Effect of noise on the identification of digitized Bragg Curves

J.J. Vega* and R. Reynoso
Departamento del Acelerador, Gerencia de Ciencias Básicas, Instituto Nacional de Investigaciones Nucleares, Apartado Postal 18-1027, 11801, México D. F., México

Recibido el 2 de marzo de 2006; aceptado el 18 de agosto de 2006

Recently, pulse shape analysis, PSA, assisted by artificial neural networks, ANN, used as pattern identifiers has received attention by several groups interested in analyzing different kind of signals. In this, as well as in many experimental fields, noise is a ubiquitous known undesirable problem when dealing with experimental signals, requiring a considerable effort to restrain it as much as possible in order to improve the reliability of the measurements. Nonetheless, the remaining noise demands a careful analysis in order to be able to asses its effect. In this paper we present results of the effect of noise on the performance of an ANN used to assist PSA of synthetic Bragg curves.

Keywords: Neural networks; Bragg curve spectroscopy; digital pulse-shape analysis; pattern identification.

Recientemente, el análisis de forma de pulsos, AFP, auxiliado por redes neuronales artificiales, RNA, usadas como identificadores de patrones, ha despertado la atención de varios grupos interesados en le análisis de diferentes tipos de señales. En éste, como en muchos otros campos, el ruido es un conocido e indeseable problema muy común cuando se manejan señales experimentales; requiriéndose de un considerable esfuerzo para disminuirlo lo más posible con objeto de mejorar la confiabilidad de las mediciones. Sin embargo, el ruido residual demanda un análisis cuidadoso para poder estimar su efecto. En este trabajo, se presentan resultados de los efectos del ruido sobre el desempeño de una RNA usada como auxiliar para el AFP de curvas de Bragg sintéticas.

Descriptores: Redes neuronales; espectroscopia de curva de Bragg; análisis digital de forma de pulsos; identificación de patrones.

PACS: 07.05.Kf; 07.05.Mh; 29.40.Cs

1. Introduction

Isotope separation of light ions using Bragg curve spectroscopy [1–5], BCS, is an interesting field where digital pulse shape analysis, DPSA, may be applied [6,7]. Based on computer simulations of the signal generated by a Bragg curve spectrometer, [7], it has been demonstrated that as a result of the distortion induced on the original Bragg curve, BC, due to the integration of the current across the Bragg curve spectrometer's Frisch grid to anode gap plus the one caused by the amplifier response, the Bragg peak height, BP, besides being a function of the ion atomic number, Z, it is also a function of its mass. Later on, using flash-ADCs, [8], they were able to resolve isotopes of light nuclei like Li and Be, etc. This is, when different isotopes are involved, they measured and recorded BC pulse-shapes, and if they were able to assign a Z value to that event, then a most probably generated reference shape was selected by the χ^2 criterion allowing specifying the range and the mass number.

In a previous paper [9], we presented a novel way to extract relevant parameters associated with the outgoing ions from nuclear reactions. It was based on digitizing the signals provided by a Bragg curve spectrometer, allowing the implementation of more thorough DPSA. Due to the complexity of this task, it was required to take advantage of new and more powerful computational paradigms. This was fulfilled using a back-propagation artificial neural network as a pattern identifier of synthetic BCs. We used the common technique of early stopping [10] in order to take care of over-training, which is a known problem during the training stage of an ANN [10–24]. The patterns analyzed in Ref. 9 were synthetic noisy BCs. A synthetic noise component was added to simulate any possible source of noise that normally goes with the experimental signal of interest. As it was expected, over-training, *i.e.*, over-fitting the data by the ANN during training, was observed. In this paper, we try to determine the effect of the size of the noise component on the appearance of over-fitting during the training stage.

2. DPSA and Bragg curve spectroscopy

Traditionally, BCS has been a two-parameter, (E_{Tot}, *BP*), analytical technique [1-5] making it easy to use, where E_{Tot} represents the total initial kinetic energy of the detected ion and *BP* its Bragg peak amplitude or maximum value of the specific stopping power of the ion ($S(E) = dE/dx \equiv$ Bragg curve) when it is stopped in a gas medium. These two signals are obtained by feeding the output signal from a Bragg curve spectrometer to two different electronic amplification branches; one with a large integration time (E_{Tot} signal) and the other with a short integration time (BP signal). In Ref. 9, a more powerful multi-parametric measurement approach was put forward, based on recording and analyzing the complete BC pulse-shape into 81 bins or parameters.

