ELSEVIER

Microelectronics Journal 30 (1999) 77–82

Microelectronics Journal

Neural network quantifier for solving the mixture problem and its implementation by systolic arrays

R.M. Pérez*, P. Bachiller, P. Martínez, P.L. Aguilar

Departamento de Informática, Escuela Politécnica, Universidad de Extremadura, Avenida de la Universidad s/n, 10071 Cáceres, Spain

Accepted 18 September 1998

Abstract

In this paper we present the application of a method, based on the orthogonal transformation, to develop an optimal neural network for solving the Mixture Problem and a linear systolic to design it is provided. We use a back-propagation neural model for determining and quantifying the components in a composite spectrum obtained from a given mixture of elements. The spectra of the possible components are used as the training patterns. The orthogonal transformation used in the present work are the singular value decomposition (SVD) and the QR with column pivoting factorization (QRcp). An interesting property of the proposed method is related to the possibility of reducing the input and hidden nodes at least to the number class. This reduction allows us to obtain an optimum VLSI implementation by a linear systolic.

Keywords: Neural networks; Orthogonal transformation; Systolic arrays; Signal processing; Pattern recognition

1. Introduction

Nowadays in most of signal processing applications (colorimetry, remote sensing, quality control and many others) it is necessary to determine and quantify the components in a composite spectrum obtained from a given mixture of elements. This problem is known as the *Mixture Problem*.

The Mixture Problem may be described formally as follows: Assuming we are given the spectra of a number (K) of elements (*basic references*), we must determine the unknown composition of a *cocktail* of the mentioned elements using a radiation spectrum obtained from this mixture.

To date, some methods have been proposed to solve the Mixture Problem. Among several approaches, the conventional digital algorithm [1], the optical neural network [2], the multiple regression algorithm [3] and the HRNN [4] have been used as powerful tools to find the components in a composite spectrum and its contributions to the mixture. In this work we propose the utilization of a back-propagation (BP) model to solve this problem.

Conventionally a symmetric and homogeneous structure is used for feedforward neural network (NN) models. To be representative, the network should have an optimum number of links, and it does not need to be homogeneous. The network will be overparametrized if the number of links is very high. In such cases, if the training set of data is not noise-free, the NN will try to learn the information along with the noise in the data, leading to poor validation results. Two related points are important here:

1. the NN should have optimum number of inputs; and
2. the NN should have an optimum number of links.

It is well known that collinearity between the inputs can lead to identification problems.

A robust and direct approach for the optimization of the size of any feedforward network is the use of orthogonal transformations. Two types of orthogonal transformations are used, namely the singular value decomposition (SVD) and QR with column pivoting (QRcp) factorization.

SVD is mainly used for null space detection; QRcp coupled with SVD is used for subset selection, which is the key of the design of optimal networks [5].

The BP model is extremely demanding in both computation and storage requirements. An enormous amount of computation has to be spent on training the network. In the retrieving phase, extremely high throughputs are required for real-time recognition. The attractiveness of the digital approach to real-time processing hinges upon its massively parallel processing capability.

The BP algorithm is computationally iterative and intensive, and it demands very high throughput. Multiprocessors,

* Corresponding author. Tel.: 0034-27-257-183; Fax: 0034-27-257-203; e-mail: rosapere@unex.es

0026-2692/99/$ - see front matter
PII: S0026-2692(98)00103-7

array processors and massively parallel processor provide a natural solution. The BP algorithm can be expressed in the basic matrix operations, such as inner-product, outer-product and matrix multiplications, which, in turn, can be mapped to basic processor arrays, systolic or wavefront arrays, for instance. They have the following key advantages:

1. The exploitation of pipelining is very natural in regular and locally connected networks. They yield high throughput and simultaneously save the cost associated with communication.
2. They provide a good balance between computation and communication, which is critical to the effectiveness of array computing.

The rest of the paper is organized as follows: First, in Section 2, the application of orthogonal transformations to reduce the NN size is presented; next, in Section 3, we make a description of the BP solving the Mixture Problem. The results obtained are presented and discussed in Section 4 and finally, in Section 5, the details of the proposed systolic array are related.

