NEURAL NETWORKS WITH HYBRID MORPHOLOGICAL/RANK/LINEAR NODES AND THEIR APPLICATION TO HANDWRITTEN CHARACTER RECOGNITION

Lúcio F. C. Pessoa[(1)] *and Petros Maragos*[(1)&(2)]
[(1)] Georgia Institute of Technology, Atlanta, GA 30332 U.S.A.
[(2)] Institute for Language and Speech Processing, Athens , Greece
[lpessoa,maragos]@ee.gatech.edu

ABSTRACT

We propose a general class of multilayer feed-forward neural networks where the combination of inputs in every node is formed by hybrid linear and nonlinear (of the morphological/rank type) operations. For its design we formulate a methodology using ideas from the back-propagation algorithm and robust techniques to circumvent the non-differentiability of rank functions. Experimental results in a problem of handwritten character recognition are described and illustrate some of the properties of this new type of nonlinear systems.

1 INTRODUCTION

Multilayer feed-forward neural networks, or simply neural networks (NNs), represent an important class of nonlinear systems widely used in problems of signal/image processing and pattern recognition. Their applications in signal/image processing usually employ networks with a single output, which are sometimes called NN-filters. Furthermore, adaptive filters and NNs are closely related, so that their adaptation/training can be studied under the same framework [2]. In this sense, the design of a NN-filter corresponds to the training process of its embedded NN. The usefulness of NNs can be efficiently investigated due to the existence of the back-propagation algorithm [5], which represents a generalization of the LMS algorithm. The perceptron, *i.e.*, a linear combiner followed by a nonlinearity of the logistic type, is the standard node structure used in NNs. However, it has been observed that logic operations, which are not well modeled by perceptrons, can be generated by some internal interactions in a neuron [7]. For the sake of a better representation of these internal properties, a possible improvement to this basic model is presented in this paper [1]. We propose the MRL-NNs, a general class of NNs where the combination of inputs in every node is formed by hybrid linear and nonlinear (of the morphological/rank type) operations. The fundamental processing unit of this class of systems is the MRL-filter [3], which is a linear combination between a morphological/rank filter and a linear FIR filter. The MRL-NNs have the unifying property that the characteristics of both multilayer perceptrons (MLPs) and morphological/rank neural networks (MRNNs) [4] are observed in the same system. An important special case of MRNNs is the class of min-max classifiers [8], which can provide classification results comparable to MLPs, but with faster training processes. Next, we formulate a simple and systematic training procedure using ideas from the back-propagation algorithm and robust techniques to circumvent the non-differentiability of rank functions. Our approach to train the morphological/rank nodes is a theoretically and numerically improved version of the method proposed by Salembier [6] to design morphological/rank filters. Finally, we apply the proposed design methodology in a problem of handwritten character recognition, and provide some experimental evidences showing that not only the MRL-NNs can generate similar or better results when compared with the classical MLPs, but also they usually require smaller processing times for training.

[1] This work was partially supported by the US National Science Foundation under grant MIP–94-21677, and by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), Brasília, Brazil, through a Doctoral Fellowship under grant 200.846/92-2.

2 THE MRL-NNs

In general terms, a (multilayer feed-forward) NN is a layered system composed by similar nodes, with some of them non-observable (hidden), where the node inputs in a given layer depend only on the node outputs from the preceding layer. Every node performs a generic composite operation, where an input to the node is first processed by some function $h(\cdot\,,\cdot)$ of the input and internal weights, and then transformed by an activation function $f(\cdot)$. The node structure is defined by the function h. In the case of MLPs, h is a linear combination. The activation function f is usually employed for rescaling purposes. A general NN is formally defined by the following set of recursive equations.

