Estimating the Number of Hidden Neurons of the MLP Using Singular Value Decomposition and Principal Components Analysis: A Novel Approach

José Daniel A. Santos
IFCE - Industry Department
Av. Contorno Norte, 10
Maracanaú, Ceará, Brazil
jdaniel@ifce.edu.br

Guilherme A. Barreto
UFC - Teleinformatics Engineering
Av. Mister Hull, S/N, Campus of Pici
Fortaleza, Ceará, Brazil
Email: guilherme@deti.ufc.br

Cláudio M. S. Medeiros
IFCE - Industry Department
Av. 13 de Maio, 2081
Fortaleza, Ceará, Brazil
claudiosa@ifce.edu.br

***Abstract*—This paper presents a novel technique to estimate the number of hidden neurons of an MLP classifier. The proposed approach consists in the post-training application of SVD/PCA to the backpropagated error and local gradient matrices associated with the hidden neurons. The number of hidden neurons is then set to the number of relevant singular values or eigenvalues of the involved matrices. Computer simulations using artificial and real data indicate that proposed method presents better results than obtained with the application of SVD and PCA to the outputs of the hidden neurons computed during the forward phase of the MLP training.**

I. Introduction

The complexity of an MLP network with a single hidden layer is related the number of hidden neurons and the corresponding number of adjustable parameters, which for a neural network is measured by the number of synaptic connections. The addition of a hidden neuron increases the number of parameters (including bias) in $(P+1)+M$ terms, where P and M denote the number of input units and output neurons, respectively.

Several authors have been devising strategies to estimate the number of hidden neurons of the MLP neural network. Gómez *et al.* [1] selected neural architectures using a measure of complexity for structures with various Boolean functions, called the complexity of generalization. This method, however, can only be applied to binary data. In Delogu *et al.* [2] an algorithm is proposed which is based on the geometric interpretation of the MLP network, followed by linear programming techniques to determine the number of hidden neurons and the number of synaptic connections. Trenn [3] used the order of function approximation by Taylor polynomials to estimate the number of hidden units in a MLP. This analysis was purely theoretical, without taking into account practical issues as the number of training vectors and the adjustable parameters. The weight pruning method proposed by Medeiros and Barreto [4] eventually led to elimination of redundant hidden neurons in a MLP network. The analysis was based on the cross-correlation between the errors produced by the output neurons and the backpropagated errors associated with the hidden neurons.

Of particular interest to this work, the use of singular value decomposition (SVD) for model selection has been proposed in a few early works [5], [6], [7], [8]. Weigend and Rumelhart [5] used SVD to demonstrate that the effective ranks of the matrices formed by the activations of hidden neurons and the weights of the hidden layer become the same as the solution converges, thus indicating that the learning algorithm (back-propagation using gradient descent) can be finished. More recently, SVD was used to estimate hidden neurons [9] based on the relevant singular values matrix of the activations of hidden neurons.

In this paper we introduce a novel methodology for estimating the number of hidden neurons that provides acceptable performance for a given pattern classification task, with the smallest number of hidden neurons capable of good generalization to unknown data. In this context, in order to estimate the number of hidden neurons of an MLP network with a single hidden layer, we propose the post-training application of SVD/PCA to the matrices of backpropagated errors and local gradient values associated with the hidden neurons. The number of hidden neurons is then set to the number of relevant singular valuesor eigenvalues of the involved matrices. Simulations using artificial and real data are performed to assess the proposed methodology.

The remainder of the paper is organized as follows. In Section 2, the fundamentals of SVD/PCA are described. In Section 3 we introduce a novel SVD/PCA-based methodology to estimate the number of hidden neurons of the MLP network. In Section 4 we present the results and discuss the main issues. The paper is concluded in Section 5.

II. Fundamentals of SVD and PCA

SVD and PCA are related mathematical techniques, both widely used for data compression, feature selection and extraction or model selection in data analysis [10]. Consider the matrix $A \in \mathbb{R}^{m \times n}$. Thus, $A^T A$ is symmetric and can be orthogonally diagonalized. Let $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ be an orthonormal basis of $\mathbb{R}^n$ comprised of the eigenvectors of $A^T A$, and $\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$ the corresponding eigenvalues

978-0-7695-4210-2/10 $26.00 © 2010 IEEE
DOI 10.1109/SBRN.2010.12

IEEE computer society

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DO CEARA. Downloaded on December 20,2022 at 16:20:13 UTC from IEEE Xplore. Restrictions apply.

of A^TA. Hence, for $1 \leqslant i \leqslant n$, we have

$$\|A\mathbf{v}_i\|^2 = (A\mathbf{v}_i)^T A\mathbf{v}_i = \mathbf{v}_i^T(A^TA\mathbf{v}_i) = \mathbf{v}_i^T(\lambda_i\mathbf{v}_i) = \lambda_i, \tag{1}$$

where $\mathbf{v}_i$ is a unit-length eigenvector of A^TA and $\|\cdot\|$ denotes the L_2-norm. According to Eq. (1) the eigenvalues of A^TA are all non-negative.

