Nuclear Instruments and Methods in Physics Research A 389 (1997) 381–387

NUCLEAR INSTRUMENTS & METHODS IN PHYSICS RESEARCH Section A

ELSEVIER

Neural computation methods and applications: Summary talk of the AI session

David Horn*

School of Physics and Astronomy, Tel Aviv University, Tel Aviv 69978, Israel

Abstract

The sixty papers selected for oral presentation in the AI session of the workshop were mostly application oriented, with high energy physics taking central stage. The most popular methodology was that of neural networks. In this summary talk I point out some general trends of the research work presented at the workshop, and concentrate on papers that I found especially interesting. I add sections that deal with recent developments in neural computation techniques that may be of relevance to applications in the future. The latter include PCA and ICA analyses, averaging over estimators and models of spiking neurons. Touching on some general questions, I indicate the advantages of the neural computation approach.

PACS: 07.05.M; 07.05.K 84.35; 10

Keywords: Neural networks; Applications to HEP; Algorithms; PCA; Temporal coding

1. Introduction: Summary of lectures

I will start this summary talk with a brief review of all 60 lectures in the AI session of the conference. I have grouped them according to the fields to which they were applied. I have devoted one line to each paper, trying to summarize briefly its message or, at least, its relevant keywords. The order in each table follows the original enumeration of the abstracts which appears in the first column.

Table 1 summarizes 37 lectures in the fields of High Energy Physics, Nuclear Physics, Astrophysics and Cosmic Rays. They are all relevant in one way or another to particle tracking and identification. Their vast majority uses neural network techniques, which is also true of the papers submitted in all other fields. A small fraction of the papers is devoted to hardware. Table 2 summarizes applications to geography and cartography. Most of these papers discuss the mapping of nuclear radiation distribution following the Chernobyl disaster. A variety of other applications is displayed in Table 3. Finally, in Table 4 we find theoretical developments, some of which were applied to synthetic data.

* Corresponding author: Tel.: + 972 3 6429305; fax: + 972 3 6407932; e-mail: horn@neuron.tau.ac.il.

In the following sections I will concentrate on some topics covered in the various lectures. I will start with hardware applications, and then turn to a discussion of different paradigms proposed by various authors. Some topics of data compression and scene analysis will be covered too. Following these brief summaries of topics that I found particularly interesting, I will devote three sections to new methods in neural computation that I expect to have an increasing influence on applications. I will conclude with a section of questions and answers on some central issues.

2. Hardware in HEP

One of the crucial tests of the acceptance of neural networks (NN) by the HEP community is the willingness of big collaborations to incorporate them in their trigger systems. Two of the papers presented here prove that this hurdle has been overcome. The L3 collaboration at the LEP collider at CERN has decided to incorporate a relatively simple neural network in its first level trigger next year [1], and the H1 HERA collaboration at DESY has started to implement a neural network in its second-level trigger last August [2].

The trigger system in HEP experiments filters a minority of interesting events (at a rate of few Hz) from a very