$$\begin{aligned} \underline{y}^{(l)} \equiv F(\underline{z}^{(l)}) = (f(z_1^{(l)}), f(z_2^{(l)}), \cdots, f(z_{N_l}^{(l)})) \ , \\ l = 1, 2, \cdots, L \ , \\ z_n^{(l)} \equiv h(\underline{y}^{(l-1)}, \underline{w}_n^{(l)}) \ , \ n = 1, 2, \cdots, N_l \ , \end{aligned} \tag{1}$$

where l is the layer number, and N_l is the number of nodes in layer l. The weight vectors $\underline{w}_n^{(l)}$ represent the tuning parameters in the system. Besides this, the input and output of the system are

$$\begin{array}{ll} \underline{y}^{(0)} = \underline{x} = (x_1, x_2, \cdots, x_{N_0}) & \text{(input)} \\ \underline{y}^{(L)} = \underline{y} = (y_1, y_2, \cdots, y_{N_L}) & \text{(output)} \end{array} \tag{2}$$

The MRL-NN is the system defined by (1) and (2) such that

$$\begin{array}{c} z_n^{(l)} \equiv \lambda_n^{(l)} \alpha_n^{(l)} + (1 - \lambda_n^{(l)}) \beta_n^{(l)} \\ \alpha_n^{(l)} = \mathcal{R}_{r_n^{(l)}}(\underline{y}^{(l-1)} + \underline{a}_n^{(l)}) \\ \beta_n^{(l)} = \underline{y}^{(l-1)} \cdot (\underline{b}_n^{(l)})' + \tau_n^{(l)} \end{array} \tag{3}$$

where $\lambda_n^{(l)},\ \tau_n^{(l)} \in \mathbb{R}$; $\underline{a}_n^{(l)},\ \underline{b}_n^{(l)} \in \mathbb{R}^{N_{l-1}}$; and "$'$" denotes transposition.

$\mathcal{R}_r(\underline{t})$ is the r-th rank function of the vector $\underline{t} \in \mathbb{R}^n$. It is evaluated by sorting the components of $\underline{t} = (t_1, t_2, \cdots, t_n)$ in decreasing order, $t_{(1)} \geq t_{(2)} \geq \cdots \geq t_{(n)}$, and picking the r-th element of the sorted list, *i.e.*, $\mathcal{R}_r(\underline{t}) = t_{(r)}$, $r = 1, 2, \cdots, n$.

Observe from (1) and (3) that the underlying function h is an MRL-filter [3] shifted by a threshold $(1-\lambda_n^{(l)})\tau_n^{(l)}$. The variables $\tau_n^{(l)}$ are important when $\lambda_n^{(l)} = 0$. For every MRL-filter, the vector $\underline{b}_n^{(l)}$ corresponds to the coefficients of a linear FIR filter, and the vector $\underline{a}_n^{(l)}$ represents the coefficients of a morphological/rank filter. The variables $r_n^{(l)}$ and $\lambda_n^{(l)}$ are the rank and the mixing parameters, respectively. The resulting weight vector for every node is then defined by

$$\underline{w}_n^{(l)} \equiv (\underline{a}_n^{(l)},\ \rho_n^{(l)},\ \underline{b}_n^{(l)},\ \tau_n^{(l)},\ \lambda_n^{(l)}) \ , \tag{4}$$

where we use a real variable $\rho_n^{(l)}$ instead of an integer rank variable $r_n^{(l)}$ because we will need to evaluate derivatives during the design of MRL-NNs. The relation between $\rho_n^{(l)}$ and $z_n^{(l)}$ will be defined later via a differential equation, and $r_n^{(l)}$ is obtained from $\rho_n^{(l)}$ using [2]

$$r_n^{(l)} \equiv \left\lfloor N_{l-1} - \frac{N_{l-1} - 1}{1 + \exp(-\rho_n^{(l)})} + 0.5 \right\rfloor \ . \tag{5}$$

Two important special cases of MRL-NNs are obtained when f is the identity, called MRL-NN of type I (*e.g.*, MRNN: $\lambda_n^{(l)} = 1\ \forall n, l$), and when f is a nonlinearity of the logistic type, called MRL-NN of type II (*e.g.*, MLP: $\lambda_n^{(l)} = 0\ \forall n, l$).