Assuming that the eigenvalues are arranged in such way that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$, the singular values of A are defined as $\sigma_i = \sqrt{\lambda_i}$, for $1 \leqslant i \leqslant n$. An important property of singular values is that they provide information about the rank of matrix A, denoted by $\rho(A)$. If $A \in \mathbb{R}^{m\times n}$ has r non-zero singular values, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$, with $\sigma_{r+1} = \ldots = \sigma_n = 0$, then $\rho(A) = r$.

Let us now assume that there is a matrix $\Sigma \in \mathbb{R}^{m\times n}$, whose first elements in the main diagonal are the r first singular values of A, and there are two orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{m\times m}$, such that

$$A = U\Sigma V^T = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \tag{2}$$

where the columns of U ($\mathbf{u}_i$) are called the left singular vectors of A and the columns of V ($\mathbf{v}_i$) are the right singular vectors of A.

If Eq. (2) is post-multiplied by A^TU, we obtain $AA^TU = U\Sigma V^TA^TU = U\Sigma\Sigma^T$. By the same token, it can be shown that $A^TAV = V\Sigma^T\Sigma$. Since $\Sigma\Sigma^T$ and $\Sigma^T\Sigma$ are square diagonal matrices with the singular values in the diagonal, and the vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ are the eigenvectors of the matrices AA^T and A^TA, we get the following simplification:

$$AA^T\mathbf{u}_i = {\sigma_i}^2\mathbf{u}_i \quad i = 1, \ldots, m. \tag{3}$$

$$A^TA\mathbf{v}_i = {\sigma_i}^2\mathbf{v}_i \quad i = 1, \ldots, n. \tag{4}$$

For a discussion about PCA, consider initially a data matrix $X \in \mathbb{R}^{m\times n}$, whose columns represent the data vectors and rows represent the features (or attributes). For zero mean vectors, the covariance matrix of X is given by

$$C_x = E[XX^T] \approx \frac{1}{n-1}XX^T, \tag{5}$$

where $C_x \in \mathbb{R}^{m\times m}$ is a positive-definite matrix and $E[\cdot]$ is the expected value operator. The diagonal entries are the variances of each feature, while the off-diagonal entries are the covariances between different pairs of features.

Let us assume that the eigenvalues of C_x are sorted in decreasing order, i.e. $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m \geq 0$. By defining a transformation matrix $S \in \mathbb{R}^{m\times m}$, whose columns are the corresponding eigenvectors $\{\mathbf{s}_i\}_{i=1}^m$ of C_x, $\mathbf{s}_i \in \mathbb{R}^m$, one can build a linear transformation $Y = S^TX$ such that the covariance matrix of Y, denoted by $C_y \in \mathbb{R}^{m\times m}$, is

$$C_y = E[YY^T] = E[(S^TX)(X^TS)] = S^TC_xS. \tag{6}$$

This transformation produces new vectors whose features are statistically uncorrelated and whose variances are equal to the eigenvalues of C_x. The columns of the transformation matrix S are the so-called principal components of X.

Since C_x is a symmetric matrix with $r \leq m$ orthonormal eigenvectors, where r is the rank of the matrix, it can be decomposed as

$$C_x = SC_yS^T = \sum_{i=1}^{r} \lambda_i \mathbf{s}_i \mathbf{s}_i^T. \tag{7}$$

At this stage, it is worth comparing SVD with PCA. If the matrix A in Eqs. (3) and (4) is symmetric, the matrices U and V become the same. Similarly, Eq. (7) is equivalent to the "square" of Eq. (2). So, each eigenvalue in PCA is then the square of its respective singular value in SVD, and the two techniques become equivalents. However, SVD is more general since it can be applied directly to rectangular matrices, without the need to extract the covariance matrix. PCA is a special case of SVD when the matrix is symmetric.