PII S0168-9002(97)00140-X

Table 1
HEP/NP/Astrophysics/Cosmic rays

003	NN fixed architecture, biologically motivated, for muon identification
012	NN elastic net for track and vertex search
014	NN BP for gamma ray analysis
016	CA vs elastic net in analysis of double beta decay
022	NN BP for extra galactic gamma radiation
023	CA data filtering for track recognition of charmed baryons
024	NN for particle identification in cosmic rays
027	NN and its feature inputs in 2nd level trigger ATLAS hardware
033	NN discriminators in ATLAS tile calorimeter separate glue versus b
034	NN BP for e/pion separation in H1
046	NN Kohonen map for pp reactions to distinguish π, γ, background
056	NN FF for 1st level trigger hardware
057	NN compare growing neural gas, FF-BP or GA, and other in LEP/aleph
060	NN Hardware 2nd level trigger using CNAPS for neutrino oscillation
066	NN Hardware TOTEM chip for the cosmic rays satellite PAMELA
070	NN BP with PCA inputs of invariant functions for e/π identification
074	NN Hardware TOTEM chip develop PCI VME boards
075	Mapping-deformable elastic hedgehogs for vertex finding
090_1	NN acceleration of training and functioning of pattern recognition
104	NN compare SA, threshold accepting, Tabu with Hopfield for tracks
105	search algorithm of branch & bound for track/vertex package
107	NN ART adaptive solutions on CNAPS for data mining in HEP
108_1	NN better than cuts in HEP/Astro
108_2	NN Bayesian formulation for cosmic rays
121	FL Hardware for particle identification
122	NN Hopfield and Elastic net for pattern recognition in H1
125_2	Biological vision elements for real time vertex/track id in HEP
126	NN Elastic arm track recognition
136	NN FF analysis of τ decays in LEP
139	NN dynamic perceptron redefinition of gain function
144	GA threshold acceptance for tuning monte carlo models in HEP
151	CA NN track/vertex recognition in HEP using ccd television
153_1	NN importance of high order correlations of inputs for particle identification
157	NN use for reconstructing faulty channels in shashlik calorimeter
160	NN FF analysis of six different τ decays at LEP
170	NN for tagging $Z \rightarrow b\bar{b}$ events in LEP.
171	NN Hardware realization of 2nd level trigger in H1 using CNAPS

Abbreviations: NN, neural networks; FF, feed forward; BP, backpropagation; GA, genetic algorithm; CA, cellular automata; SA, simulated annealing; FL, fuzzy logic.

Table 2
Geography

	NN interpolation of maps
	NN for Geographical information system
	NN for spatial interpolation
133	NN incremental algorithm, for BP with no bad minima
134_2	NN FF 2D interpolation of residual kriging
137	NN parameter elimination for cartography of lake sediments
164	SA NN GA are compared in geostatistical applications

Table 3
Other applications

071	GA magnetic measurements in rare earth compound
072	NN BP ultrasonic flaw classification (cracks porosity)
082	NN BP sample selection. Application: spectroscopic ellipsometry. SA
084	CA GA density classification applied to random number generation
103	NN feedback is important in biological vision to resolve conflicts
118	NN GA parametric learning: population of neuron activities
129_145	GA partial evaluation, graph drawing, ellipsometric measurement
129_2	NN image recognition ARTMAP compare with geometric invariant nets
129_6	GA NN 2D Gabor transform in image recognition
134_1	NN time series for radioactivity monitoring
140	data compression of satellite images. Hardware implementation

noisy input that can reach 10 MHz at the detector level. These numbers are the ones of the H1 system [2] where the hardwired logic of the first level trigger reduces the input to 5 kHz and the second level trigger puts the throughput at 200 Hz. This means that the second level

Table 4
General theory

025	CA for noise filtering in simulated data
035	NN for fractal function interpolation
106	NN Data mining for knowledge discovery
128	GA optimal solutions on graphs
132	NN FL forecasting time series. Synthetic examples

trigger has to act within 20 μs, a constraint that is met by the system [2]. Its neural network is implemented on a CNAPS VME-board. Its feed forward (FF) architecture is 64-64-1, and it performs its calculation within 8 μs. The speed was of utmost importance in this application, and this is what the NN solution could deliver.

In contradistinction to previous conferences, this time only two NN chips appeared in implementations. The most popular one seems to be CNAPS. The other is TOTEM, a product that stems from Physics oriented research. The new developments in this chip were discussed in one of the lectures [3]. A clever new trick is to perform multiplications by a plog function, that closely approximates the log function, yet can be implemented by simple binary operations. This allows for much better performance of the upgraded TOTEM++ with the high operating speed of 100 MHz. This processor is advertised as being able to handle a NN with 10 inputs and hundreds of neurons at a rate of 10 MHz.