3 ADAPTIVE DESIGN

Based on the LMS criterion and using ideas from the back-propagation algorithm, we propose a steepest descent method to optimally design general NNs, and then apply it to MRL-NNs. The design goal is to achieve a set of parameters $\underline{w}_n^{(l)}$, $n = 1, 2, \cdots, N_l$, $l = 1, 2, \cdots, L$, such that some cost function is minimized using a supervised procedure. Consider the training set

$$\{(\underline{x}(k), \underline{d}(k))\ ,\ \ k = 0, 1, \cdots, K-1\} \ , \tag{6}$$

where $\underline{d}(k)$ is the desired system output to the training sample $\underline{x}(k)$. From (6) we generate the training sequence [3]

$$(\underline{x}([k] \bmod K), \underline{d}([k] \bmod K))\ ,\ \ k \in \mathbb{Z} \ , \tag{7}$$

by making a periodic extension of (6). Every period of (7) is usually called an *epoch*. A general supervised training algorithm is of the form

$$\begin{array}{c} \underline{w}_n^{(l)}(i+1) = \underline{w}_n^{(l)}(i) + \mu\ \underline{v}_n^{(l)}(i)\ ,\ \mu > 0\ , \\ n = 1, 2, \cdots, N_l\ ;\ l = 1, 2, \cdots, L\ , \end{array} \tag{8}$$

where the positive constant μ controls the tradeoff between stability and speed of convergence, $\underline{v}_n^{(l)} = -\nabla J$, and J is some cost function to be minimized. Let us define the error signal

$$\begin{array}{c} \underline{e}(k) = (e_1(k), e_2(k), \cdots, e_{N_L}(k)) \equiv \\ \underline{d}([k] \bmod K) - \underline{y}(k)\ , \end{array} \tag{9}$$

and the cost function

$$J(i) \equiv \frac{1}{M} \sum_{k=i-M+1}^{i} \xi(k)\ ,\ 1 \leq M \leq K\ , \tag{10}$$

where

$$\xi(k) \equiv \sum_{n=1}^{N_L} e_n^2(k)\ . \tag{11}$$

Based on the steepest descent algorithm, it follows from (8) and (10) that

$$\underline{v}_n^{(l)}(i) = \frac{1}{M} \sum_{k=i-M+1}^{i} \underline{u}_n^{(l)}(k)\ , \tag{12}$$

where

$$\underline{u}_n^{(l)}(k) = -\ \frac{\partial \xi(k)}{\partial \underline{w}_n^{(l)}}\ . \tag{13}$$

If we define the matrices $W^{(l)}$, $V^{(l)}$ and $U^{(l)}$ by

$$\begin{array}{c} W^{(l)} \equiv \begin{pmatrix} \underline{w}_1^{(l)} \\ \underline{w}_2^{(l)} \\ \vdots \\ \underline{w}_{N_l}^{(l)} \end{pmatrix}\ , \\ V^{(l)} \equiv \begin{pmatrix} \underline{v}_1^{(l)} \\ \underline{v}_2^{(l)} \\ \vdots \\ \underline{v}_{N_l}^{(l)} \end{pmatrix}\ ,\ U^{(l)} \equiv \begin{pmatrix} \underline{u}_1^{(l)} \\ \underline{u}_2^{(l)} \\ \vdots \\ \underline{u}_{N_l}^{(l)} \end{pmatrix}\ , \end{array} \tag{14}$$

[2] $\lfloor \cdot \rfloor$ denotes the usual truncation operation, so that $\lfloor \cdot + 0.5 \rfloor$ is the usual rounding operation.