III. The Proposed Methodology

Consider a MLP network with P input nodes, Q hidden neurons and M output neurons. The output of the hidden neuron i at iteration t is given by [11], [12]

$$y_i^{(h)}(t) = \varphi_i\left[u_i^{(h)}(t)\right] = \varphi_i\left[\sum_{j=0}^{P} w_{ij}(t)x_j(t)\right], \tag{8}$$

where w_{ij} is the weight between the j-th input and the i-th hidden neuron, P is the dimension of the input vector $\mathbf{x}$ (excluding the bias) and $\varphi_i(\cdot)$ is a sigmoidal activation function. Let $\tilde{W} \in \mathbb{R}^{Q\times(P+1)}$ be the connection matrix comprised of all the synaptic weights between input units and hidden neurons, represented by

$$\tilde{W} = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \vdots \\ \mathbf{w}_Q^T \end{bmatrix}, \tag{9}$$

where each row $\mathbf{w}_i^T = [w_{i0}\ w_{i1}\ \ldots\ w_{iP}]$ corresponds to the synaptic weights from the $P+1$ input units, including the bias, to the i-th hidden neuron.

Similarly, let $X \in \mathbb{R}^{(P+1)\times N}$ be the matrix whose columns are the N input training vectors:

$$X = [\mathbf{x}(1) \mid \mathbf{x}(2) \mid \ldots \mid \mathbf{x}(N)], \tag{10}$$

where $\mathbf{x}(t) = [x_0(t)\ x_1(t)\ \cdots\ x_P(t)]^T$ denotes the input pattern at iteration t, with $x_0(t) = -1$. Therefore, Eq. (8) can now be written in matrix form as

$$Y^{(h)} = \varphi_i\left[\tilde{W}X\right], \tag{11}$$

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DO CEARA. Downloaded on December 20,2022 at 16:20:13 UTC from IEEE Xplore. Restrictions apply.

where the activation function $\varphi_i[\cdot]$ is applied component-wise to the matrix resulting from the product $\tilde{W}X$. The matrix $Y^{(h)} \in \mathbb{R}^{Q\times N}$ stores the outputs of the Q hidden neurons, computed for all N examples in the training set. In expanded form, the matrix $Y^{(h)}$ is represented as

$$Y^{(h)} = \begin{bmatrix} y_1^{(h)}(1) & y_1^{(h)}(2) & \cdots & y_1^{(h)}(N) \\ y_2^{(h)}(1) & y_2^{(h)}(2) & \cdots & y_2^{(h)}(N) \\ \vdots & \vdots & \vdots & \vdots \\ y_Q^{(h)}(1) & y_Q^{(h)}(2) & \cdots & y_Q^{(h)}(N) \end{bmatrix}. \quad (12)$$

The error-back-propagation phase starts from the output layer, by projecting the errors $e_k^{(o)}(t) = d_k(t) - y_k^{(o)}(t)$ into the hidden layer, where $d_k(t)$ and $y_k^{(o)}(t)$ are the target and output responses of the k-th output neuron, respectively. Let $e_i^{(h)}(t)$ be the back-propagated error for the i-th hidden neuron:

$$e_i^{(h)}(t) = \sum_{k=1}^{M} m_{ki}(t)\delta_k^{(o)}(t), \quad i = 0, \ldots, Q, \quad (13)$$

where m_{ki} is the weight between the i-th hidden neuron and the k-th output neuron. The term $\delta_k^{(o)}(t) = \varphi'_k(t)e_k^{(o)}(t)$ is the local gradient of the k-th output neuron, where $\varphi'_k(t)$ is the derivative of its activation function.

Let $\tilde{M} \in \mathbb{R}^{M\times(Q+1)}$ be the connection matrix formed by all synaptic weights from hidden to output neurons:

$$\tilde{M}^T = [\mathbf{m}_1 \mid \mathbf{m}_2 \mid \cdots \mid \mathbf{m}_M], \quad (14)$$

where each column-vector $\mathbf{m}_k = [m_{k0}\ m_{k1}\ \ldots\ m_{kQ}]^T$ corresponds to the weight vector of the k-th output neuron, including bias.