2. Application of orthogonal transformations for reducing the neural network size

Orthogonal transformations can lead to relative decorrelation and compaction of information in a data set, which has several applications. There have been some efforts for using orthogonal transformations using these transformations in NN modelling. For example, Karhunen–Loève transformation (KLT) has been used in image processing context [6]; principal components analysis (PCA) has been used for pruning of NNs [7,8]. KLT and PCA are both eigenvalue based transformations, whereas SVD is singular value based; it is know that the singular values can be computed more efficiently with much greater numerical and computational stability than the eigenvalues, and hence SVD based methods are superior.

The basic principle of NN size reduction is to detect collinearity between the candidate information sets at various stages of the network and to eliminate the same. The use of SVD can cure the numerical ill-conditioning problems without reduction of the parameter set; but subset selection and the elimination of the redundant set can solve numerical problems with consequent reduction in the network size.

Two basic problems are addressed:

1. which of the candidate inputs to the NNs constitute the optimal set of inputs and
2. at the hidden layer(s) which links between the post-hidden layer stage and the subsequent stage should be retained for representative modelling.

The SVD of an $N \times M$ matrix $\mathbf{R}$ is given by [9]:

$$\boldsymbol{R} = \mathbf{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}} \tag{1}$$

where $N \times N$ matrix $\mathbf{U}$ and $M \times M$ matrix $\mathbf{V}$ are orthogonal matrices; and $\boldsymbol{\Sigma}$ is an $N \times M$ matrix given by

$$\boldsymbol{\Sigma} = \left[\mathrm{diag}\left\{\sigma_1, \ldots, \sigma_p\right\} : 0\right] \qquad \sigma_1 \geq \sigma_2 \geq \ \ldots \geq \sigma_p \tag{2}$$

where $p = \min(N, M)$ and $\sigma_1, \sigma_2, \ldots, \sigma_p$, are the singular values. For a small value ϵ, we can define the rank of the matrix $\boldsymbol{R}$, as:

$$\mathrm{rank}(\boldsymbol{R}, \varepsilon) = \min_{R - B_2 < \varepsilon} (\mathrm{rank}(\boldsymbol{B})) \tag{3}$$

It can be shown that if $q_\epsilon = \mathrm{rank}(\boldsymbol{R},\ \epsilon)$ then

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{q\varepsilon} > \sigma_{q\varepsilon - i} \geq \sigma_p.$$

Thus, using SVD, the rank of a given matrix can be computed.

Contrarily, the QRcp factorization of the matrix is given by:

$$\boldsymbol{Q}^T \mathrm{R}\Pi = \begin{bmatrix} S_{11} & S_{12} \\ 0 & 0 \end{bmatrix}_{N-q}^{N} \tag{4}$$

$$qM - q$$

where $q = \mathrm{rank}(\boldsymbol{R})$, matrix $\mathbf{Q}$ is orthogonal and matrix $\boldsymbol{S}_{11}$ is an upper triangular and singular matrix.

The QRcp provides a basis for:

$$r_n(R) = \left\{y \in \mathscr{R}^N; y = \boldsymbol{R}x \qquad \text{for } x \in \mathscr{R}^{\mathrm{M}}\right\}. \tag{5}$$

The matrix Π provides the r columns in $\boldsymbol{A}$, which compose such basis.

The application of the above orthogonal transformation depends on which nodes are going to be optimized.

Let $N \times M$ matrix $\mathbf{R}$ comprise the input data set (reference set), where there are N input nodes and M classes

Optimum number of input nodes

The objective is to determine which of the N variables (or features) are relatively redundant and hence can be eliminated. Using SVD the optimum number of input nodes (say l) of a NN for an input data set is determined. QRcp provides l of the N features for the M sets of data points, which are enough for correct training and recognition.

Optimum number of hidden links and nodes

Consider a network which has been trained with M input data sets (*reference set*). An $\mathrm{M} \times q$ matrix $\mathbf{R}^*$ is formed with the q-pseudo outputs for the concerned hidden layer for each of the M input data sets. SVD is performed on $\boldsymbol{R}^*$ for determining the number of hidden nodes which are enough for the given network. Once remaining nodes have been eliminated the reduced-size network is retrained.