Continuing with the theme of hardware, let me mention an interesting off-line application that comes from the DELPHI collaboration at CERN [4]. The problem tackled in this paper is one of dead channel recovery. The case at hand was that one of the channels in their STIC luminosity monitor stopped working. Since it could not be replaced for a lengthy period, until the next break in the operation of the accelerator, the information lost by the channel had to be reconstructed by other means. It turned out that a NN was the best solution. Using 48 channels surrounding the faulty one, and a FF architecture of 48-25-5-1, they have simulated an output that reproduced well the missing one. This was tested when the faulty channel was replaced, and served to correct the data taken previously with the faulty channel in place.

3. Different paradigms

Different NN paradigms, as well as other techniques, were presented in this workshop. Clearly the most popular is the FF architecture of a NN, usually trained with a backpropagation (BP) algorithm. An example is the DELPHI paper [5], that described a 17-18-1 network for the identification of $\tau^+\tau^-$ production by the Z^0. The 17 input variables contain 9 global variables, such as acollinearity and total EM energy, and 4 variables describing each of the two hemispheres. This network has a small advantage over conventional classification based on cuts in these variables. The choice of variables is a general problem, and I will touch upon it later when I get to a discussion of PCA and ICA.

Other types of NN are feedback or recurrent ones, that flow into fixed points representing encoded memories, or networks based on competitive learning that serve for clustering or feature mapping. An interesting example of the latter is the application of an elastic NN for vertex search [6], an important problem in particle and nuclear physics. A well-known example of the elastic ring is its application to the travelling salesman problem [7], where it is attracted to the various points representing cities, and it expands from some little structure until it fills the plane and sticks to the cities, representing an allowed path that forms the desired solution. Here, however, the ring implodes from some large structure to a point representing the vertex. This comes about through an attraction to the number of tracks, and the final vertex represents the largest density of tracks that the ring can encounter.

An elastic ring algorithm can be also used for track identification. A Dubna collaboration [8] has presented a cellular automation algorithm that serves as a first processing stage for track finding. Here the idea is to connect points (representing hits in the detector) with nearest and next to nearest neighbors. Assigning unit weights to all of them, one proceeds with weight updating by taking account of the weights that occur within some fixed angle to the left. Further applications of this step produce chains with increasing weights from left to right. At the end one identifies the highest weights occurring on the right, and proceeds to the left to collect all links belonging to a track. This was used in an application to a nuclear physics experiment.

Finally, I wish to comment on a paper that I have presented [9], whose special feature is a FF architecture with fixed synaptic weights. Here the main idea was to use orientation selective neurons, of the type known to exist in the visual cortex, for the discovery of straight tracks of cosmic muons in a segmented detector. This leads to specific connectivity of neurons on the hidden layers to the previous layer, implementing filters for predefined orientations within local windows on the input plane. Such a structure allows for easy hardware implementation.

4. Miscellaneous topics

Neural networks are usually employed to capture the general characteristics of the data sets on which they are trained. Hence, we often try to avoid fitting all fine details of a given set of data, i.e. overfitting, in order to allow for

good generalization. This turns out to be bothersome for the geographers who tried to use neural networks to interpolate a given set of data (a map) as accurately as possible. They ended up using a NN for describing the large scale features, and employing conventional geostatistical methods for the interpolation of the small scale data [10]. The application that most geographical papers focussed on was the spatial estimation of Chernobyl radioactive fallout. It turns out that once the large scale trends are taken into account by a NN structure, the residues, i.e. the differences between the data and the NN estimate, display short range correlations that allow indeed the employment of geostatistical methods.

A different paper [11] has also considered the possibility of using a NN for the representation of a specific function. Assuming that the function corresponds to a minimum of the network, the question exists how this minimum can be retrieved by a BP method, given the fact that it is a rare occurrence. The answer was sample selection. The basic idea is not to apply BP blindly to all errors, but select first the largest error, correct it, and proceed in steps of eliminating the remaining largest errors. This ensures zeroing in onto the desired minimum.

A different topic that I wish to mention is that of data compression. An interesting suggestion is to regard any given data set, e.g. a picture, in the same way one investigates a chaotic time series [12], since, after all, it can be represented by a chaotic string of bits. The representation of the latter as a series of many limit cycles, can be implemented in a dynamically adjusted wavelet transformation that serves as a basis for the new system. It competes favorably with conventional commercial algorithms.