Consider now $\Delta^{(o)} \in \mathbb{R}^{M\times N}$ as the matrix whose N columns are formed by the local gradients of the M output neurons, built for the N training examples:

$$\Delta^{(o)} = [\boldsymbol{\delta}^{(o)}(1) \mid \boldsymbol{\delta}^{(o)}(2) \mid \ldots \mid \boldsymbol{\delta}^{(o)}(N)], \quad (15)$$

where each vector $\boldsymbol{\delta}^{(o)}(t) = [\delta_1^{(o)}(t)\ \delta_2^{(o)}(t)\ \cdots\ \delta_M^{(o)}(t)]^T$.

Therefore, Eq. (13) can now be written in matrix form as

$$E^{(h)} = \tilde{M}^T\Delta^{(o)}, \quad (16)$$

where the matrix $E^{(h)} \in \mathbb{R}^{(Q+1)\times N}$ stores in its columns the backpropagated errors associated with the neurons of the hidden layer, for the whole training set. The matrix $E^{(h)}$ in its expanded form is represented by

$$E^{(h)} = \begin{bmatrix} e_0^{(h)}(1) & e_0^{(h)}(2) & \cdots & e_0^{(h)}(N) \\ e_1^{(h)}(1) & e_1^{(h)}(2) & \cdots & e_1^{(h)}(N) \\ \vdots & \vdots & \vdots & \vdots \\ e_Q^{(h)}(1) & e_Q^{(h)}(2) & \cdots & e_Q^{(h)}(N) \end{bmatrix}. \quad (17)$$

The first row of $E^{(h)}$ corresponds to the backpropagated errors associated with the biases $m_{k0} = \theta_k^{(o)}$, $k = 1, \ldots, M$. For the purposes of this paper, the first line of $E^{(h)}$ is not necessary and, hence, may be removed.

The local gradients of the hidden neurons are defined as

$$\delta_i^{(h)}(t) = \varphi'_i(t)\sum_{k=1}^{M} m_{ki}(t)\delta_k^{(o)}(t) = \varphi'_i(t)e_i^{(h)}(t), \quad (18)$$

for $i = 0, ..., Q$. The term $\varphi'_i(t)$ is the derivative of the activation function of the hidden neuron *i*. Let $\Phi^{(h)} \in \mathbb{R}^{Q\times N}$ be the matrix formed by all these derivatives for the N training examples:

$$\Phi^{(h)} = [\boldsymbol{\varphi}'(1) \mid \boldsymbol{\varphi}'(2) \mid \cdots \mid \boldsymbol{\varphi}'(N)], \quad (19)$$

where $\boldsymbol{\varphi}'(t) = [\varphi'_1(t)\ \varphi'_2(t)\ \cdots\ \varphi'_i(t)\ \cdots\ \varphi'_Q(t)]^T$. Then, Eq. (18) can be written in matrix form as

$$\Delta^{(h)} = \Phi^{(h)} \odot E^{(h)}, \quad (20)$$

where $\odot$ defines the component-wise multiplication operator of the involved matrices. Thus, the matrix $\Delta^{(h)} \in \mathbb{R}^{Q\times N}$ stores the local gradients of the hidden neurons for all training set. In its expanded form $\Delta^{(h)}$ can be written as

$$\Delta^{(h)} = \begin{bmatrix} \delta_1^{(h)}(1) & \delta_1^{(h)}(2) & \cdots & \delta_1^{(h)}(N) \\ \delta_2^{(h)}(1) & \delta_2^{(h)}(2) & \cdots & \delta_2^{(h)}(N) \\ \vdots & \vdots & \vdots & \vdots \\ \delta_Q^{(h)}(1) & \delta_Q^{(h)}(2) & \cdots & \delta_Q^{(h)}(N) \end{bmatrix}. \quad (21)$$

The two proposed approaches correspond to the application of PCA/SVD to the rows of the matrices $E^{(h)}$ and $\Delta^{(h)}$, as detailed in the following section.

A. Estimating Q: A Novel Approach

The procedure to be described assumes an MLP(P, Q, M) network, fully connected, initially with a large number Q of hidden neurons, resulting in a network very susceptible to data overfitting. The network is trained to achieve the smallest possible value of the mean squared error (ε_{train}) computed at the end of the training procedure using the training data vectors:

$$\varepsilon_{train} = \frac{1}{2N}\sum_{t=1}^{N}\sum_{k=1}^{M}\left[d_k(t) - y_k^{(o)}(t)\right]^2. \quad (22)$$

The network is said to be converged if the variation of ε_{train} ceases after a certain number of training epochs.