Table 1
Mixture spectrum (30% Mn and 70% Co)

Pattern	Contribution		
	Case (a) $1024 \times 10 \times 4$	Case (b) $4 \times 10 \times 4$	Case (c) $4 \times 4 \times 4$
Co 70%	0.699999	0.7	0.7
Mn 30%	0.3	0.3	0.299999
Ba	0	0	0
Cs	0	0	0
Iteration number	170	95	90

3. BPNN for solving the mixture problem

In order to describe the neural model suggested in the present work, it must be taken into account that a composite spectrum $\boldsymbol{x}$ may be seen as an *N-dimensional* vector, built sorting the emission intensities associated to each energy channel vs. the channel number, where N is the total number of energy channels:

$$\boldsymbol{x} = [x_0, x_1, \ldots, x_{N-1}]^{\mathrm{T}} \qquad x_n \geq 0 \;\; 0 \leq n \leq N-1 \tag{6}$$

x_n being the intensity measured as the number of photons whose energy is comprised in the nth energy channel's interval.

Thus, a *reference spectrum* is a spectrum of the same nature, but produced by an *individual source*. We denote these spectra as r_k with $0 \leq k \leq K-1$. The set of K reference vectors is called the *reference set*, and it must be evaluated in advance. For the sake of compactness, it is denoted as a *reference matrix* $\boldsymbol{R}$ composed by the reference column vectors:

$$\boldsymbol{R} = [r_0, r_1, \ldots, r_{K-1}]. \tag{7}$$

In general sense, the set of composite spectra is the set of all possible spectra that may be produced by a linear combination of all elements belonging to the *reference set*. When the *reference set* is composed by K linearly independent vectors, this would result in a *K-dimensional vector space*, integrated by all the vectors $\boldsymbol{y}$ given by:

$$\boldsymbol{y} = \mathbf{Rc} = \sum_{i=0}^{K-1} c_i r_i \tag{8}$$

where c is the *contribution vector* defined as:

$$c = [c_0, c_1, \ldots, c_{\mathrm{K}-1}]^{\mathrm{T}} \qquad c_k \geq 0 \;\; 0 \leq k \leq K-1. \tag{9}$$

Every contribution c_i is a function of the relative intensities of the composite and reference spectra.

Our goal is to estimate $\mathbf{c}$ assuming that $\boldsymbol{R}$ and $\boldsymbol{y}$ are known, for this we consider an approach based on a feed-forward NN which uses the BP learning algorithm. The neural model has a single hidden layer and it is formed with N linear input nodes and K linear hidden and output nodes, where N is the number of energy channels of the training spectra and K is the number of such spectra.

It is interesting to note that at most only K hidden nodes are needed for the correct learning of the network. It is due to the fact that SVD gives the optimum number of hidden nodes as the rank of a $K \times M$ matrix, K being the number of the learning patterns and M the number of hidden nodes considered first. Hence, if we take M greater than K, it can be always reduced to K. The same is true for input nodes so N can be reduced to K.

We consider a linear function as activation function and we take for each training spectrum i, the desired output vector as $[0_0 \ldots 1_i \ldots 0_{K-i}]^{\mathrm{T}}$.

Once the network has been trained, the outputs for a spectrum x are given by:

$$o_i = \sum_{j=0}^{K-1} w_{ij}^{o} h_j \qquad 0 \leq i \leq K-1 \tag{10}$$

where

$$h_j = \sum_{l=0}^{N-1} w_{jl}^{h} x_j \qquad 0 \leq j \leq K-1 \tag{11}$$

x_j is the jth energy channel of the spectrum x, w_{jl}^{h} is the connection weight from the lth input node to the jth hidden node and w_{ij}^{o} is the connection weight from the jth hidden node to the ith output node.

$x_1, \ldots, x_k$ are the training spectra. If we consider a mixture spectrum $\boldsymbol{y}$ given by:

$$\boldsymbol{y} = \sum_{\mathrm{m}=1}^{\mathrm{k}} \mathrm{c_m x_m} \tag{12}$$

where c_m is the mth training pattern contribution, then the outputs of the network described above will provide the c_ms values. It can be shown as follows:

$$h_j = \sum_{l=0}^{N-1} w_{jl}^{h} Y_j = \sum_{m=1}^{K} c_m \sum_{l=0}^{N-1} w_{jl}^{h} x_{mj}, \tag{13}$$

$$o_i = \sum_{m=1}^{K} c_m \sum_{j=0}^{K-1} \sum_{l=0}^{N-1} w_{ij}^{o} w_{jl}^{h} x_{mj} = \sum_{m=1}^{K} c_m o_{im} \tag{14}$$

where o_{im} and x_{mj} are the ith output and the jth energy channel for the spectrum, respectively. But the ith output is 1 for x_i and 0 for each other, so we can conclude that