[3] $[k] \bmod K \equiv k - K \lfloor k/K \rfloor$ denotes the index k modulo K.

then the algorithm (8) can be written as

$$W^{(l)}(i+1) = W^{(l)}(i) + \mu\, V^{(l)}(i) \ ,$$
$$V^{(l)}(i) = \frac{1}{M} \sum_{k=i-M+1}^{i} U^{(l)}(k) \ , \qquad (15)$$
$$l = 1, 2, \cdots, L$$

In this way, using the chain rule to evaluate $U^{(l)}(k)$, it can be shown that

$$U^{(l)}(k) = 2\ \mathrm{diag}(\underline{\delta}^{(l)}(k)) \cdot \Gamma^{(l)}(k) \ , \qquad (16)$$

where [4]

$$\underline{\delta}^{(l)}(k) = \underline{e}^{(l)}(k) \odot \dot{F}(\underline{z}^{(l)}(k)) \ , \qquad (17)$$

$$\underline{e}^{(l)}(k) = \begin{cases} \underline{e}(k) & , \ l = L \\ \underline{\delta}^{(l+1)}(k) \cdot \Theta^{(l+1)}(k) & , \ \text{otherwise} \end{cases} \qquad (18)$$

The matrices $\Gamma^{(l)}$ and $\Theta^{(l)}$ are defined by

$$\Gamma^{(l)} = \begin{pmatrix} \underline{\gamma}_1^{(l)} \\ \underline{\gamma}_2^{(l)} \\ \vdots \\ \underline{\gamma}_{N_l}^{(l)} \end{pmatrix} \ , \ \Theta^{(l)} = \begin{pmatrix} \underline{\theta}_1^{(l)} \\ \underline{\theta}_2^{(l)} \\ \vdots \\ \underline{\theta}_{N_l}^{(l)} \end{pmatrix} \ , \qquad (19)$$

where

$$\underline{\theta}_n^{(l)} = \frac{\partial z_n^{(l)}}{\partial \underline{y}^{(l-1)}} \ , \qquad (20)$$

$$\underline{\gamma}_n^{(l)} = \frac{\partial z_n^{(l)}}{\partial \underline{w}_n^{(l)}} \ . \qquad (21)$$

Based on this framework, the design of MRL-NNs can easily be derived. The difficulty is due to the non-differentiability of rank functions, but we can circumvent this problem by using pulse functions as follows [3].

$$\frac{\partial \alpha_n^{(l)}}{\partial \underline{y}^{(l-1)}} = \frac{\partial \alpha_n^{(l)}}{\partial \underline{a}_n^{(l)}} = \underline{c}_n^{(l)} \equiv \frac{Q(\alpha_n^{(l)} \underline{1} - \underline{y}^{(l-1)} - \underline{a}_n^{(l)})}{Q(\alpha_n^{(l)} \underline{1} - \underline{y}^{(l-1)} - \underline{a}_n^{(l)}) \cdot \underline{1}'} \qquad (22)$$

$$\frac{\partial \alpha_n^{(l)}}{\partial \underline{\rho}_n^{(l)}} = s_n^{(l)} \equiv 1 - \frac{1}{N_{l-1}} Q(\alpha_n^{(l)} \underline{1} - \underline{y}^{(l-1)} - \underline{a}_n^{(l)}) \cdot \underline{1}' \ . \qquad (23)$$

In (22) and (23), $Q(\underline{v}) \equiv (q(v_1), q(v_2), \cdots, q(v_n))$, where

$$q(v) \equiv \begin{cases} 1 & , \text{ if } v = 0 \\ 0 & , \text{ if } v \in \mathbb{R} \setminus \{0\} \end{cases} \qquad (24)$$

and $\underline{1} = (1, 1, \cdots, 1)$. To avoid abrupt changes and achieve numerical robustness, we frequently replace the function $q(v)$ by smoothed impulses $q_\sigma(v)$, $\sigma \geq 0$, such as $\exp[-\frac{1}{2}(v/\sigma)^2]$ or $\mathrm{sech}^2(v/\sigma)$.

[4] We denote $\dot{F}(\underline{z}) \equiv (\dot{f}(z_1), \dot{f}(z_2), \cdots, \dot{f}(z_n))$. The symbol '$\odot$' denotes an array (element-by-element) multiplication.