The new powerful DPSA approach pursued in this paper is described in Ref. 9. It is essentially based on analyzing synthetic BCs and saving them as 81-tuples of bins, discrete values or parameters $\{\mathrm{S(i)}\}_{i=1,81}$. In Fig. 1, it is shown examples of the used ideal BCs (solid line) together with a simulated synthetic experimental BC (dots) that includes a fast changing component taking into account any possible origin of experimental noise of sizes equal to 2, 6 and 10% (size of standard deviation).

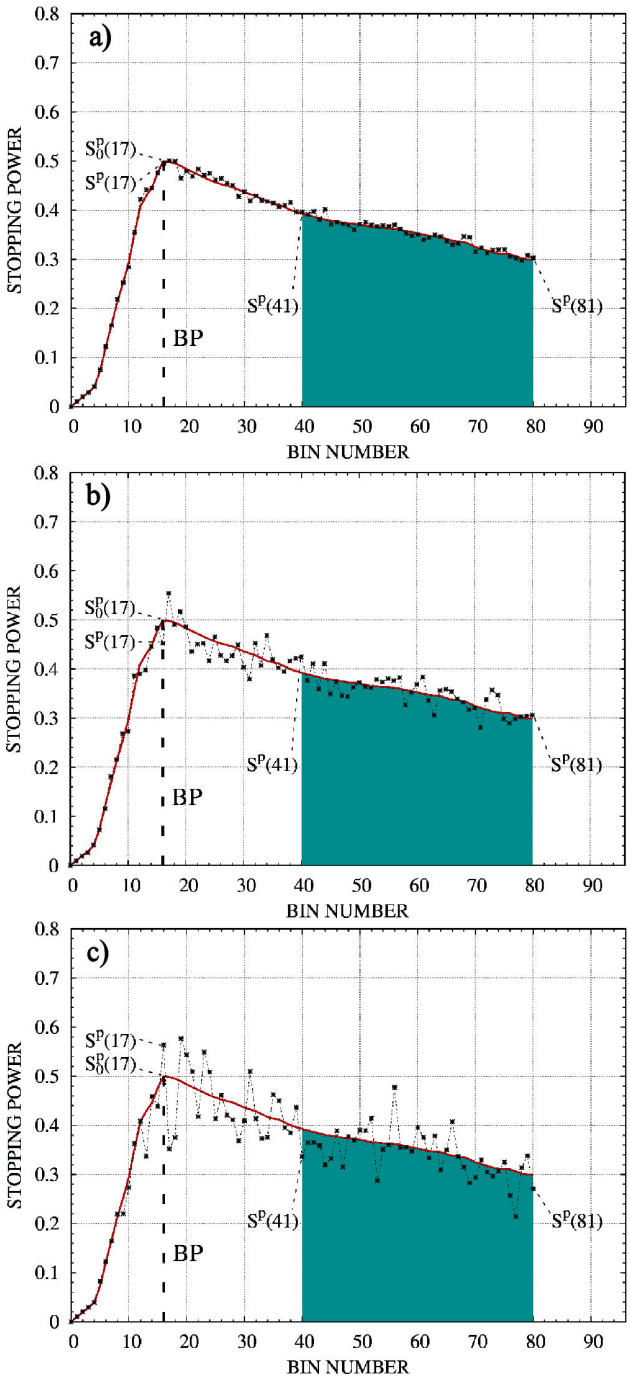

FIGURE 1. Plot of an ideal synthetic BC (solid line) together with a simulated synthetic experimental BC (dots) which includes a fast changing component that takes into account any possible origin of experimental noise. The training and validation data sets were built using BCs of a length consisting of at least 41 bins. a) size equal to 2%; b) size equal to 6%; c) size equal to 10%.

3. Artificial neural networks

Before one can use an ANN to solve any problem it has to be train in order to learn how to solve it, *i.e.*, by means of an algorithm, the learning law, it has to be determined the link array, $\vec{w}$, containing all the values of the links among the different interconnected neurons. In general terms, when following a supervised learning law, as the learning paradigm, the problem is to construct a function $f(\vec{x};\vec{w})$, an ANN, based on a data set of input-output pairs, $(\vec{x},\vec{y}^t)$, where $\vec{x}$ is an input pattern and $\vec{y}^t$ its corresponding classification, tutorial or target value, so that $f(\vec{x};\vec{w})$ approximates $\vec{y}^t$. During the training stage, the ANN is presented the patterns from the training data set many times, and it will learn by adapting itself to the data through a slow modification of its link array, $\vec{w}$, guided by a learning algorithm. A training epoch consists in presenting the ANN one time each one of the BCs that belong to the training data set in an arbitrary order. A way to measure how well an ANN is learning its task is by observing, as a function of the number of training epochs, the reduction in the training and validation sum of squares error functions calculated over the entire training and validation data sets, D^T and D^V, respectively, *i.e.*:

$$E^T(\rho)=\frac{1}{K}\sum_{p\in D^T}^{|\vec{y}^t-f[\vec{x}_p;\vec{w}(\rho)]|^2}$$

or (1)

$$E^V(\rho)=\frac{1}{K}\sum_{p\in D^V}^{|\vec{y}^t-f[\vec{x}_p;\vec{w}(\rho)]|^2}$$

where $\vec{w}(\rho)$ represents the link array after training the ANN for ρ epochs, K represents the number of patterns in D^T or D^V and $f[\vec{x}_p;\vec{w}(\rho)]$ is the ANN output for pattern p after ρ training epochs. Summarizing, in an error-correction learning algorithm, the goal of the learning process is to adjust the free parameters, the link array $\vec{w}$, so as to minimize the training sum of squares error function, $E^T(\rho)$, considered it as a cost functional over the number of training epochs ρ. The initial values of the link array components are chosen according to a uniform random distribution over the interval $[-0.5, 0.5]$. In order that the ANN keeps its generalization capability (the ability to identify patterns from the validation data set rather than the training data set), during the training stage, over-training has to be prevented, this means, one should impose an early stopping of the training process, this is, ones $E^V(\rho)$ reaches a minimum value, say at $\rho_{\min}$, even though $E^T(\rho)$ keeps on decreasing. In Ref. 9, it was shown that this is not exactly correct, in some cases it is required to keep on training the ANN a little bit more over $\rho_{\min}$.

For the problem of BCs identification using a feed-forward ANN, we chose the error back-propagation learning law including a momentum term [25], since it has proved to be efficient for pattern identification tasks [26–31].

4. Pattern representation and ANN architecture

The used data sets, D^Tor D^V, consist of discrete noisy synthetic BCs, $\{\text{S(i)}\}_{i=1,n}$, simulating experimental BCs be-

longing to 11 different BP values, all of them corresponding to 11 ideal noiseless (smooth) BCs, $\{S_0(i)\}_{i=1,81,}$ with an ideal tutorial value BP^t located at the 17^{th} bin, *i.e.*, $BP^t = S_0(17)$. The ANN output layer consists of two neurons with outputs corresponding to the predicted value of the total energy, E^o_{Tot}, and of the BP amplitude, BP^o, of the incoming ion. The number and the size of the hidden layers were determined on a trial and error basis, trying to find the simplest architecture capable of performing an acceptable identification of BCs. In this way, it was determined that 5 hidden layers of 9 neurons each were appropriate for the task. Rather than building the data sets using synthetic BCs spanning the whole range of 81 bins corresponding to the different possible discrete values of E_{Tot}, we used BCs corresponding to the 41 largest total energies only, *i.e.*, $41 \leq n$, this range is indicated in cyan in Fig. 1. In this way, one warrants that all the analyzed BCs achieve their BP, which, for all our ideal synthetic BCs, corresponds to the 17^{th} bin, considerably simplifying, in this way, the ANN task. Summarizing, the training and validation sets are defined as:

$$D^T \equiv \{[\{S^p(i; E^t_{Tot}(n), BP^t)]_{i=1,81}\}_{p=1,K}$$
$$\text{and} \qquad (2)$$
$$D^V \equiv \{[\{S^p(i; E^t_{Tot}(n), BP^t)]_{i=1,81}\}_{p=1,K},$$

where, for a given p, $[\{S^p(i; E^t_{Tot}(n), BP^t)]_{i=1,81}$ models an experimental synthetic noisy BC corresponding to a total ideal energy target value equal to and to an ideal Bragg peak target value equal to BP^t, where: $41 \leq n \leq 81$ and $S^p(i; E^t_{Tot}(n), BP^t) = 0$ if $n < i$. K is the number of patterns in each one of the data sets, and it is equal to 45,100, corresponding to 100 BCs for each one of the 451=11x41 different classes of BCs, 11 different BP^t values times the 41 different $E^t_{Tot}(n)$ values. The experimental noisy BCs were defined as:

$$S^p(i; E^t_{Tot}(n), BP^t) = S^p_0(i; E^t_{Tot}(n), BP^t) + S^p_G(i; E^t_{Tot}(n), BP^t) \quad (3)$$

where represents a smooth ideal BC (solid lines in Fig. 1), and a fast noise component that follows a Gaussian distribution with a mean value equal to 0 (dashed lines in Fig. 1) and an energy dependent standard deviation equal to $e \times S^p_0(i; E^t_{Tot}(n), BP^t)$, where e is equal to 0.02, 0.06 and 0.1 corresponding to error sizes of 2, 6 and 10%. The architecture of the employed ANN is a fully connected feed-forward network with: an input layer of 81 neurons, 5 hidden layers of 9 neurons each, and an output layer of 2 neurons. For all neurons, a sigmoidal nonlinear activation function was defined in terms of a logistic function, *i.e.*:

$$g(\vec{x}) = \frac{1}{1 + exp[-(\vec{w} \cdot \vec{x} - \theta)]}, \qquad (4)$$

where $\vec{x}$ is the neuron input array with information coming from all neurons from the previous layer which is pondered by the weight array, $\vec{w}$, associated with the links that carry each one of the inputs, and θ is the bias value of the neuron. In Table I, it is presented a summary of all the parameters of the ANN and of the training and validation data sets.