$$o_i = c_i. \tag{15}$$

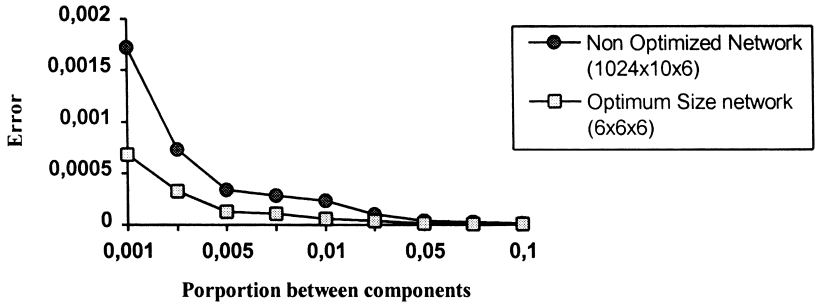

Fig. 1. Influence of the proportion between components.

Once the network was trained, it can be optimized using the methods SVD and QRcp (explained in Section 2).

4. Experimental results

The method stated above was widely simulated using a set of 1024 dimensional composite spectra obtained from individual sources. In order to evaluate it, two sets of experiments were performed. Through the first one, we have compared the results obtained using different number of hidden nodes and input nodes. Table 1 shows the results obtained for a mixture spectrum (which was composed of 30% Mn and 70% Co), using Mn, Ba, Co and Cs as training patterns. For the case (a) the total channel number (1024) was used; after applying orthogonal transformations to reduce the input and hidden nodes, results shown in columns (b) and (c) were obtained for other channel numbers. In all the related experiments similar results were found.

An important feature of the explained decomposition method is its capacity for recognizing very small ratios of the components that form a mixture spectrum.

We have carried out experiments using spectra of mixture of two individual sources in different ratios. Fig. 1 shows the degradation of the precision as a function of the relative proportions between the components of composite spectra of two elements. In spite of such degradation, results can be considered good, even with elements in the ratio of 1:1000.

In Fig. 1 the error was calculated as:

$$\text{Error} = \sqrt{\sum_i (c_i - c_i^*)^2} \tag{16}$$

where c_i and c_i^* are the known and the obtained contribution of the ith component, respectively.

In the remainder sets of experiments, we were dedicated to studying the influence of the different parameters used in the BPNN model.

5. Linear systolic arrays for the optimized network

The systolic designs exploits the potential of parallel processing offered by VLSI/ULSI technologies. A very desirable architecture is based on pipelined processing on primarily locally interconnected processor elements. To achieve such a design, dependence-graph-based mapping methodology is the most effective tool for neural information-processing applications [10].

In a general sense, the different phases in the BP processing, can be implemented in terms of consecutive matrix–vector multiplications, interleaved with two strips of non-linear processing units. In our case the non-linear units are not necessary, because we use nodes with linear activation function. The neural model proposed is depicted in Fig. 2. According to the discussions presented in the previous sections, we have considered K input, hidden and output nodes.

Using a systematic mapping methodology for deriving systolic arrays [10], this model can be mapped into a systolic array with K process elements (PEs). In order to describe the data movements and the PEs design requirements we distinguish between the retrieving and the training phases.

5.1. Retrieving phase

In this phase an input pattern is presented to the network, and the outputs are calculated, evaluating firstly the hidden nodes and using them for evaluating the network outputs.

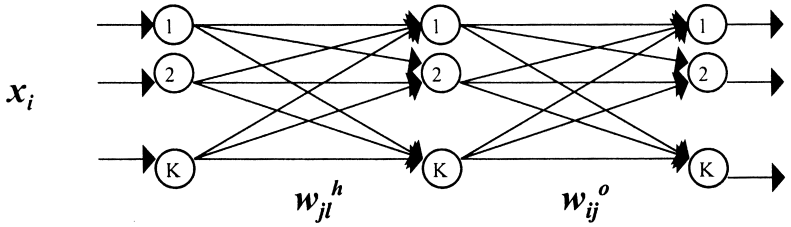

Fig. 2. Optimum back-propagation NN for solving the mixture problem.

