatically evolve to reduce the error of the network (see Fig. 4). Then, the incremental process repeats itself until the desired accuracy is reached or the evolution of the error between two consecutive configurations of the network becomes too small (under a given threshold). In that last case, it is assumed that the overall limit of the network has been reached and that adding hidden neurons will not improve the results.

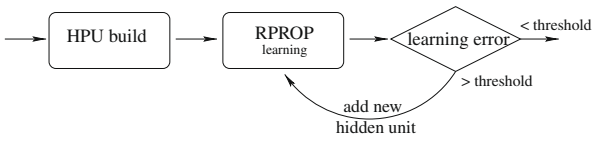


**Fig. 3.** The incremental learning building rule.

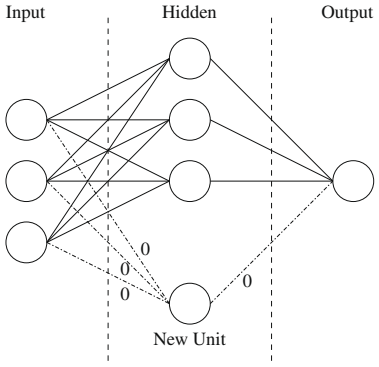


**Fig. 4.** Addition of a new unit in the hidden layer.

As in other learning algorithms, the specification of a validation data set is possible in order to control the learning process and avoid over-learning. Additionally, the possibility of specifying an upper bound to the number of hidden neurons has also been included in the learning process in order to limit the size of the network and, as a consequence, the learning time in case of extremely slow convergence. Such slow convergence mostly happens with very high required accuracies.

Finally, this incremental learning process allows us to build and train efficient and accurate neural networks for function approximation while limiting their number of neurons. Moreover, even if this algorithm has been designed for a particular need in the Neurad project, it is usable for any kind of data, real or synthetic. Moreover, this learning algorithm is not limited to a given feed-forward network architecture since it can be applied to any kind of neurons (linear or sigmoid) which is not the case of Dunkin et al.'s algorithm [3] which can only be applied to linear outputs.

The efficiency and accuracy of our algorithm is compared in the next section against two other existing incremental algorithms for function approximation.

## 4. Testing and evaluation

In this section, we compare our new incremental algorithm with two other popular algorithms: the cascade correlation [4] and the incremental learning technique of Dunkin et al. [3], in different contexts. Our algorithm has been implemented in standard C++ and used on a classical workstation.

In order to give a pertinent comparative basis, we begin our comparison with one of the analytic functions used in [3]. Then, real data containing radiation doses in two homogeneous environments are used.

### 4.1. A classical function

The first function used in our tests is the function two of paper [3]. It is a classical two dimensional function of the form:

$$f(x,y) = (\exp(\cos(4 * (x+y)))/2)$$

whose representation is given in Fig. 5.

Ten training sets respectively containing 20, 30, 40, 50, 60, 70, 80, 90, 100 and 125 points distributed on the $[0,1]^2$ range were used in order to study the influence of the size on the results. In order to get relevant information about the function to approximate on the domain of interest, most of the points in each set were uniformly distributed in a square mapped onto the domain, using the square root of the number of points. For the sets whose the size was not a perfect square, the few additional points were randomly distributed on the domain.