5. Results

In order to appreciate the magnitude of the pattern recognition task to be performed by the ANN, in Fig. 2, it is shown *BP* vs. E_{Tot} scatter plots of the ANNs input synthetic raw data; the crosses indicate the target values for each one of the 11×41 different pattern classes used to train the ANN. The three illustrated cases correspond to noise sizes equal to 2%,

TABLE I. ANN architecture and training parameters.

PARAMETER	VALUE
Learning law	Back-propagation with momentum term
ANN size	Input layer: 81 units
	par 5 hidden layers: 9 units each
	par Output layer: 2 units
	par Fully connected
$\alpha =$ learning rate	0.3
$\mu =$ momentum term	0.15
w initialization range	$[-0.5, 0.5]$
Order of pattern presentation	Shuffle
Activation function	Sigmoidal
Neurons update order	Serial order
$\mathbf{D^T}$ and $\mathbf{D^V}$ data sets	100 samples of each one of the 451 classes:
	par (11 *BP* values) x (41 E_{Tot} values)
CPU time (SUSE LINUX) Intel Pentium 4, 1.7 GHz	30 days of execution time for a training of 1,000,000 epochs

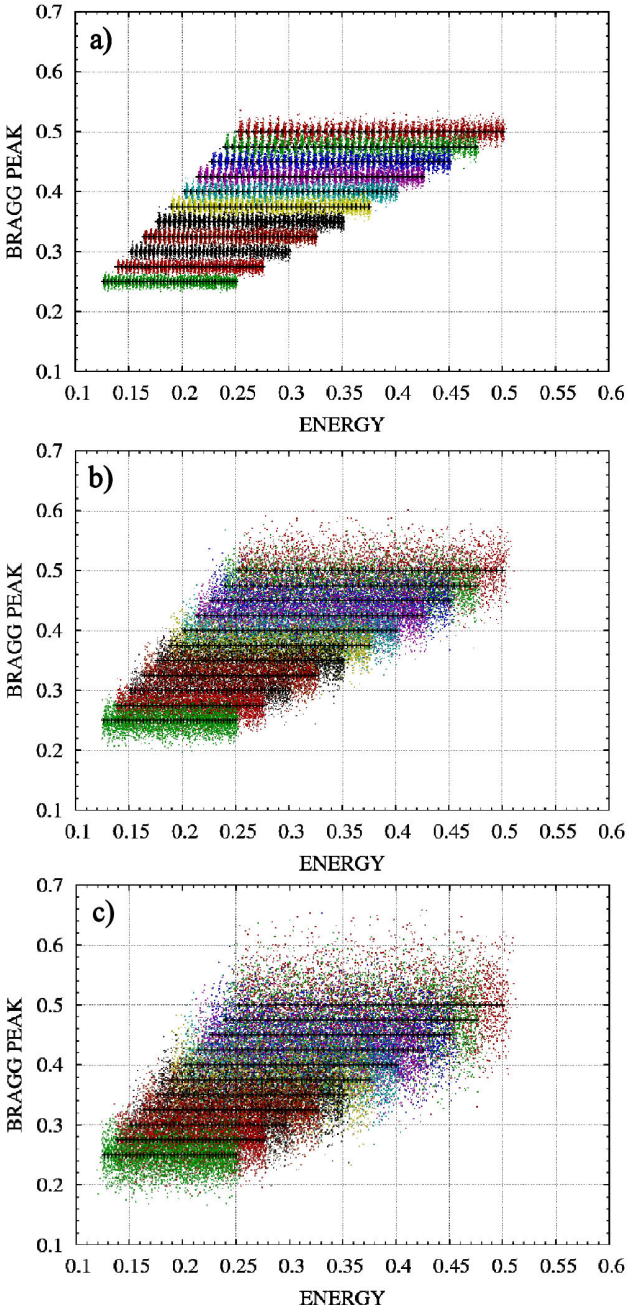

FIGURE 2. *BP* vs. E_{Tot} scatter plot of the ANNs input synthetic raw data. The crosses indicate the target values for each one of the 11×41 different pattern classes used to train the ANN. The error size of the synthetic BC data are a) 2%; b) 6%; c) 10%. The definitions of the two plotted quantities, *BP* and E_{Tot}, are explained in the text.