Finally, let me point out another method of picture analysis, this time not from the point of view of data compression, but that of edge and texture detection. The novel suggestion [13] is to feed the picture as input to a two-dimensional pulse-coupled NN with short range interactions. I will give a simplified example of such a network below, in the section on spiking neurons. Their network flows into a periodic pattern that, for different points of time in each period, accentuates edges, or segments the picture into different components, or emphasizes different textures. Thus, it shifts the analysis into the temporal domain.

5. Choice of variables: PCA and ICA

One of the problems often faced in neural network applications is the choice of the right variables. Sometimes the number of candidate variables in huge, and one wonders which should be the right choice of inputs into a network that is supposed to perform some classification or decision task. At this point one may invoke the idea of PCA (principal component analysis) or the more general one of ICA (independent component analysis).

PCA can be simply described in a problem with n variables x_i, $i = 1, \ldots, n$, on which a multivariate Gaussian distribution is defined:

$$P(x_1, \ldots, x_n) = \frac{1}{(2\pi)^{n/2}\sqrt{\det(C)}} \exp(-\tfrac{1}{2} x_i C_{ij}^{-1} x_j), \quad (1)$$

where, for simplicity, we have assumed that the average is at the center $x_i = 0$. C is the covariance matrix

$$C_{ij} = \langle (x_i - \langle x_i \rangle)(x_j - \langle x_j \rangle) \rangle. \quad (2)$$

As is well known, this problem can be diagonalized by a similarity transformation:

$$W^{\mathrm{T}} C W = \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n). \quad (3)$$

The various eigenvalues λ_i represent the variance σ_i^2 of the corresponding eigenvector $y_i = \Sigma_j W_{ij} x_j$. Enumerating them in decreasing order, $\lambda_1 > \lambda_2 > \cdots > \lambda_n$, we obtain the set of eigenvectors y_i which are the principal components, ordered according to decreasing importance. Quite often the eigenvalues decrease rapidly. In that case, the first few principal components with the high eigenvalues, are the most relevant variables. They may often suffice for a good description of the data.

The intuitive feeling that one has to give a more precise definition to a variable whose variance is high, whereas one can use the average value if the variance is very low, can be cast into a statement about information content. For a stochastic variable with probability distribution $P(x)$ one may interpret the quantity $-\log P(x)$ as the amount of information required to specify that the variable X acquires the value x. Its average is the entropy

$$H(X) = -\sum P(x) \log P(x). \quad (4)$$

For the Gaussian distribution this is equal to

$$H(X) = \tfrac{1}{2}\log[(2\pi e)^n \det(C)] = \frac{n}{2}\log[2\pi e] + \sum_i \log(\sigma_i).$$

For any general probability distribution this equation turns into an upper bound (see, e.g., the recent review [14]). By going into the PCA representation $y_i = W_{ij} x_j$, we transformed H into a sum over independent variables, due to the factorization $P(y) = \prod_i P(y_i)$. In the principal component approximation we retain the eigenvectors with the highest information content.

The PCA is important in problems where we have a large number of variables. Accomodating all these variables in a neural network, requires large training time and a large training set. By limiting ourselves to just a few variables in the input, we can construct a moderate

network that can hopefully lead to good generalization. We wish however that this data compression procedure leaves us with, as much as possible, the same information as in the original data. In terms of information theory, this is defined as the maximization of mutual information

$$I(X, Y) = \sum_x \sum_y P(x, y) \log \frac{P(x, y)}{P(x)P(y)} \tag{6}$$

between the two sets of variables x and y. This quantity is non-negative, and vanishes only when the two variables are independent of one another. Its maximal value is obtained for $Y = X$, when $I(X, Y) = H(X)$. This condition is obeyed for the PCA. In general, however, we have to consider a situation where the probability distribution is non-Gaussian and, in particular, the relevant correlations are higher than second order. Trying to solve the maximization condition as best as possible defines the independent component analysis (ICA).