The remaining unknown is $\dot{F}(\cdot)$, that depends on the type of the MRL-NN in use. For the MRL-NN of type I, $F(\underline{z}^{(l)}) = \underline{z}^{(l)}$, so that $\dot{F}(\underline{z}^{(l)}) = \underline{1}$. For the MRL-NN of type II, we will use $f(z) = [1 + \exp(-\eta\, z)]^{-1}$, $\eta \geq 1$, whose derivative is $\dot{f}(z) = \eta\, f(z)[1 - f(z)]$, so that $\dot{F}(\underline{z}^{(l)}) = \eta \underline{y}^{(l)} \odot [\underline{1} - \underline{y}^{(l)}]$.

4 APPLICATION IN OCR

Using the design framework discussed in the previous section, we now describe some experimental results in a problem of optical character recognition (OCR). Our approach is to perform a comparative analysis of MRL-NNs versus MLPs, illustrating some of the characteristics of both systems. We show that the MRL-NNs are a good alternative to MLPs, usually providing equal or better performance with smaller training times.

To do so, we used a large database of handwritten characters provided by the National Institute of Standards and Technology (NIST) [1]. We selected a total of $K = 61,094$ samples of handwritten digits to form our data set. In our simulations, we normalized the feature vectors (64 dimensional Karhunen-Loève transforms) by

$$\underline{x}(k) \rightarrow \frac{1}{2}\left(\frac{\underline{x}(k)}{\max_k \|\underline{x}(k)\|_\infty} + 1\right) \qquad (25)$$

The data set was split such that $45,000$ digits were used for training and the remaining $16,094$ digits were used for testing. The first $15,000$ elements of the training set were used as a validation set during the training process. The training sequence was ordered such that one instance of every digit is presented to the system in each iteration.

After making several tests, we have set a group 12 experiments with 3 different network topologies: 64-N-10 MRL-NNs and MLPs, $N = 5, 10, 20$. This notation indicates a system with 64 inputs, N hidden nodes, and 10 outputs. Two different step sizes were tested: $\mu = 0.005, 0.05$. Every experiment was repeated 5 times with different random initial conditions, and the best result is reported here. Among many possible ways to initialize the systems, and after performing various tests, we initialized the weights randomly in the ranges: $\underline{a}_n^{(l)} : [-0.1, 0.1]$, $r_n^{(l)} : [1, N_{l-1}]$, $b_n^{(l)} : [-1/\sqrt{N_{l-1}}, 1/\sqrt{N_{l-1}}]$, $\tau_n^{(l)} : [-0.1, 0.1]$, $\lambda_n^{(l)} : [0.4, 0.6]$. Further, in order to estimate gradients, we smoothed impulses with $q_\sigma(v) = \exp[-\frac{1}{2}(v/\sigma)^2]$, $\sigma = 0.05$. Due to the size of the training set, we have used the proposed training algorithm with $M = 1$ only. We have tested the case $M > 1$ with a small subset of the training set, but no signanificant improvements were observed. Both MRL-NNs and MLPs were defined using a sigmoid activation function with $\eta = 1$ (MRL-NNs of type II). As usual, the desired system output $\underline{d} = (d_0, d_1, \cdots, d_9)$ was defined by

$$d_n = \begin{cases} 1 & , \underline{x} \leftrightarrow \text{digit n} \\ 0 & , \text{otherwise} \end{cases} \qquad (26)$$

FM $/\|E(\theta)\|/\|R(\theta)\|$						
	MRL5	MLP5	MRL10	MLP10	MRL20	MLP20
Training	11.8/13.2/10.5	9.9/8.8/11.1	**7.4**/6.9/7.8	7.4/7.0/7.7	8.4/7.8/9.0	**7.5**/6.8/8.2
Testing	18.7/22.4/15.0	18.4/19.9/16.9	**11.0**/13.1/8.9	11.1/10.9/11.4	17.4/24.6/10.2	**11.8**/12.8/10.9
Epoch	3	10	**62**	96	9	**88**

Table 1: Figure of merit / mean error rate / mean rejection rate corresponding to the optimal set of weights of best MRL-NNs vs. best MLPs for $\mu = 0.05$.