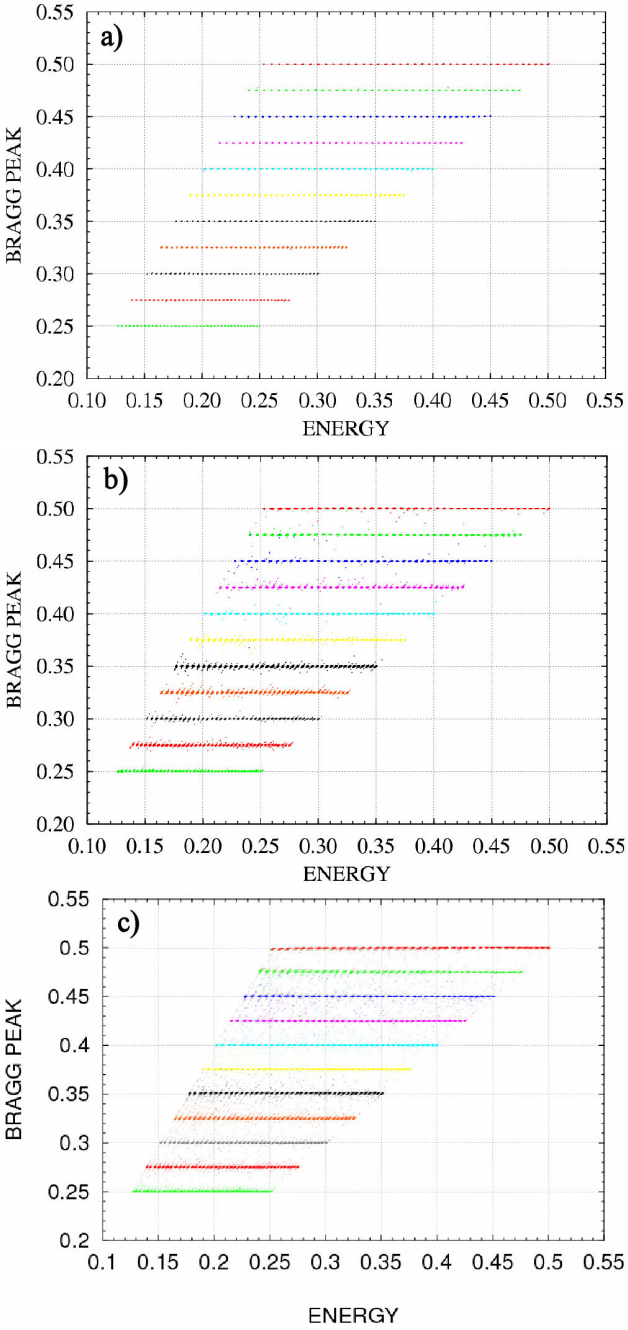

FIGURE 3. Scatter plot corresponding to the minimum error values of the $E^V(\rho)$ error curve corresponding to and error size equal to a) 2%; b) 6%; c) 10%. The minimum was reached after a) 2,600,000; b) 7500,000; c) 196,000 training epochs.

6% and 10%. The two plotted quantities were obtained in a very simple fashion. Since each one of the ideal BCs reaches its BP at the 17^{th} bin, then a simple way to define the *BP* value for the simulated synthetic BCs is to define it as $BP = S^p(17; E^t_{Tot}(n), BP^t)$. In relation to the value of E_{Tot}, it was defined as $E_{Tot} = \sum_{i=1}^{n} S^p(i; E^t_{Tot}(n), BP^t)$, which is a very simple way to estimate the area under the specific stopping power curve. It is reminded that only BCs, with $41 \leq n \leq 81$are analyzed. The task for the ANN is to learn from the training data set D^T, and from this to generalize, *i.e.*, to provide output values (BP^o, E^o_{Tot}) that cluster near the corresponding target values (crosses in Fig. 2) when the ANN is presented new data. For comparison, in Fig. 3, the predictions of well-trained ANNs for this task are shown.