The PCA solution, $y_i = W_{ij}x_j$, fits a simple linear neural network structure. One can therefore use neural network algorithms [15,16] to search for it. The implementation of the ICA involves much more than the diagonalization of the covariance matrix. One has to make sure that all higher correlation coefficients are minimized [14], which is quite complicated in general. This becomes easier if one limits oneself to a neural network structure, with a nonlinear sigmoid transfer function, and imposes on it the maximization of mutual information. This approach was recently implemented in studies of noisy networks [17,18] and in applications to blind separation and blind deconvolution of time series [19].

I started out with the question of which variables have to be used in an analysis and ended up with a neural network after all. This may lead to the question why not start with a large NN to begin with, and let it do the work of both finding the best variables and proceeding with their analysis. The answer is that it is worthwhile to think of the problem in stages in order to have better understanding and better control of the various stages of the calculation.

6. Averaging over predictors

Many of the papers in the conference ended up showing results of some NN that was selected by their algorithms. I wish to point out here that there exists a simple way that will, in general, assure better performance by simply averaging over an ensemble of networks [20].

Each FF NN can be interpreted as the predictor $f(x)$ of some function $y(x)$ which we do not know in a closed form. The predictor is trained on some training set, and is then tested on some test set with a measure $d\mu(x)$. Its mean square error is

$$\mathrm{MSE} = \int d\mu(x)(y(x) - f(x))^2. \tag{7}$$

In general, x is a vector in some large space of variables, and y may have stochastic variations, e.g. through noise. In the latter case it suffices to replace it by its average value, because we cannot expect the predictor to do better than that. One can define [21] in a natural way a division of this error into bias and variance components. This involves considering an average MSE over the set of all possible data as well as all possible architectures and training parameters.

For the purpose of our discussion consider the simple case in which the architecture of the NN is fixed. There still exists the arbitrary choice of initial conditions of all weights. A characteristic network will then have an error of

$$\langle \mathrm{MSE} \rangle = B + V, \tag{8}$$

where the averaging procedure is carried out over all possible choice of initial conditions [22]. Its division into bias and variance is defined as follows:

$$B = \int d\mu(x)(y(x) - \langle f(x) \rangle)^2, \tag{9}$$

$$V = \int d\mu(x) \langle (f(x) - \langle f(x) \rangle)^2 \rangle. \tag{10}$$

Both terms are positive definite. The first is characteristic of the architecture of the network, and the data on which it was trained, but is independent of the particular choice of initial conditions. The second describes the variance in the error expected because of these initial conditions.

Consider now the simple possibility of using as a predictor not just any f but its average over initial conditions $\langle f \rangle$. This way we are left with a reduced error, presented by the bias term only. This is the major idea behind averaging over predictors. You try to minimize your dependence on factors that you can handle. Although, by some statistical fluke, a single network can do better than the average on a given set of data, you learn from this analysis that, on the average, you are better off by considering the predictor to be the average over networks.

7. Spiking neurons

Biological neurons are communicating with one another through their action potentials, which are voltage spikes generated by neurons when their membrane potentials reach some threshold. Most artificial neural networks use variables that may be interpreted as spiking rates of natural neurons. It is generally believed that the spiking rate plays a major role in the neural code.

However, this may well be only part of the general picture. There is some evidence that the exact timing of the spike may carry some information, and the synchrony between spikings of different neurons may be meaningful.

A simple representation of spiking neurons is given by the integrate-and-fire model, in which the membrane potential of the neuron obeys the equation

$$C\frac{dV}{dt} = -\frac{V - V_0}{R} + I, \tag{11}$$

where V_0 is a resting potential and I represents the incoming current. If, at some time $t = t_1$, V reaches its threshold value θ, a spike is emitted, $S \to S + s(t - t_1)$, and V is reset to V_0 within a short refractory period. S represents the action potential, and s is a sharp function, that can be approximated by a δ function. This can be easily made into a network, by connecting the input of neuron number i to the outputs of all other neurons:

$$I_i = I_i^{\text{ext}} + \sum_{j \neq i} W_{ij} S_j. \tag{12}$$

One may of course introduce also time delays into the system, as well as temporal structure into the synaptic coupling. Neurons may be divided into excitatory and inhibitory ones, and feed forward or feedback systems can be constructed. Such networks are sometimes employed in applications, as already mentioned in Section 4, but their mathematical properties are not yet as well understood as standard artificial neural networks.