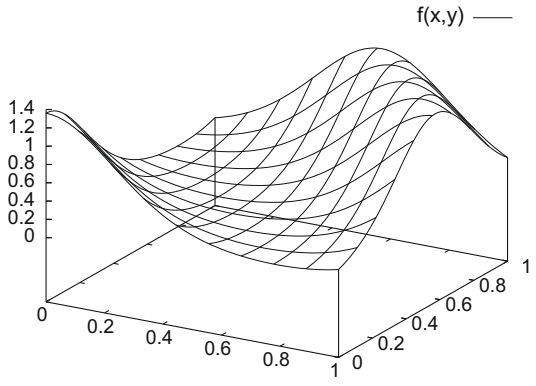


**Fig. 5.** Aspect of the classical test function.

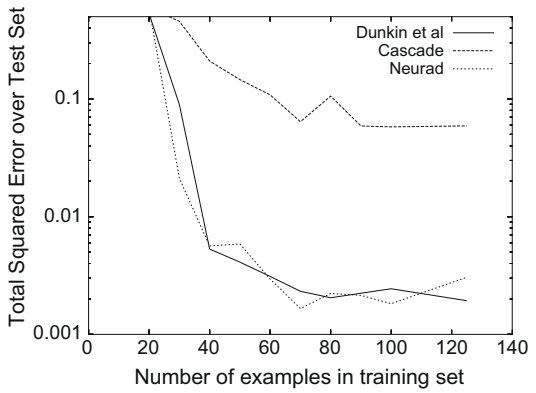


**Fig. 6.** TSE in function of the size of the training set.

For each complete learning, the error threshold used to stop the process is set to $1e-4$. Once each network is trained, the Total Squared Error is computed for a set of 25 points randomly chosen on the considered domain of the approximated function. Random points are taken in place of the points in the training set in order to see if the generalization of the neural network is efficient. The results are given in Fig. 6 in which the TSE is plotted as a function of the size of the training set for each learning method. The results presented are the average of ten successive learnings for each algorithm. It can be seen that Dunkin et al.'s algorithm and ours have a rather similar behavior which is far better than the cascade correlation one. Our algorithm tends to oscillate slightly more than Dunkin et al.'s one but its TSE follows the same general decrease. This is quite important since our algorithm is to be used with large training sets.

### 4.2. Real world data

The most difficult case which may occur in function approximation is when dealing with real world data obtained, for example, from a physical phenomenon. The major difference with the previous kind of data is that they usually contain some noise with unknown properties.

In our context, we consider the radiation dose distribution located on a plane in the middle of the tridimensional environment and aligned with the axis of the accelerator as shown in Fig. 7 (left). The data structure used to represent the plane of interest is a two dimensional discrete grid in which the absorbed dose is given at each discrete position in that grid. An example of dose distribution on that plane is given in Fig. 7 (right) for an environment of water.

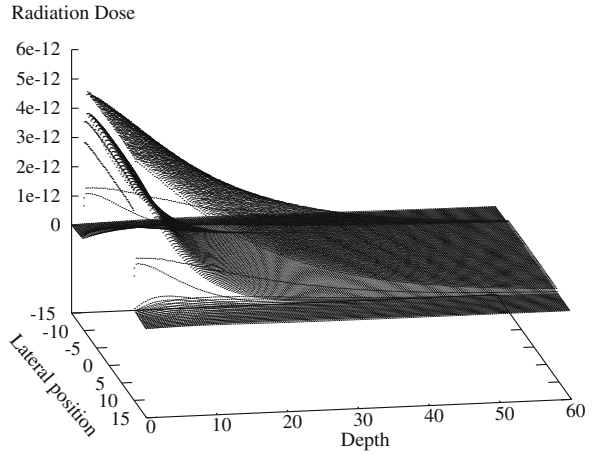


**Fig. 8.** The training set composed of two dose distributions.

**Table 1**
Results of the training with radiation dose distributions from two homogeneous environments.

| | Dunkin et al. | Our algorithm |
|---|---|---|
| Mean error (%) | 4.03 | 1.38 |
| Mean bias (%) | 0.66 | 0.032 |
| Min error (%) | 2.7e−4 | 1.3e−5 |
| Max error (%) | 429.9 | 179.9 |
| # Hidden neurons | 37 | 50 |
| Learning time (s) | 64,446 | 69,209 |