In Ref. 9, in order to see how the number of training epochs depends on the initialization of the ANN, this is, on the starting values of the link array, $\vec{w}_0$, it was decided to

train 16 times an ANN using in all cases the error back-propagation learning law with a noise size of 10%. It was observed that the minimum of the validation sum of the square error function $E^V(\rho)$ is reached at quiet different values of ρs from $\rho_{\min}$= 136,000 epochs, the fastest learning one, up to $\rho_{\min}$ = 847,000 epochs; but, in all cases, the minimum ρ values are approximately equal. In addition, it was observed that it is a little bit over $\rho_{\min}$ when over-fitting to the noise onsets and that was confirmed by watching at the corresponding *BP* vs. E_{Tot} scatter plots. From this behavior, it was concluded that the error surface landscape is quite complex. This means that the random initialization value of the link array, $\vec{w}_0$, sometimes picks values that fall in regions of the error surface with a large slope and some other times fall in regions with a small one. In the first case, the learning or weight adaptation will be faster than in the second one. Now, in this paper, we are interested in studying the effect of the size of the noise component [size of standard deviation of , see Eq. (3)] on the onset of over-fitting. For that purpose, it is quite illustrating to look at the training and validation sum of squares error functions $E^T(\rho)$ and $E^V(\rho)$ corresponding to the error sizes we are interested in, 2, 6 and 10%, Fig. 4. Perhaps the most obvious feature observable from these figures is that over-fitting is present for error sizes equal to 6 and 10% but it is not present for a small error size as 2%. This means that, as expected, the amount of noise present in the data strongly determines the generalization capability of an ANN. In all three cases, the ANN learns in a step-like fashion, *i.e.*, its learning capability, as measured by the error-decreasing rate, speeds up at certain number of epochs and then slows down for a while until it reaches the next step. This behavior ought to be a consequence of the shape of the error surface in link space with many ridges; this is, when the link array w gets close to one ridge, and falls through it, it causes a fast learning rate, which explains the observed steps. Since the error surface is quite complex, it is not possible to conclude anything from the number of epochs it takes, in the different cases, to reach their corresponding minimum value. This is, as was shown in Ref. 9, the number of epoch required to get to the minimum depends a lot on the initialization value of the ANN link array $\vec{w}_0$. Nonetheless, one can compare the scatter plots for different error sizes corresponding to a specific error values. For example, at 600,000 epochs, the error values of the $E^V(\rho)$curves corresponding to 2% and 6% are both approximately equal, the first one is 0.0476 and the second one is 0.0487. Fig. 5 display the two scatter plots corresponding to 600,000 training epochs. It can be seen in these figures that in the 6% error case, the ANN has been able to learn all the groups reasonably well but, definitively, there are patterns with a bad identification. In the 2% error case, the ANN is able of a better classification of all the groups (one could say a perfect classification), but it has not learnt all the groups with the same accuracy. The groups with BP tutorial values equal to 0.5, 0.475 and 0.45 have been learnt better than the rest of the groups. In the 2% case, it is only after 2,600,000 epochs that the ANN learns all the groups with a

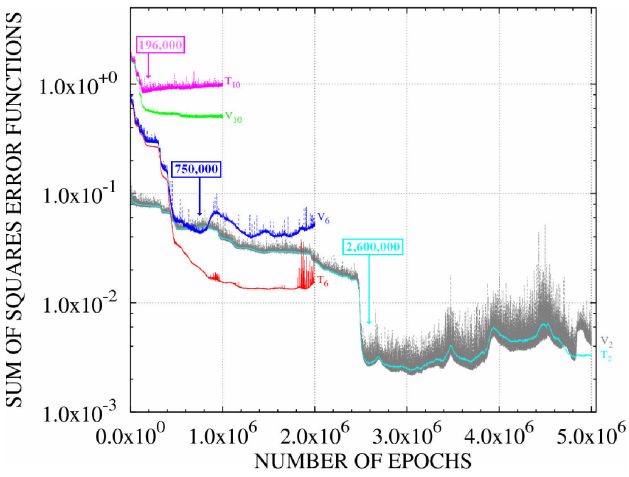

FIGURE 4. Training and validation sum of squares error functions $E^T(\rho)$ and $E^V(\rho)$ corresponding to the error sizes equal to: 2%, 6% and 10%.