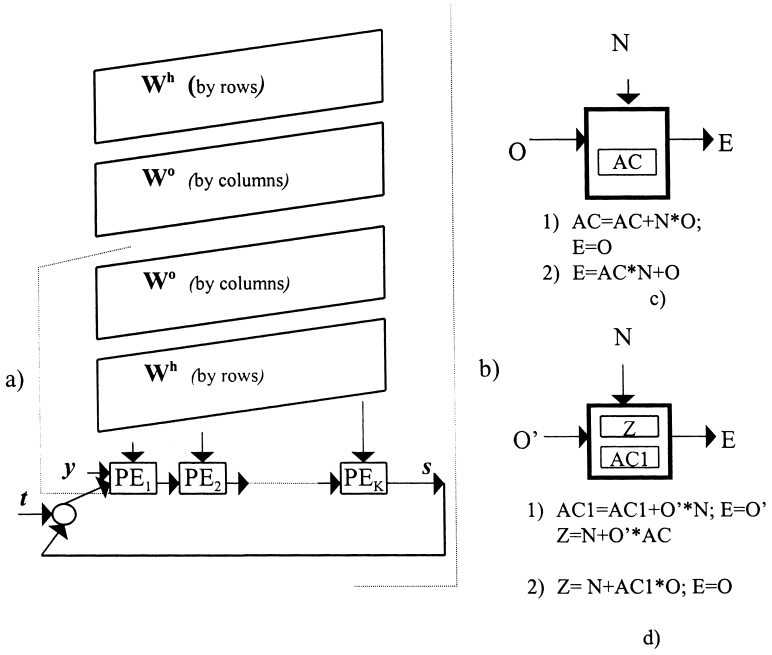

Fig. 3. Systolic for an optimum BPNN for solving the mixture problem diagram; we depict the elements used during retrieving phase (a) and learning phase (b). We describe briefly the PEs for the different learning steps, (c) during the feedforward phase and (d) the error feedback propagation and weight modification calculus.

We consider the net separated as the lower-layer net and the upper-layer net.

5.1.1. Data movements in linear systolic design

The data movements comprise:

1. The lower-layer net
   - The data y_i are input sequentially into the first PE in the original natural order. They are subsequently propagated downward to all other PEs.
   - The weight data w_{ji}^h are stored (row by row) in the PEs.
   - When the value y_l arrives at the jth PE (from the top), it is multiplied with the stored data w_{jl}^h to yield a partial sum h_l to be accumulated in the same PE.
   - The PE changes mode (by the control unit) to be ready for the next (upper-layer) phase.
2. The upper-layer net
   - The data h_l stays in the jth PE and need not be propagated to other PEs. The weight data of this layer are stored (column by column) in the PEs.
   - Multiply the values h_j and w_{jl}^o, and add the product to the partial sum received from the upper PE. The new partial sum is then propagated to the lower PE.
   - As soon as the final partial sum is computed at the last PE, the activation value can be produced s_i.

5.1.2. Processor element design requirements

The processor elements comprise the following key components:

Memory

Each PE should store a row of the lower-layer weight matrix and a column of the upper-layer weight matrix.

Communication

Data are transmitted in one direction between two neighbouring PEs.

Arithmetic processing

Each PE should support all the arithmetic processing capabilities including the multiply-and-accumulate operations.

5.2. Learning phase

5.2.1. Data movements

In this phase we distinguish three steps:

1. Forward step: The forward step in the learning phase follows the same data movements as the retrieving operations described in Section 4. However, it is not necessary to store the input values x in the first PE so that they can be reused in a latter step.
2. Backward step for the upper-layer net: In this step, both the BP algorithm and the training are executed:
   - Value s_i is routed by the feedback link to a special processor above the first PE. At the special processor, the values $(t_k - s_k)$ are produced.
   - After the error value is produced, it is propagated downward to all the other Pes.
   - Data (w_{ij}^o) and h_i are stored in the ith PE.
   - When the error value arrives at the ith PE:

It is multiplied with the stored data w_{ij}^{o}, and the product is added to the partial sum being accumulated in the same PE. (After K active clocks, all the final partial sums are generated.)
It is also multiplied by value h_i to yield the weight increments, which is the used to update the new weight at the ith PE.

- The back-propagated error functions can be derived.

3. Training step for the lower-layer weights:

- Recall that value $\boldsymbol{x}$ was stored in the first PE during the forward step; now it can be propagated again downward to all the others PEs.
- When x_j arrives at the ith PE, it is multiplied with the error value to yield the weight increments, which is in turn used to update the new weight in the ith PE.