The proposed approach starts right after the training phase. The training data vectors must be presented once again to the network. Weight updating is not allowed at this phase. The only objective now is to build the matrices $E^{(h)}$ and $\Delta^{(h)}$. Once these matrices are available, the next step is to compute the covariance matrix for each one of them in order to apply PCA. SVD can be directly applied to the matrices $E^{(h)}$ and $\Delta^{(h)}$. The next step is to extract all the required eigenvalues (PCA) and singular values (SVD), and place them in decreasing order.

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DO CEARA. Downloaded on December 20,2022 at 16:20:13 UTC from IEEE Xplore. Restrictions apply.

Table I
CLASSIFICATION RESULTS OF THE PROPOSED APPROACH FOR 5 DATASETS.

Data	Technique	Matrix	Q,q	N_c	$CR_{train}(\%)$	$CR_{test}(\%)$	ε_{train}	ε_{test}
Initial Architecture	-	-	10	52	100.00	99.58	0.0046	0.0183
	⌈	$Y^{(h)}$	3	17	99.35	98.92	0.0276	0.0536
	PCA	$E^{(h)}$	1	7	69.33	67.33	0.7821	0.8189
Artificial	⌊	$\Delta^{(h)}$	3	17	99.40	99.08	0.0266	0.0403
Dataset	⌈	$Y^{(h)}$	3	17	100.00	99.50	0.0082	0.0236
	SVD	$E^{(h)}$	1	7	69.94	67.75	0.7794	0.8243
	⌊	$\Delta^{(h)}$	3	17	99.94	99.67	0.0108	0.0236
Initial Architecture	-	-	24	243	89.27	81.67	0.3093	0.5042
	⌈	$Y^{(h)}$	4	43	88.53	86.40	0.3226	0.4059
	PCA	$E^{(h)}$	2	23	87.76	86.08	0.3556	0.3761
Spine	⌊	$\Delta^{(h)}$	4	43	90.28	81.61	0.3023	0.4964
Dataset	⌈	$Y^{(h)}$	4	43	90.19	81.24	0.2827	0.5034
	SVD	$E^{(h)}$	2	23	86.85	86.67	0.3556	0.3566
	⌊	$\Delta^{(h)}$	4	43	91.47	78.66	0.2874	0.5373
Initial Architecture	-	-	50	1702	99.98	95.38	0.0026	0.1570
	⌈	$Y^{(h)}$	3	104	99.67	95.88	0.0144	0.1322
	PCA	$E^{(h)}$	1	36	99.08	97.16	0.0333	0.0931
Breast Cancer	⌊	$\Delta^{(h)}$	2	70	99.63	96.37	0.0162	0.1219
Dataset	⌈	$Y^{(h)}$	4	138	99.85	95.44	0.0080	0.1452
	SVD	$E^{(h)}$	1	36	99.22	95.48	0.0284	0.1507
	⌊	$\Delta^{(h)}$	3	104	99.71	95.66	0.0116	0.1400
Initial Architecture	-	-	28	1010	100.00	86.76	0.0007	0.4400
	⌈	$Y^{(h)}$	8	290	100.00	88.78	0.0007	0.3716
	PCA	$E^{(h)}$	1	38	97.76	86.95	0.0882	0.4708
Ionosphere	⌊	$\Delta^{(h)}$	2	74	99.67	89.95	0.0131	0.3464
dataset	⌈	$Y^{(h)}$	8	290	99.99	88.22	0.0014	0.3924
	SVD	$E^{(h)}$	1	38	97.74	86.43	0.0899	0.4855
	⌊	$\Delta^{(h)}$	2	74	99.73	90.00	0.0112	0.3434
Initial Architecture	-	-	30	363	66.87	61.50	0.8342	0.9221
	⌈	$Y^{(h)}$	3	39	65.18	65.04	0.8602	0.8546
	PCA	$E^{(h)}$	2	27	64.94	65.26	0.8698	0.8631
Abalone	⌊	$\Delta^{(h)}$	4	51	65.92	65.08	0.8452	0.8725
Set	⌈	$Y^{(h)}$	3	39	65.14	65.59	0.8622	0.8255
	SVD	$E^{(h)}$	2	27	65.06	65.12	0.8697	0.8746
	⌊	$\Delta^{(h)}$	4	51	65.88	64.79	0.8506	0.8468

The occurrence of eigenvalues (or singular values) of small magnitude indicates relatively small contributions to the total variance of the data under consideration. Hence, the elimination of eigenvalues or singular values of minor relevance should not affect the performance of MLP significantly. Applying this reasoning to the eigenvalues or the singular values of the matrices $E^{(h)}$ and $\Delta^{(h)}$, is equivalent to eliminating the number of redundant hidden neurons.