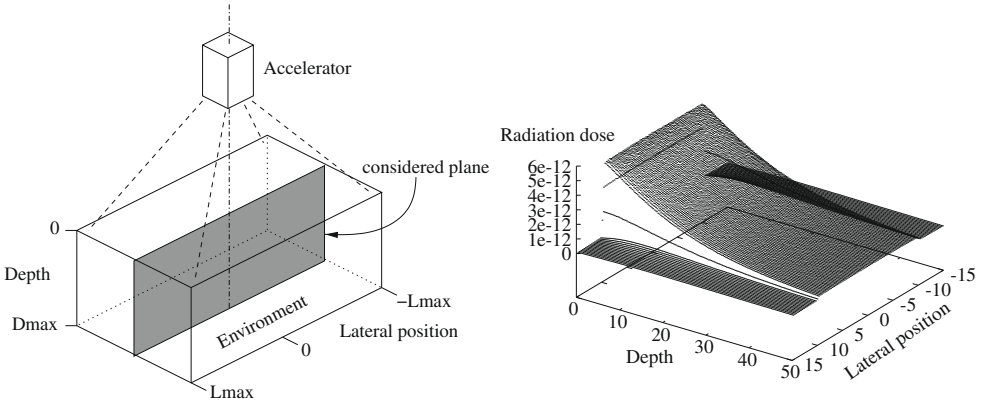


**Fig. 7.** Location of the plane of interest in the environment (left) and dose distribution on that plane in a homogeneous environment of water (right).

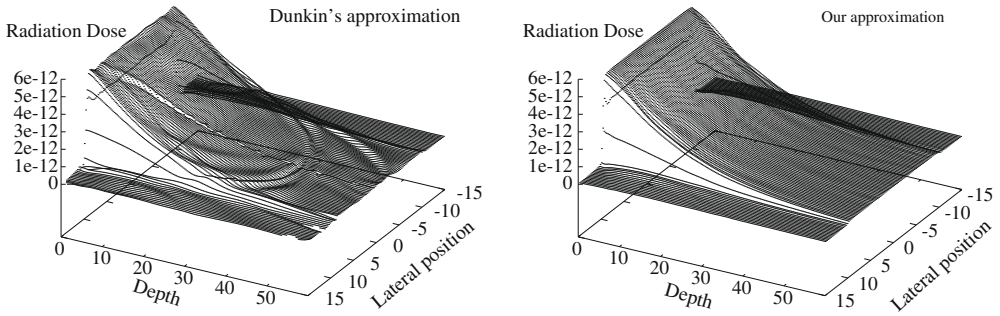


**Fig. 9.** Approximation with Dunkin et al.'s learning algorithm (left) and approximation with our learning algorithm (right).

Such dose distribution can be obtained either by physical measures or by Monte-Carlo simulation. The former is quite difficult to perform and not very accurate since it requires the introduction of sondes in the environment which themselves interfere on the measures. The latter is far more accurate but very slow as it requires a large amount of computation.

The training set used in our comparison is composed of two distinct dose distributions in homogeneous environments respectively of water and titanium as illustrated in Fig. 8. The two dose distributions can be distinguished in the shallower depths. The total size of the training set is 40,000 points (20,000 points for each dose distribution) uniformly distributed on the plane of interest.

The results obtained with Dunkin et al.'s algorithm and ours are presented in Table 1. The results of the cascade algorithm are not presented here since its convergence was so long that we have considered it was not reasonably usable in such a context. In fact, based on the results obtained with that algorithm on the classical function described above, it could not be expected for that algorithm to perform better with the doses set considering the higher complexity involved in this problem.

It can be seen that our learning time and size of network are larger than those of Dunkin et al.'s algorithm. The increase of the learning time comes from the fact that in our algorithm, all the weights of the neurons can be modified at each learning step. This implies a larger number of computations at each step and a larger overall learning time. Concerning the size of the network, our algorithm does not seem to be optimal. However, it is quite difficult to precisely interpret such results since they are directly linked to the final accuracy of the network, which is better with our method. So, if Dunkin et al.'s algorithm had been able to attain the same accuracy as our method, it can be expected that the size of the network obtained would have been closer to the one produced with our method.

In terms of accuracy of the results, our algorithm outperforms Dunkin et al.'s algorithm. Better results are obtained on the mean bias, mean and maximum error. This is important since the mean error alone does not ensure an overall good quality of the approximation. So, those results indicate that our learning algorithm produces a network which better approximates the dose distributions. This is confirmed by the overall aspect of the approximations obtained by the two networks produced, as presented in Fig. 9. It can be seen that the result of Dunkin et al.'s network contains undesired oscillations on the surface of the curve which are not present with our network.