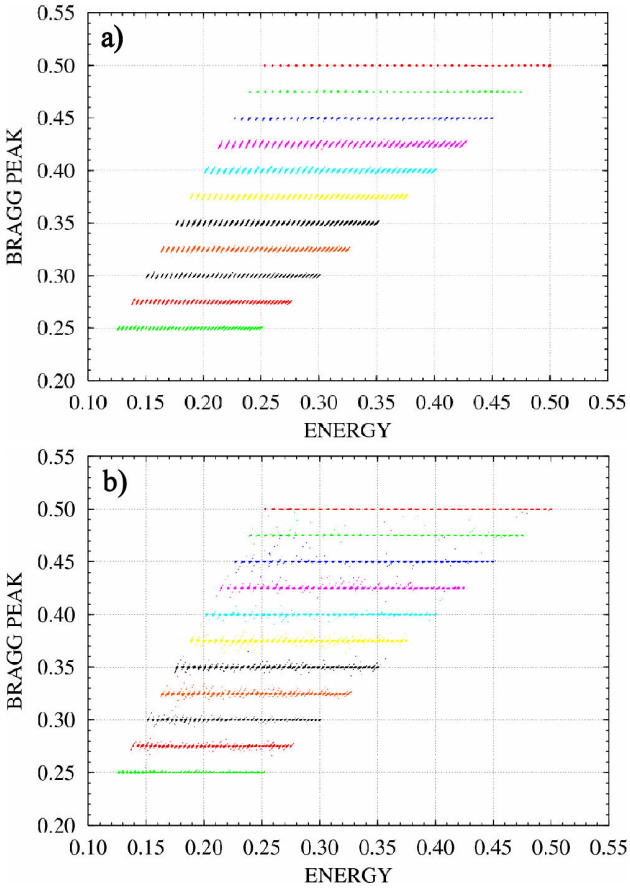

FIGURE 5. Scatter plot corresponding to 600,000 training epochs for a noise sizes equal to a) 2% and b) 6%. In figure a) can be seen that the ANN has been able of a good classification of all the groups (one could say a perfect classification), but it has not learnt all the groups with the same accuracy. In figure b) can be seen that the ANN has been able to learn all the groups reasonably well but, definitively, there are patterns with a bad identification.

similar accuracy; reaching a minimum error equal to 0.00287, which is much smaller than the minimum corresponding to the 6% error case, after 750,000 training epochs, and equal to 0.0435. In Fig. 3, for the three error sizes of interest, it is displayed the scatter plots corresponding to the minimum error values of the $E^V(\rho)$ error curves. In the 10% error case, this value was reached after 196,000 training epochs.

The effect of noise is to hide part of the information present in the patterns to be identified. During the learning process, the ANN has to build an architecture capable of representing, at least in principle, all the relevant information present in the patterns. Since the patterns are contaminated with noise, there is some point at which the remaining not so conspicuous relevant information starts being concealed by the noise. This concealing noise effect starts the sooner the larger the size of noise. In other words, in the 2% case in comparison to the 6% case, there is much retrievable relevant information that will take more epochs to the ANN to self adapt to it.

6. Conclusions

Perhaps the effect of the noise size on limiting the minimum value of $E^V(\rho)$is due to those patterns more severely affected by the noise component present in all patterns, which, consequently, are harder to classify correctly by the ANN. This cannot be solved by simply using a more complex ANN. It may happen, that any additional ANN architecture complexity be used to learn the noise present in the training patterns rather than in increasing the ANN generalization capability, that is what we ought to pursue. In principle, one way to enhance the generalization capability of the used ANNs without modifying their architecture is by increasing the training data set size. The problem with this option is that it would be very time consuming to train an AAN using a larger training data set. At this point, it is good to remind that noise learning implies a more complex model, and that, in turns, is related to larger values of the link array components. So, if one implements a way to penalize large link array components, the ANN generalization capability would improve by inhibiting noise learning. One way to implement this idea is achieved by modifying the cost functional $E^T(\rho)$ by adding a penalty or regularization term $\Omega o\rho p$ to it, *i.e.*:

$$\tilde{E}^T(\rho) = E^T(\rho) + \alpha\Omega(\rho), \tag{5}$$

where a simple form of the regularization term $\Omega o\rho p$ called weight decay is:

$$\Omega(\rho) = \frac{1}{2}\sum_{i=1}^{W}[w_i(\rho)]^2. \tag{6}$$

This term encourages a smoother network mapping, *i.e.*, reducing its effective complexity or the shape of the error surface landscape. In this expression W is the dimension of the weight array $\vec{w}$.The parameter α controls the extent to which the penalty Ω influences the form of the solution. In other words, the idea is to penalize the unnecessary ANN complexity. The use of this weight decay procedure [32, 33] for model selection is one popular way to achieve the regularization goal, and, in some way, its effect, is equivalent to that of the early stopping procedure that we followed in this paper, and it will be worth trying it, in the future, for BC identification using ANNs.