5.2.2. Process elements design requirements

The process elements comprised basically the same components as are required for the retrieving phase. The PEs should be programmable due to the varying functionalities involved. Some others differences are discussed as follows:

Memory
This is the same as the retrieving phase. Each PE should store a row of the lower-layer weights and a column of the upper-layer weights. In addition, in the first PE, an extra FIFO is required to recycle the $\boldsymbol{x}$ data for processing.

Communication
It is just like in the retrieving phase. Data transmitted unidirectionally between neighbouring PEs, and circular link between the first PE and the last PE is added to facilitate pipelining.

Arithmetic processing
This includes multiplications, additions and accumulations operations.

Fig. 3 shows a diagram for explaining the linear array and its phases. The operation executed by each PE depends on the actual phase. So, in this figure, we distinguish the retrieving phase, that coincides with the first step of the learning phase and is depicted in Fig. 3(c). During this step the hidden-layer outputs (which are labelled in Fig. 3 as 1) are computed and stored in AC, in order to obtain the net outputs, from this hidden outputs (2 in Fig. 3). Fig. 3(d) corresponds with a description of PEs for computing the error in the output layer and weight modification step. For the upper subnet, the propagated signal between each PE O' corresponds with the error function; Z corresponds with weight modifications of this subnet and it serves for calculating the error function in the lower subnet (in Fig. 3, these operations are labelled as 1). For the lower subnet (2 in Fig. 3), the propagated signal between each PE O corresponds with the input $\boldsymbol{x}$, that should be stored into a FIFO. For calculating weight modifications Z, we must use AC1 that has been previously calculated.

6. Summary and conclusions

In this work, an NN, based on the BP model, has been introduced for solving the Mixture Problem. We have shown how this model finds the composition of a mixture of the spectra used during the training.

In order to reduce the NN size, orthogonal transformations has been applied, which allows us to optimize the net neuron number (such as in the hidden layer as in the input layer).

An interesting property related with this optimization is the redundant information reduction, which allows to increase the speed of the method and to decrease the execution time and number of floating point operations.

In this work, we also presented the design of a systolic array for this optimized network.

In experimental results we have presented the behaviour of the proposed method with mixture spectra and we have showed their robustness vs. different ratios of mixture's components.

This method may be applied to many other problems in spectroscopy, such as IR and visible spectrum decomposition, with applications in colorimetry, remote sensing of the Earth's surface, environmental control, and many others.

References

[1] W. Lawton, M. Martin, The advanced mixture problem—principles and algorithms. Technical report, IOM384, Jet Propulsion Laboratory, 1985.
[2] E. Barnard, P. Cassasent, Optical neural net for classifying imaging spectrometer, Applied Optics 18 (15) (1989) 3129–3133.
[3] J.C. Díaz, P. Aguayo, P. Gómez, V. Rodellar, P. Olmos, An associative memory to solve the mixture problem in composite spectra, in: Proc. of the 35th Midwest Symposium on Circuits and System, Washington, DC, 1992, pp. 422–428.
[4] R.M. Pérez, P. Martínez, J. Moreno, A.M. Silva, P.L. Aguilar, Adaptive algorithm to solve the mixture problem with a neural networks methodology, in: 3rd International Workshop on Image and Signal Processing, vol. 1, Manchester (UK), 1996, pp. 133–136.
[5] P.P. Kanjilal, D.N. Banerjee, On the application of orthogonal transformation for the design and analysis of feedforward networks, IEEE Transactions on Neural Networks 6 (5) (1995) 1061–1070.
[6] H.A. Malk, A. Moghadddamjoo, Using the Karkhunen–Loève transformation on the back-propagation training algorithm, IEEE Transactions on Neural Networks 2 (1) (1991) 162–165.
[7] A. Levln, T.K. Leen, E. Moody, Fast Pruning Using Principal Components, John Hopkins University Press, Baltimore, MD, 1989.
[8] J.D. Cowan, G. Tesauro, J. Alspector (Eds.), Advances on Neural Information Processing Systems, 6, M. Kaufman Publishers, San Francisco, CA, 1994.
[9] G.H. Golub, C.F. Van Loan, Matrix Computations, John Hopkins University Press, Baltimore, MD, 1989 pp. 150–234.
[10] S.Y. Kung, Digital Neural Networks, PTR Prentice-Hall Inc, Englewood Cliffs, NJ, 1993 pp. 337–362.