## 5. Conclusion

An incremental learning algorithm for feed-forward neural networks for function approximation problems has been presented.

This algorithm is decomposed in learning phases in which the neural network is trained using any classical learning algorithm. Between each learning phase, if the network has not reached the desired accuracy, a new neuron is added to the hidden layer and a new learning phase is started. The main difference between this algorithm and previous ones such as Dunkin et al.'s algorithm is that all the neurons are adjusted at each learning phase. This implies a slightly larger learning time but produces far better accurate networks.

The experimental comparison with two other well-known incremental algorithms has pointed out the good behavior of our algorithm with synthetic data as well as with real data. Better levels of accuracy are critical in numerous applications and especially in the medical context. In the case of radiation dose evaluation, since the neural network is used to evaluate dose distributions in human body environments, the error must be under 3% in some particular locations in the radiated environment and must never be over 5%. As presented in [1], the results obtained with our learning method have an error under 1%. With such accurate networks and a particular algorithm to evaluate radiation dose in any kind of complex environment (human body), it becomes possible to design an optimization process allowing the treatment planning of cancerous tumors to be enhanced.

Finally, even if our incremental algorithm gives good results, some aspects could be further investigated. The two main points which seem interesting are the speed up of the learning process using, for example, a parallel approach with domain decomposition, and the possibility for an existing neural network to learn new data using our incremental algorithm.

## References

[1] Bahi J, Contassot-Vivier S, Makovicka L, Martin E, Sauget M. Neural network based algorithm for radiation dose evaluation in heterogeneous environments. In: Artificial neural networks–ICANN 2006, Athens, Greece. Lecture notes in computer science, vol. 4132/2006. Berlin/Heidelberg: Springer; 2006. p. 777–87.
[2] Cybenko G. Approximations by superpositions of sigmoidal functions. Math Control Signal 1989;2:303–14.
[3] Dunkin N, Shawe-Taylor J, Koiran P. A new incremental learning technique. In: Neural Nets Wirn Vietri-96. Proceedings of the eighth Italian workshop on neural nets. Springer Verlag; 1997. p. 112–8.
[4] Fahlman SE, Lebiere C. The cascade-correlation learning architecture. In: Touretzky DS, editor. Advances in neural information processing systems, Denver 1989, vol. 2. San Mateo: Morgan Kaufmann; 1990. p. 524–32.
[5] Fritzke B. Fast learning with incremental RBF networks. Neural Process Lett 1994;1(1):2–5.
[6] Ghosh Joydeep, Shin Yoan. Efficient higher-order neural networks for classification and function approximation. Int J Neural Syst 1992;3(4):323–50.
[7] Hornik Kurt, Stinchcombe Maxwell, White Halbert. Multilayer feedforward networks are universal approximators. Neural Networks 1989;2(5):359–66.
[8] Kurkova V, Beliczynski B. An incremental learning algorithm for gaussian radial-basis-function approximation. In: Second international symposium on methods and models in automation and robotics; 1995. p. 675–80.

[9] Le Cun Y. Modèles connexionnistes de l'apprentissage. PhD thesis, Université Pierre et Marie Curie; 1987.
[10] Pitts W, McCulloch WS. A logical calculus of ideas immanent in nervous activity. Bull Math Biophys 1943;5:115–33.
[11] Polikar R, Udpa L, Udpa SS, Honavar V. Learn++: an incremental learning algorithm for supervised neural networks. IEEE Trans Syst Man Cyber Appl Rev 2001;31(4):497–508.
[12] Riedmiller M, Braun H. A direct adaptative method for faster backpropagation learning: The RPROP algorithm. In: Proceedings of the IEEE international conference on neural networks (ICNN93), San Francisco, April 1993.
[13] Rumelhart DE, McClelland JL, the PDP research group. Parallel distributed processing, vol. 1–2. The MIT Press; 1986–87.
[14] Tertois S. Réduction des effets des non-linéarités dans une modulation miltiporteuse á l'aide de réseaux de neurones. PhD thesis, Rennes 1; 2003.