From the exposed in the previous paragraph, the final step is then to select the $q^* < Q$ largest singular values (or eigenvalues) which are responsible for most of the total variance of the matrices $E^{(h)}$ and $\Delta^{(h)}$. For this purpose an application-dependent threshold $\gamma > 0$ is required, the value of which defines what percentage of the total variance is acceptable for the problem. Thus, the criterion used for pruning the hidden neurons in SVD case is given by

$$q^* = \arg \min_{q=1,\ldots,Q} \left\{ \frac{\sum_{i=1}^{q} \sigma_i^2}{\sum_{j=1}^{Q} \sigma_j^2} \geq \gamma \right\}, \qquad (23)$$

while for PCA it is given by

$$q^* = \arg \min_{q=1,\ldots,Q} \left\{ \frac{\sum_{i=1}^{q} \lambda_i}{\sum_{j=1}^{Q} \lambda_j} \geq \gamma \right\}. \qquad (24)$$

Eq. (23) states that, for a given γ, the number of hidden neurons is to be set to the smallest value of q that satisfies the inequality $\sum_{i=1}^{q} \sigma_i^2 / \sum_{j=1}^{Q} \sigma_j^2 \geq \gamma$. The same reasoning applies to Eq. (24). Once q^* is determined, the network is trained and tested again to validate the new topology.

IV. RESULTS AND DISCUSSIONS

All the simulations to be described were performed for a fully connected MLP(P, Q, M) network trained with standard back-propagation algorithm with momentum. A pattern-by-pattern training mode is adopted with the hyperbolic tangent as activation function for hidden and output neurons. The MLP training/testing procedure is repeated for 30 runs for each data set. The number of training epochs varies between 500 and 1.500, depending on preliminary

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DO CEARA. Downloaded on December 20,2022 at 16:20:13 UTC from IEEE Xplore. Restrictions apply.

evaluations of the learning curve for each data set. The chosen number of epochs should ensure that the network has converged. For each training/testing run, pattern vectors were randomly selected in the proportion of 80% and 20% for training and testing, respectively. The inputs were rescaled to the range $[-1, +1]$. Synaptic weights were randomly initialized in the range $[-0.1, 0.1]$, the momentum parameter was set to $\alpha = 0.75$ and the initial learning rate was set to $\eta_0 = 0.01$, decaying to zero linearly with time. The pruning threshold was set to $\gamma = 0.95$.

The numeric results are shown in the Table I, where N_c is the total number of weights, CR_{train} and CR_{test} denote the classification rates for the training and testing data, respectively, while ε_{train} and ε_{test} are the mean squared errors computed for the training and testing data. Application of PCA/SVD to the matrix $Y^{(h)}$ as proposed by [9], [5] were implemented for the sake of performance comparison.

Artificial Dataset - This data set is composed of two-dimensional vectors, divided into two non-linearly separable classes. The data was artificially generated by [4] for the purpose of visualizing the hyperplanes of each hidden neuron and the resulting global decision surface. A total of 200 pattern vectors were generated and equally divided into the two classes. The initial architecture was an MLP$(2, 10, 2)$.

From Table I we can observe that the application of SVD and PCA to the matrices $Y^{(h)}$ and $\Delta^{(h)}$ suggested the pruning of 7 neurons, resulting in a pruned architecture with $q^* = 3$ hidden neurons. For these cases the network achieved classification rates close to 100% for test data. The hyperplanes of each hidden neuron and the global decision surface for the cases with $Q = 10$ and $q^* = 3$ hidden neurons are shown in Fig. 1.

An odd result was achieved by the application of SVD and PCA to the matrix $E^{(h)}$ resulted in the elimination of 9 neurons, leading to a pruned architecture with poor classification rates. For these cases the matrix $E^{(h)}$ presented a single non-zero singular value, while the covariance matrix of $E^{(h)}$ presented a single non-zero eigenvalue. This is an indication that the columns of $E^{(h)}$ are, indeed, linearly dependent ones. The suggestion of $q = 1$ hidden neuron is then not feasible to this dataset, since it is nonlinearly separable. As a conclusion, the best architectures for this dataset resulted from the application of SVD and PCA to the matrices $Y^{(h)}$ and $\Delta^{(h)}$.