In a recent paper, Hopfield [23] has suggested that a scheme based on the comparison of the spiking time of neurons versus some internal periodic structure may have an important computational advantage, allowing the system to perform analog matching. This concept refers to the recognition of patterns based on the ratio between individual components rather than on their absolute values, and plays an important role in the recognition of visual patterns as well as sound and odor. In his scheme there is a subthreshold oscillatory potential added to the external input I^{ext}. As a result, the neuron will fire periodically with some time lag τ_i with respect to the underlying oscillation. For some range of strengths of the inputs, $\tau_i \approx \log(I_i^{\text{ext}})$. Thus, differences of time lags are functions of ratios of input amplitudes, leading to the proposed basis for analog matching.

The idea that the precise timing of the neuron's spike is of computational importance was adopted by Maass [24], who investigates the computational power of networks of spiking neurons. In his model of feed forward networks, the synaptic input $W_{ij}S_j$ from neuron j to neuron i is considered to be effectively constant from the time that neuron j fired until the time neuron i fires. Thus the earlier the firing of j the stronger its influence becomes, leading to a temporal representation of analog amplitudes. His conclusion is that such networks can perform any function that regular artificial networks can do, and more.

The schemes sketched above need some basic clocking device to turn the timing of the neuron's spike into a computational variable. This may seem as a natural candidate for future hardware applications, and it remains to be seen how useful spiking networks will become.

8. Frequently asked questions

Instead of having a conventional summary let me try to answer some FAQs, using the internet lingo. These questions are often raised in conversations, especially when trying to convince your fellow researchers that neural networks or genetic algorithms are methods one wishes to employ.

- Why follow heuristic approaches such as neural networks or genetic algorithms at all? Can't we do better by just solving the problem, the way Physics has advanced in the last few centuries? After all, this amounts to real understanding of the issues involved.

Clearly, nothing is better than the real solution. But when complex questions arise, simple solutions are difficult to come by. Complexity is the key issue here. It may arise from the large number of variables, or from complicated dynamical relations, or just from the structure of data as in very complex detector systems. Our tools allow us to attack such questions in the absence of any prior understanding. In contradistinction to traditional statistical methods, we have nonlinear functions at our disposal. If some clue is known we can easily incorporate it into our system and improve it. In fact, we better do it, rather than expect the network to learn what we already know. If and when an alternative, that incorporates physical or statistical understanding, is found, we should not hesitate to use it. An example is the PCA, discussed above in Section 5.

- Given the heuristic technique that is inspired by biology, how can we be sure that it will lead to the best results?

Well, this question is somewhat ill defined. It really depends on what it is that one wishes to maximize (or minimize) when one talks about an optimal solution. This depends on available hardware and software. In issues that are relevant to biological survival, such as pattern recognition, the engineering of the biological hardware, and the machinery for its analysis, fill us with awe. For some of these issues it has been argued that biological solutions do correspond to optimal solutions from the point of view of information theory [25,26].

- How do we know that, with a given method, we will be able to generate the global minimum of its energy (or error) function?

We do not usually. However, a low local minimum may suffice for all practical purposes. Sometimes a low

minimum in a FF net may lead to bad generalization. We have seen, in Section 6, that using an ensemble of networks guarantees reduction of the error.

– How trustworthy are neural networks?

Neural networks have very simple algorithms, and that is their strength. The difficulty is to prove that the algorithm adapts correctly to the problem. Confidence measures of neural networks can be defined for randomly generated data, measuring the capacity of an associative neural network, or the generalizability of a feedforward net. These can be used to estimate the capability of a given net.