1. C.R. Gruhn *et al., Nucl. Instr. and Meth.* **196** (1982) 33.
2. A. Moroni *et al., Nucl. Instr. and Meth.* **225** (1984) 57.
3. M.F. Vinyard *et al., Nucl. Instr. and Meth.* **A255** (1987) 507.
4. K.E. Rhem and F.L. Wolfs, *Nucl. Instr. and Meth.* **A273** (1988) 262.
5. J.J. Vega, J.J. Kolata, W. Chung, D.J. Henderson and C.N. Davids, Proc. *XIV Symposium on Nuclear Physics, Cuernavaca, Mexico, 1991*, M. Brandan, (ed., World Scientific, Singapore, 1991) 221.
6. L. Andronenko *et al.*, Preprint PNPI NP-3-1998 Nr. 2217.
7. N.J. Shenhav and H. Stelzer, *Nucl. Instr. and Meth.* **228** (1985) 359.
8. M.N. Andronenko and W. Neubert, *Annual report 1998-1999* **FZR-271** (1999) 67.
9. J.J. Vega and R. Reynoso, *Nucl. Instr. and Meth.* **B243** (2006) 232.
10. C. M. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press-Oxford, (1995).
11. C. M. Bishop, *IEEE Transactions of Neural Networks* **4** (1993) 882.
12. C. Wang, S.S. Venkatesh and J.S. Judd, *NIPS'1993* Vol. 6, eds. J. Cowan, G. Tesauro, J. Alspector, Morgan-Kaufmann, (1994) 303.
13. W.S, Sarle, in Proc. of the 27^{th} Symposium on the Interface of Computing Science and Statistics, (1995) 352-360,
14. A.S. Weigend, M. Mangeas and A.N. Srivastava, *International Journal of Neural Systems* **6** (1995) 373.
15. C.M. Bishop, *Neural Computation* **7** (1995) 108.
16. S. Bös, *ICANN'1995* Vol. 2, ed. by EC2 & Cie, 111.
17. S. Bös, *NIPS'1995* Vol. 8, eds. G. D. Touretzky, M. Mozer, M. Hasselmo, MIT Press, (1996) 218.
18. P. Sollich and A. Krogh, in *Advances in Neural Information Processing Systems* Vol. 8, eds. D. S. Touretzky, M. C. Mozer and M. E. Hasselmo, MIT Press, (1996) 190.
19. S. Amari, N. Murata, K.-R. Finke, M. Finker and H. Yang, *IEEE Transaction on Neural Networks* **8** (1997) 985; and *NIPS'1995* Vol. 8, eds. G. D. Touretzky, M. Mozer, M. Hasselmo, MIT Press, (1996) 190.

20. S. Amari and N. Murata, *IWANN'1997*, eds. J. Mira, R.Moreno-Diaz, J. Cabestany, Springer, (1997) 284.

21. S. Lawrence and C. L. Giles, *IJCNN'00* Vol. 1, eds. Shun-Ichi Amari, C. Lee Giles, Marco Gori, and Vincenzo Piuri, IEEE Press, (2000) 114.

22. P. Domingos, *ICML'2000*, Morgan Kaufman, (2000) 223.

23. G. N. Karystinos and D.A. Pados, *IEEE Transactions on Neural Networks* **11** (2000) 1050.

24. R. Caruana, S. Lawrence and C.L. Giles, *NIPS'2000* Vol , 13, eds. T. Leen, T. Dietterich, V. Tresp, MIT Press, (2001) 28.

25. A. Zell *et al.*, *SNNS - Stuttgart Neural Network Simulator, version 4.2*, University of Stuttgart, Institute for Parallel and Distributed High Performance Systems; and University of Tübingen, Wilhem-Schickard-Institute for Computer Science, (1998).

26. B. Majorovits and H.V. Klapdor-Kleingrothaus, *Eur. Phys. J.* **A6** (1999) 463.

27. J.J. Vega, M.R. Reynoso, M. Arias E. and L. Altamirano R., Proc. IJCNN2000, *Neural Computing: New Challenges and Perspectives for the New Millennium*, Como, Italy, IV 2000, Shun-Ichi Amari, C. Lee Giles, Marco Gori, and Vincenzo Piuri, eds., IEEE Computer Society, Los Alamitos California, USA, (2000) 379.

28. J.J. Vega and M.R. Reynoso, *Proc. of The 7^{th} World Multiconference on Systemics, Cybernetics and Informatics*, Vol. XIV Computer Science Engineering Applications, ed. N. Callaos, W. Lesso, B. Sánchez and S. Li, July 27-30, Orlando, Florida, USA (2003) 396.

29. J. Damgov and L. Litov, *Nucl. Instr. and Meth.* **A482** (2002) 776.

30. M. Ambrosio *et al.*, (The MACRO Collaboration), *Nucl. Instr. and Meth.* **A492** (2002) 376.

31. E. Yoshida, K. Shizuma, S. Endo and T. Oka, *Nucl. Instr. and Meth.* **A484** (2002) 557.

32. C.M. Bishop, *Neural networks for pattern recognition*, Clarendon Press-Oxford, (1995).

33. J.E. Moody, in *Advances in Neural Information Processing Systems 4*, J.E. Moody, S.J. Hanson and R.P. Lippmann, eds., Morgan Kaufman, Palo Alto, (1992) 847.