Spine Dataset - This biomedical dataset was kindly provided by *Group of Applied Research in Orthopedics* (GARO) of the *Center of Medical and Surgical Rehabilitation of Massues*, Lyon, France. The problem consists in classifying patients as belonging to one of three categories: normal (100 patients), disk hernia (60 patients) or spondylolisthesis (150 patients). Each patient is represented in the database by six biomechanical attributes derived from the shape and orientation of the pelvis and lumbar spine [13]: pelvic incidence, pelvic tilt, sacral slope, pelvic radius, lumbar lordosis angle and grade of spondylolisthesis.

The initial architecture was an MLP$(6, 24, 3)$. Analyzing the results in Table I we observe that the application of SVD and PCA to the matrices $Y^{(h)}$ and $\Delta^{(h)}$ suggested the elimination of 18 neurons. The application of SVD and PCA to the matrix $E^{(h)}$, however, was even more aggressive, suggesting the elimination of 22 hidden neurons. The interesting issue is that the resulting MLP$(6, 2, 3)$ network achieved classification performance comparable to those achieved by larger architectures, including the ones reported in [13].

UCI Datasets - The proposed approach was further applied to three benchmarking datasets downloaded from the machine learning repository from the University of California at Irvine[1]: breast cancer, ionosphere and abalone. For these datasets we initially trained the following MLP architectures: MLP$(31, 50, 2)$, MLP$(33, 28, 2)$ and MLP$(8, 30, 3)$.

For the breast cancer dataset the results show that the smaller number of hidden neurons was achieved after the application of PCA to the matrix $\Delta^{(h)}$. The application of PCA and SVD to the matrix $E^{(h)}$ suggested the elimination of 49 hidden neurons. The matrix $E^{(h)}$ presented a single non-zero singular value and its covariance matrix presented a unique non-zero eigenvalue. However, unlike the results achieved for the artificial dataset, the classification rate for testing data achieved the highest value with $q^* = 1$. We speculate that this could be an indication that this dataset is indeed linearly separable. To confirm our expectations, we trained a simple perceptron network that achieved an average classification rate of 97.83% for testing data.

For the ionosphere dataset the best result was achieved by the application of PCA and SVD to the matrix $\Delta^{(h)}$, which suggested the pruning of 26 hidden neurons. The application of PCA/SVD to the matrix $E^{(h)}$ suggested $q^* = 1$, but the classification rate of the MLP(33,1,2) with testing data was smaller than that of the MLP(33,2,2). it is worth remembering that the best result must be a trade-off between the smallest number of hidden neurons and the highest classification rate among all the pruned architectures.

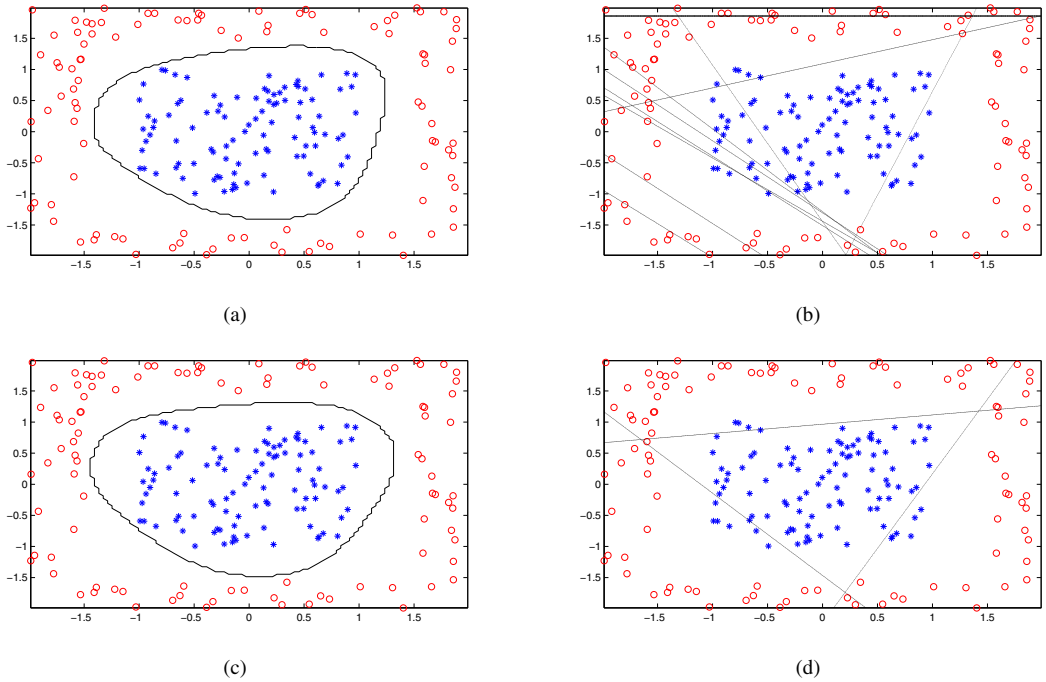
Finally, for the abalone dataset, which is a very hard one to classify, we can observe that there were no significant differences among the classification rates for the testing data for each of the pruned architectures. In this case, one can then choose the pruned architecture with the smallest number of hidden neurons, which was the one resulting from the application of PCA/SVD to the matrix $E^{(h)}$, i.e. MLP(8,2,3).