Since the algorithms are simple, their implementation is straightforward and trustworthy. Compare that with standard computer algorithms, when there is no uncertainty in the validity of an algorithm but its implementation in a program may be so complex, as to allow only a phenomenological proof of its usefulness: it works on numerous examples. And if it fails, the bug has to be traced down and corrected. I guess the same should hold for any complex system that is applied to data collection and analysis, the subject of most papers in this session of our conference.

Acknowledgements

I thank Halina Abramowicz and Eytan Ruppin for many helpful suggestions. This work was partly supported by the Israel Science Foundation.

References

[1] A. Vlachos, A first level trigger using specially developed neural network hardware, Nucl. Instr. and Meth. A, these proceedings.
[2] J. Fent et al., The realization of a second level neural network trigger for the H1 expreiment at HERA, Nucl. Instr. and Meth. A, these proceedings.
[3] P. Lee, A. Sartori, G. Tecchiolli, I. Lazzizzera and A. Zorat, Advances in the design of the TOTEM neurochip, Nucl. Instr. and Meth. A, these proceedings.
[4] M. Bonesini, P. Ferrari, S. Gumenyuk, M. Paganoni, L. Petrovyck and F. Terranova, Application of neural nets to Shashlik calorimetry, Nucl. Instr. and Meth. A, these proceedings.
[5] F. Grotti, F.R. Cavallo and F.L. Navarria, Use of neural networks in the analysis of $\tau \to K_s^0$ inclusive decays at LEP, Nucl. Instr. and Meth. A, these proceedings.
[6] K. Ivan, Elastic Neural net for track and vertex search, Nucl. Instr. and Meth. A, these proceedings.
[7] R. Durbin and D. Willshaw, Nature 326 (1987) 689.
[8] M.P. Bussa, V.V. Ivanov, I.V. Kisel and G.B. Pontecorvo, Application of cellular automata and neural networks for track recognition in experiments DISTO and STREAMER, Nucl. Instr. and Meth. A, these proceedings.
[9] H. Abramowicz, D. Horn, U. Naftaly and C. Sahar-Pikielny, Orientation selective neural network for cosmic muon identification, Nucl. Instr. and Meth. A, these proceedings.
[10] M. Maignan, M. Kanevski, M.F. Maignan and V. Demianov, How neural network 2-D interpolations can improve spatial data analysis – Neural network residual kriging (NNRK), Nucl. Instr. and Meth. A, these proceedings.
[11] L. Redei and H. Wallinga, A novel modification to backpropagation sample selection strategy, Nucl. Instr. and Meth. A, these proceedings.
[12] A.L. Perrone, Applications of chaos to lossless and lossy satellite image compression, Nucl. Instr. and Meth. A, these proceedings.
[13] T. Lindblad and C.S. Lindsey, Intelligent detector systems modelled from the cat's eye, Nucl. Instr. and Meth. A, these proceedings.
[14] G. Deco and D. Obradovic, An Information theoretic approach to neural computing (Springer, 1996).
[15] E. Oja, Int. J. Neur. Syst. 1 (1989) 61.
[16] T.D. Sanger, Neural Networks 2 (1989) 459.
[17] J.-P. Nadal and N. Parga, Network 5 (1994) 565.
[18] A. Campa, P. Del Giudice, N. Parga and J.-P. Nadal, Maximization of mutual information in a linear noisy network: a detailed study, preprint, 1995.
[19] A.J. Bell and T.J. Sejnowski, Neural Comput. 7 (1995) 1004.
[20] D.H. Wolpert, Neural Networks 5 (1992) 241.
[21] S. Geman, E. Bienenstock and R. Doursat, Neural Comput. 4 (1992) 1.
[22] U. Naftaly, N. Intrator and D. Horn, Optimal ensemble averaging of neural networks, preprint, 1996.
[23] J.J. Hopfield, Pattern recognition computation using action potential timing for stimulus representation, Nature 376 (1995) 33.
[24] W. Maass, Neural Comput. 8 (1996) 1; Neural Comput. 9 (1997) 279.
[25] R. Linsker, IEEE Computer 21 (1988) 105.
[26] H. Barlow, Neural Comput. 1 (1989) 295.