In the attempt to compare different systems, a figure of merit (FM) was defined as follows

$$FM(t) = \frac{1}{2}\left(\|E(\theta)\| + \|R(\theta)\|\right) \tag{27}$$

where $\theta \in [0,1]$ is the confidence threshold; t is the epoch; E is the error rate (%), computed, for a given θ, as the ratio of the number of misclassified digits over the number of digits that were not rejected during the classification (in a percentage basis); R is the rejection rate (%), computed, for a given θ, as the ratio of the number of rejected digits over the total number of elements in the set under consideration (also in a percentage basis); and

$$\|E(\theta)\| = \frac{1}{10}\sum_{i=0}^{9} E(\frac{i}{10}) \tag{28}$$

$$\|R(\theta)\| = \frac{1}{10}\sum_{i=0}^{9} R(\frac{i}{10}) \tag{29}$$

The training process tends to decrease the figure of merit, and good performance corresponds to small values of FM. A given classification is rejected if the desired n-th output is $d_n = 1$ but the actual n-th output has the property $\max\{\underline{y}\} = y_n < \theta$, where $\underline{y}$ is the system output. An error (misclassification) is obtained when $d_n = 1$ but $\max\{\underline{y}\} \neq y_n$. The error rate is computed excluding the rejected digits.

Using our proposed training algorithm with all the above considerations, we observed that, either for a step size $\mu = 0.005$ or $\mu = 0.05$, the MRL-NNs required a smaller number of iterations than MLPs, and provided similar performances (FMs). Computing the figures of merit of MLPs with equal number of iterations of MRL-NNs, we usually observed better performances of MRL-NNs. Table 1 summarizes some of the results. The best training performance was obtained with a 64-10-10 MRL-NN (MRL10, FM=7.4%). Similar results were obtained with a 64-10-10 MLP (MLP10, FM=7.4%) and a 64-20-10 MLP (MLP20, FM=7.5%), but with a larger number of iterations. These best results were all obtained with $\mu = 0.05$. Due to space limitations, we will present further details of our experiments in a forthcoming longer paper.

References

[1] M. D. Garris, J. L. Blue, G. T. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and Charles L Wilson. NIST form-based handprint recognition system. Technical Report NISTIR 5469, National Institute of Standards and Technology, July 1994.

[2] S. Marcos, O. Macchi, C. Vignat, G. Dreyfus, L. Personnaz, and P. Roussel-Ragot. A unified framework for gradient algorithms used for filter adaptation and neural network training. *Int'l. Journal of Circuit Theory and Applications*, 20:159–200, 1992.

[3] L. F. C. Pessoa and P. Maragos. MRL-Filters: A general class of nonlinear systems and their optimal design for image processing. *IEEE Trans. on Image Processing*, July 1998.

[4] L. F. C. Pessoa and P.Maragos. Morphological/rank neural networks and their adaptive optimal design for image processing. In *Proc. IEEE Int'l Conf. Acoust., Speech, Signal Processing*, volume 6, pages 3399–3402, May 1996.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Willians. *Learning Internal Representations by Error Propagation*, volume 1 of *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, chapter 8, pages 318–362. MIT Press, Cambridge, MA, 1986. D. E. Rumelhart and J. L. McClelland, eds.

[6] P. Salembier. Adaptive rank order based filters. *Signal Processing*, 27:1–25, Apr. 1992.

[7] G. M. Shepherd and R. K. Brayton. Logic operations are properties of computer-simulated interactions between excitable dendritic spines. *Neuroscience*, 21(1):151–165, 1987.

[8] P. Yang and P. Maragos. Min-max classifiers: Learnability, design and application. *Pattern Recognition*, 28(6):879–899, 1995.