V. Conclusions

This paper described a simple and efficient method for pruning redundant hidden neurons in a trained MLP network. The method was based on the application of PCA and SVD to the matrices formed by the backpropagated errors and the local gradients of hidden neurons. The magnitudes

[1] http://www.ics.uci.edu/∼mlearn/MLRepository.html

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DO CEARA. Downloaded on December 20,2022 at 16:20:13 UTC from IEEE Xplore. Restrictions apply.



(a) (b) (c) (d)

Figure 1. Decision surface and individual hyperplanes for a trained MLP: (a)-(b) $Q = 10$ and (c)-(d) $q^* = 3$ hidden neurons.

of singular values or eigenvalues served as reference the pruning of redundant neurons in the hidden layer. Computer simulations with artificial and real-world data indicated that, in terms of performance in generalization, the proposed method showed equivalent or better results when compared to application of SVD and PCA to the matrix formed by the activations of hidden neurons of the MLP network.

REFERENCES

[1] I. Gómez, L. Franco, and J. M. Jerez, "Neural network architecture selection: Can function complexity help?" *Neural Processing Letters*, vol. 30, pp. 71–87, 2009.

[2] R. Delogu, A. Fanni, and A. Montisci, "Geometrical synthesis of MLP neural networks," *Neurocomputing*, vol. 71, pp. 919–930, 2008.

[3] S. Trenn, "Multilayer perceptrons: Approximation order and necessary number of hidden units," *IEEE Transactions on Neural Networks*, vol. 19, no. 5, pp. 836–844, 2008.

[4] C. M. S. Medeiros and G. A. Barreto, "Pruning the multilayer perceptron through the correlation of backpropagated errors," in *7th International Conference on Intelligent Systems Design and Applications (ISDA'07)*, 2007, pp. 64–69.

[5] A. Weigend and D. Rumelhart, "The effective dimension of the space of hidden units," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN'91)*, vol. 3, 1991, pp. 2069–2074.

[6] M. Hayashi, "A fast algorithm for the hidden units in a multilayer perceptron," in *International Joint Conference on Neural Networks (IJCNN'93)*, vol. 1, 1993, pp. 339–342.

[7] S. Tamura, M. Tateishi, M. Matsumoto, and S. Akita, "Determination of the number of redundant hidden units in a three-layered feedforward neural network," vol. 1, 1993, pp. 335–338.

[8] D. Psichogios and L. Ungar, "SVD-NET: An algorithm that automatically selects network structure," *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 513–515, 1994.

[9] E. J. Teoh, K. C.Tan, and C. Xiang, "Estimating the number of hidden neurons in a feedforward network using the singular value decomposition," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1623–1629, 2006.

[10] G. Strang, *Linear algebra and its applications*, 4th ed. Brooks Coley, 2005.

[11] J. C. Principe, N. R. Euliano, and W. C. Lefebvre, *Neural and Adaptative Systems: Fundamentals Through Simulations*. John Wiley & Sons, Inc., 2000.

[12] S. Haykin, *Neural networks: a comprehensive foundation*, 2nd ed. Prentice Hall, 1998.

[13] A. R. Rocha Neto and G. A. Barreto, "On the application of ensembles of classifiers to the diagnosis of pathologies of the vertebral column: A comparative analysis," *IEEE Latin America Transactions*, vol. 7, no. 4, pp. 487–496, 2009.

Authorized licensed use limited to: UNIVERSIDADE FEDERAL DO CEARA. Downloaded on December 20,2022 at 16:20:13 UTC from IEEE Xplore. Restrictions apply.